Thirty-Five Dollars

Cromemco

Octart

Asynchronous Communications Processor

Instruction Manual

Cromemco

Octart

Asynchronous Communications Processor

Instruction Manual

February 1984

023-2028

CROMEMCO, Inc.
280 Bernardo Avenue
Mountain View, Ca. 94043

Copyright © 1984
CROMEMCO, Inc.
All Rights Reserved

This manual was produced using a Cromemco System Three computer running under the Cromemco Cromix Operating System. The text was edited with the Cromemco Cromix Screen Editor. The edited text was proofread by the Cromemco SpellMaster Program and formatted by the Cromemco Word Processing System Formatter II. Camera-ready copy was printed on a Cromemco 3355B printer.

The following are registered trademarks of Cromemco, Inc.

Cromemco®
Cromix®
FontMaster®
SlideMaster®
SpellMaster®
System Zero®
System Two®
System Three®
WriteMaster®

The following are trademarks of Cromemco, Inc.

C-10™
C-Net™
CalcMaster™
DiskMaster™
System One™
TeleMaster™

Appendix B is reprinted with permission of the Signetics Corporation, copyright 1983.

TABLE OF CONTENTS

OCTART TECHNICAL SPECIFICATIONS v

OCTART HARDWARE SPECIFICATIONS vii

INTRODUCTION 1

Chapter 1: SETUP AND INSTALLATION 3

Octart ROM 3
Octart Base I/O Address 4
Connectors and Cables 6
Connectors J1 and J2 8
Connector J3 9
Reset Connector 9

Chapter 2: PROGRAMMING INFORMATION 11

Introduction 11
Octart Reset 13
Octart Interrupts 14
Octart/Host Communications 16

LIST OF ILLUSTRATIONS

Figure 1-1: Octart Connectors and Solder Trace 4
Figure 1-2: Octart Switch Settings 5
Figure 1-3: Octart RS-232C Interface 8

LIST OF TABLES

Table 1-1: Host/Octart Communication Ports 5
Table 1-2: Octart Connectors and Pin-Outs 7

Table 2-1: Octart Register Summary 12
Table 2-2: 2681 DUART Register Addressing 13
Table 2-3: Registers after Octart Reset 14

Table B-1: 2681 DUART Register Addressing 33
Table B-2: Baud Rate Generator Characteristics 46
Table B-3: ACR [6:4] Field Definition 46

LIST OF APPENDICES

Appendix A: Octart Register Descriptions 19
Appendix B: 2681 DUART Register Descriptions 33
Appendix C: Six-Bit Transcode 51
Appendix D: ASCII Character Codes 53
Appendix E: EBCDIC Character Codes 55
Appendix F: Parts List 57
Appendix G: The Octart Monitor 61

SCHEMATICS 127

INDEX 133

LIMITED WARRANTY

OCTART TECHNICAL SPECIFICATIONS

Processor:	Z80A, 4 MHz clock
Memory:	Software-selectable configurations: - 64K bytes DRAM, or - 16K bytes ROM, 32K bytes DRAM
Parallel Channels:	One 8-bit input channel; one 8-bit output channel
Parallel Interface:	RS-232 levels
Serial Channels:	8 independent channels
Timer:	4 independent 16-bit counter/timers
Serial Protocols:	Asynchronous Byte
Serial Interface:	RS-232C with circuits TxD, RxD, CTS, and RTS
Channel Data Rates:	Asynchronous, 50 to 38.4K baud with x16 clock multiplier, baud rate varied by choosing a local, crystal-controlled 3.6864-MHz clock by a program-selectable baud rate generator
Character Length:	Transmitter: 1 to 8 bits; Receiver: 5 to 8 bits
Error Detection:	Parity
Interrupts:	Software-controlled, vectored, maskable interrupts from the Octart directed to the host processor; vectored maskable interrupts from DUART channel conditions, the timer, and from host I/O with the Octart registers directed to the Octart Z80A
LSI Device Types:	1 - Z80A CPU (Central Processing Unit) 4 - SCN2681 DUART (Dual Asynchronous Receiver/Transmitter) 8 - 4164 64K x 1-bit DRAM (dynamic RAM) 1 - 27128 16K x 8-bit ROM (user supplied) 1 - Custom CMOS Circuit

Host Interface:	Two bi-directional S-100/IEEE-696 bus I/O ports; S-100/IEEE-696 maskable vectored interrupts
S-100/IEEE-696 Power:	+ 8 VDC @ 1.5 A +18 VDC @ 250 mA -18 VDC @ 250 mA
Operating Environment:	0 - 55 degrees Celsius

OCTART HARDWARE SPECIFICATIONS

Basic Structure

CPU	Z80
ROM	16K; Address = 0000h to 3FFFh (same ROM also appears at ADDR = C000h to FFFFh)
RAM	64K; Address = 0000h to FFFFh
DUART	Signetics SC2681 (24-pin)
S-100 Interface	Similar to Cromemco IOP board.

Communication Ports

1. Eight independent serial channels provided by SC2681 chip

2. One parallel 8-bit handshaking input/output port (RS-232 level)

Bank Switch

ROM can be enabled by:

1. Reset

2. Output 01h to Port 40h

ROM can be disabled by:

1. Output 00h to Port 40h

Reset

Octart can be reset by:

1. Power On

2. S-100 Reset

3. Ground Reset Pin

4. Send 6 bytes of command from S-100 to the command register (Octart input port 00h) of Octart. The commands are 7Eh, 55h, 0Fh, 2Ah and 7Eh. These commands should be sent continuously and can be disabled by setting switch one, position 8 to the OFF position.

I/O Address Assignment

00h Input: Command Register (S-100 Port Base+00h)
Output: Status Register
01h Input: Data Register (S-100 Port Base+01h)
Output: Data Register
02h Input: Flag Register
Output: Control Register
03h Input: Not Used
Output: Interrupt Vector
1Xh DUART Chip 2 (Channel 2 and Channel 3)
2Xh DUART Chip 3 (Channel 4 and Channel 5)
3Xh DUART Chip 4 (Channel 6 and Channel 7)
4Xh Bank (ROM/RAM) Switch
5Xh DUART Chip 1 (Channel 0 and Channel 1)
6Xh Parallel Port (Input/Output)

S-100 Interface Ports (00h to 03h)

Similar to ports of the IOP except for the following bits:

1. Input Port 02h - D2, D4 and D5 are wired to 1.

2. Output Port 00h - D0 is wired to RESET* of Octart (this RESET* is status of Octart).

3. Output Port 02h - Only D0 and D2 are implemented.

4. Input Port 00h - Octart can be reset by receiving a series of commands at this port.

Interrupt Vector

Interrupts can be generated from DUART chips or from the S-100. There is no hardware priority for interrupts that should be taken care of by software. Each source controls one bit of interrupt vector. When an interrupt occurs, the bit which it controls will be 0, otherwise the bit is 1. These bits are:

D0 - always 0;
D1 - S-100;
D2 - Chip 1;
D3 - Chip 2;
D4 - Chip 3;
D5 - Chip 4;
D6 and D7 - always 1.

A dummy S-100 interrupt is pending after reset. This is for power-on diagnostic purposes.

INTRODUCTION

This manual provides installation, operating, and programming instructions for Cromemco's Octart eight-channel communications processor board. The Octart board is a second-generation, co-processing subsystem which interfaces eight serial channels and a bi-directional parallel port to a host S-100/IEEE-696 bus. A typical Octart application might consist of interfacing eight computer terminals to the host system. Unlike earlier serial interface boards, which merely formatted and exchanged individual data characters, the Octart features a sophisticated DUART communications circuit plus an independent Z80A processor with 64K bytes of memory. This enables the Octart to:

1. Perform all protocol and error-detection/recovery functions.

2. Buffer large amounts of serial data.

3. Pass only preprocessed data over the host bus using interrupt-driven I/O.

This reduces the processing load on the host CPU and dramatically increases system throughput.

The Octart is a versatile serial subsystem. Under program control, it can switch its internal memory configuration from 16K bytes of ROM and 32K bytes of RAM to a full 64K bytes of RAM. Thus the board can include a ROM bootstrap program which loads an application program, and then switches to 64K bytes of RAM for maximum buffer space. The eight serial channels can operate independently of one another in any of four modes: full duplex, auto echo, local loopback, and remote loopback. Each channel can be programmed to automatic wake-up mode for multidrop applications.

The Octart board can only be used with version 11.24 (or higher) of the Z80 Cromix Operating System and with version 20.61 (or higher) of the 68000 Cromix Operating System.

Chapter 1 describes how to install the Octart board in a host S-100 bus system. Information provided includes Octart switch settings, connector pin-outs, cables, and S-100 bus interrupt-priority wiring.

Chapter 2 presents Octart programming information. This chapter assumes the reader is familiar with programming in general, and with Z80 Assembly Language in particular (refer to Reference 1). Most of the information in this chapter relates to several Octart registers through which the Z80A processor manages all board functions. Several of these registers are mapped to access internal DUART registers. Detailed programming information for the 2681 DUART is contained in Appendix B and in Reference 2 below.

Positive logic is assumed throughout the manual. That is, logic 0 is associated with a more negative voltage (near 0 VDC), and logic 1 with a more positive voltage (near +4 VDC). **Reset** means logic 0, and **set** means logic 1, as these terms apply to bit states. The * notation appearing after a signal name means that the signal is active in the logic 0 state (e.g., signal RESET* is active when at logic 0 and inactive when at logic 1).

Reference 1: Zilog, Inc., Z80 Assembly Language Programming Manual, 1977.

Reference 2: Signetics, MOS Microprocessor Data Manual, 1982, pp. 1-68 to 1-85.

Chapter 1

SETUP AND INSTALLATION

To set up the Octart board:

1. Make sure there is a ROM device in board socket IC10.

2. Cut a solder trace if you insert a different ROM with an access time of 150 nSec or less (see Figure 1-1).

3. Set switch SW-1 to select an S-100 base I/O port address for the board, as shown in Figure 1-2.

After the board is set up:

1. Insert it in an empty S-100 bus slot with power off.

2. Connect cables between the serial DTE equipment and the Octart.

3. Place the Octart in the S-100 bus interrupt-priority daisy chain.

These steps are explained in the following sections.

OCTART ROM

Whenever the Octart board is reset (many events can reset the board--refer to the section Octart Reset in Chapter 2):

1. The Octart memory configuration is switched to 16K bytes of ROM from 0000h - 3FFFh, and 32K bytes of RAM from 4000h - BFFFh, and

2. The Z80A processor is reset. This means, among other things, that it starts executing whatever program code starts at memory address 0000h.

These facts imply that ROM firmware must be in place to start up the Octart board after a reset.

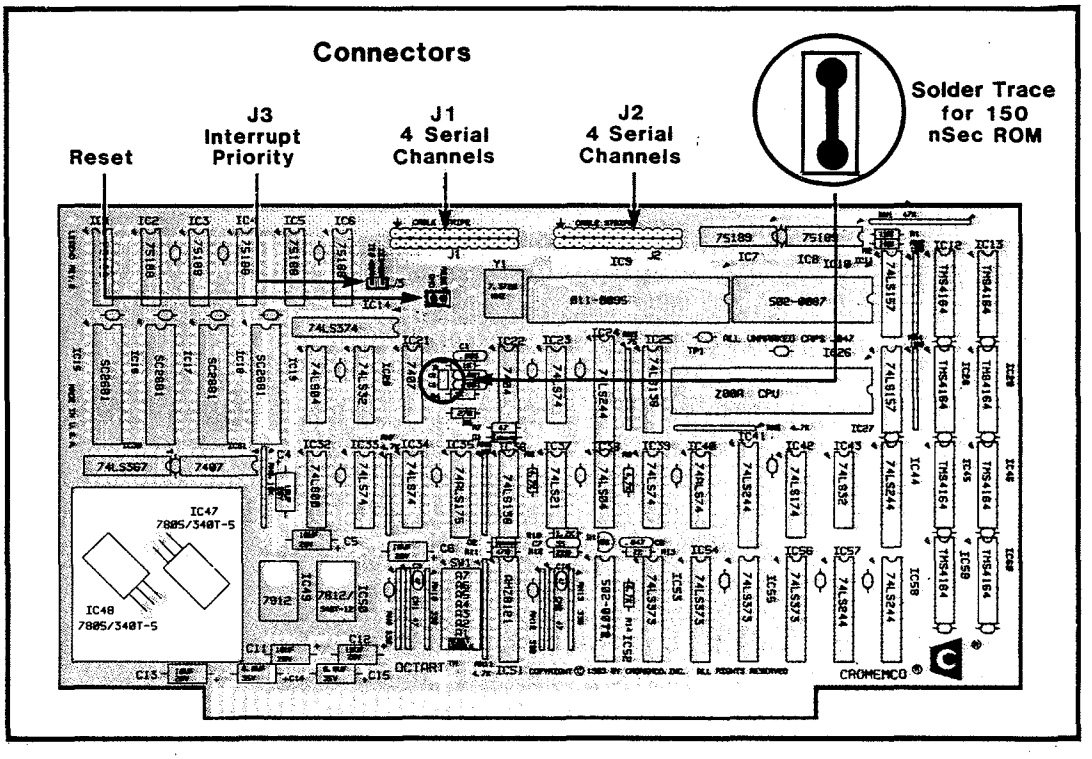


Figure 1-1: OCTART CONNECTORS AND SOLDER TRACE

The Octart board supports the 16K x 8-bit 27128 ROM, or any pin-compatible equivalent. (If you insert a different ROM with an access speed of 150 nSec or less, cut the solder trace shown in Figure 1-1; otherwise, leave the solder trace intact.)

OCTART BASE I/O ADDRESS

The host processor and the Octart communicate through two bi-directional S-100 bus I/O ports. The communication can be either polled or interrupt-driven. Octart switch SW-1 defines the base I/O port number, **Bbase**, which the host uses to access these two ports. Table 1-1 shows these port number assignments. Notice that port addresses may be changed relative to the host processor with switch SW-1, but are fixed relative to the Octart's Z80A processor.

Table 1-1: HOST, OCTART COMMUNICATION PORTS

Register Name	Host		Octart
Commands from Host	OUT Bbase+00h	→	IN 00h
Status to Host	IN Bbase+00h	←	OUT 00h
Data from Host	OUT Bbase+01h	→	IN 01h
Data to Host	IN Bbase+01h	←	OUT 01h

Figure 1-2 shows the location of switch SW-1 and the appropriate switch settings for up to four Octart boards. Octart boards 1, 2, 3, and 4 should have base port addresses CEh, BEh, AEh, and 9Eh.

Note that the IOP and Octart boards share the same port assignments. If your system contains both IOPs and Octarts, they must not use the same addresses.

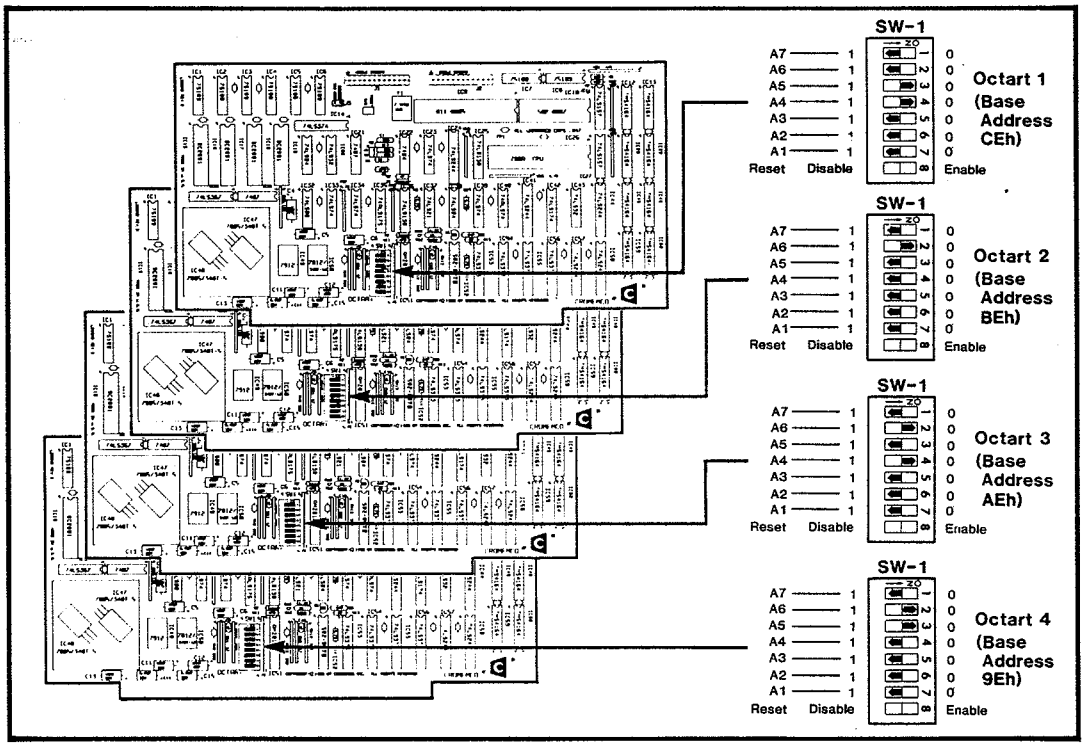


Figure 1-2: OCTART SWITCH SETTINGS

CONNECTORS AND CABLES

There are four connectors on the Octart board (see Figure 1-1): J1 connects to serial channels 0 through 3, and J2 connects to serial channels 4 through 7. J3 connects the Octart in the S-100 interrupt-priority chain, and the RESET connector resets the Octart board.

Table 1-2 defines the pin-outs of all Octart connectors. A dashed table entry (---) denotes no connection (a floating pin). GROUND entries denote a direct connection to S-100 bus GROUND at 0 VDC.

Note that the eight CTS signals and eight RTS signals in Table 1-2 are not standard RS-232 signals. Rather, they are eight individual bits of the Octart parallel I/O register 6Xh. Refer to the description of this register in Appendix A.

Install the Octart board in an empty S-100 bus slot. Do **not** turn on system power until all cabling is installed.

Table 1-2: OCTART CONNECTORS AND PIN-OUTS

PIN	CONNECTOR	
	RESET	J3
1	RESET*	PRIORITY IN*
2	GROUND	PRIORITY OUT*

PIN	CONNECTOR	
	J1	J2
1	---	---
2	Rx0	Rx4
3	Tx0	Tx4
4	CTS0	CTS4
5	RTS0	RTS4
6	---	---
7	GND	GND
8	GND	GND
9	Rx1	Rx5
10	Tx1	Tx5
11	CTS1	CTS5
12	RTS1	RTS5
13	RTS2	RTS6
14	Rx2	Rx6
15	GND	GND
16	Tx2	Tx6
17	---	---
18	---	---
19	CTS2	CTS6
20	---	---
21	Tx3	Tx7
22	RTS3	RTS7
23	Rx3	Rx7
24	CTS3	CTS7
25	GND	GND

Connectors J1 and J2

Connectors J1 and J2 interface serial RS-232C, DTE equipment to the Octart board. Connector J1 attaches to serial channels 0 through 3, and J2 attaches to serial channels 4 through 7. Figure 1-3 shows the relationships among incoming and outgoing voltage levels and logic states.

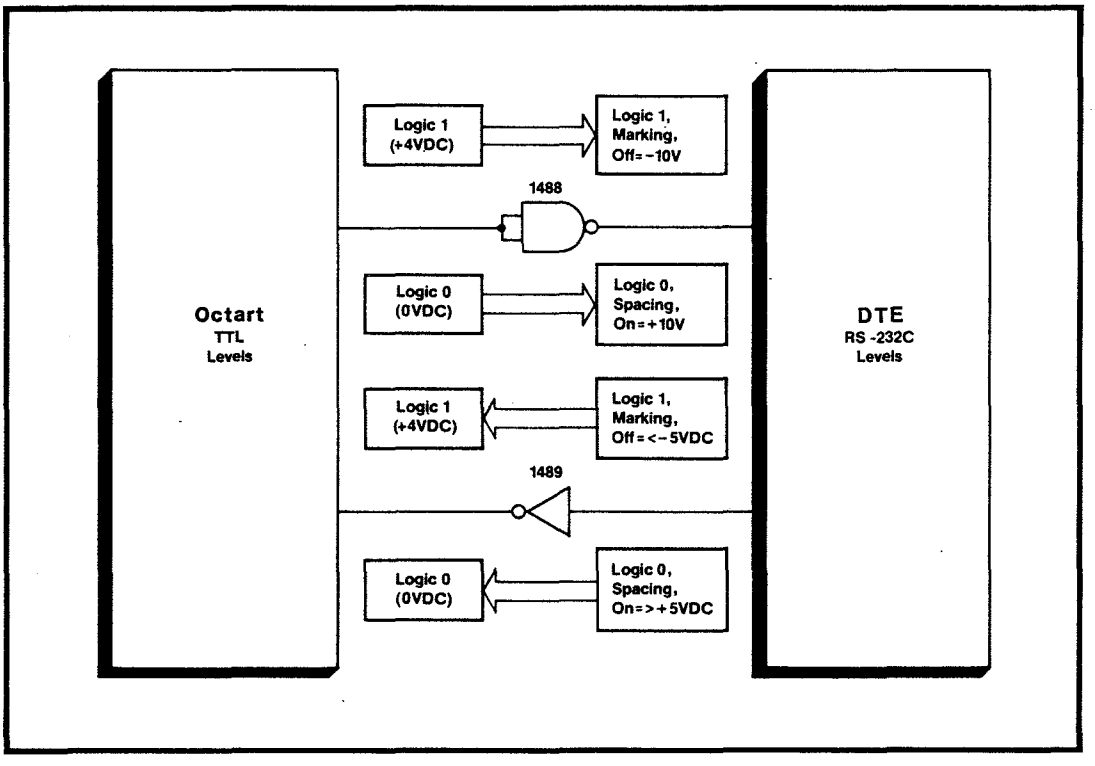


Figure 1-3: OCTART RS-232C INTERFACE

With system power off, route two 25-conductor ribbon cables (Cromemco part number 519-0017, 62 cm long, or 519-0008, 110 cm long) through the computer system housing. Secure the DB-25S socket end of each cable to the system rear housing knockouts with screws and nuts. Clearly mark each ribbon cable with the connector number that appears on the outside of the system back panel. The 26-pin female connectors should comfortably reach connectors J1 and J2 when the Octart board is installed. Align the cable stripe of each cable with the Octart board legend arrowheads, and attach the 26-pin female connectors to Octart connectors J1 and J2.

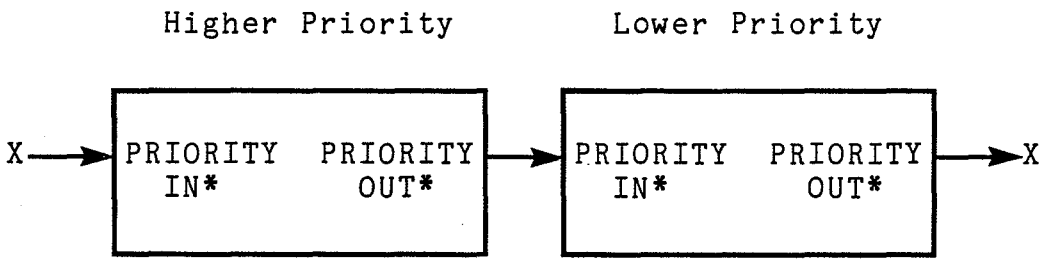
Attach the terminals or other DTE equipment at the system back panel DB-25S sockets. A special cable assembly, CBL-OCT, splits each rear panel connector into

four independent DB-25 connectors. The DB-25 connectors on each CBL-OCT are labeled 1 through 4; these connectors are used for terminals 0 through 3 on connector J1 and terminals 4 through 7 on connector J2.

If a standard cable assembly is used instead of CBL-OCT, only one terminal will be connected (terminal 0 on connector J1, or terminal 4 on connector J2).

Connector J3

Connector J3 is used to place the Octart board in the S-100 interrupt priority daisy chain. Interconnect the Octart board among other S-100 bus boards (use Cromemco part number 519-0029, or an equivalent cable) as shown below. Leave the PRIORITY IN* pin of the highest priority board open, and repeatedly connect OUT* to IN* to form the daisy chain. Also leave the PRIORITY OUT* pin of the lowest priority board open.



The S-100 bus interrupt daisy chain resolves concurrent interrupt requests to the host processor in favor of the highest priority board requesting service. Cromemco recommends that the board priorities be assigned as follows: 64FDC (Highest Priority) -> TU-ART -> IOP -> OCTART -> PRI (Lowest Priority). Do **not** connect the WDI-II hard disk interface board in the interrupt daisy chain.

Reset Connector

Although no pins are installed at this connector location, the connector can be wired to a normally open pushbutton switch, or RESET* can be wired to any output capable of sinking 2 mA to ground (one bit of an output port, for example). Forcing RESET* to logic 0 resets the entire Octart board.

Chapter 2

PROGRAMMING INFORMATION

INTRODUCTION

The Octart is a co-processing I/O management subsystem for the S-100 bus host processor. Application software for this architecture is normally structured as follows:

1. The host to Octart link is made as short, fast, and simple as possible. This means the routine running in host memory should view the Octart as a simple data source or sink, with a minimum amount of status and command information required to carry out a data exchange. Normally, only pre-processed data should travel on the S-100 bus on a vectored, interrupt-driven basis.

2. The routine running in Octart exchanges commands, status, and data with the host processor. To off-load the host processor, the Octart routine should also assume responsibility for connecting and disconnecting serial links; managing serial protocols; formatting or processing the serial data (such as data encryption and decryption); managing the parallel printer; monitoring data integrity; and attempting all possible error recovery procedures for the host processor.

The Octart program store is 16K bytes of ROM from 0000h - 3FFFh, and 32K bytes of RAM from 4000h - BFFFh, immediately after an Octart reset. Under program control, the memory configuration can later be switched to 64K bytes (0000h - FFFFh).

The Octart Z80A processor manages all board functions through several I/O-mapped registers, which are listed in Table 2-1. Detailed descriptions for these registers appear in Appendix A and Appendix B.

Table 2-1: OCTART REGISTER SUMMARY

OCTART PORT	HOST PORT	REGISTER FUNCTION
IN 00h	OUT Bbase+00h	Commands from Host
OUT 00h	IN Bbase+00h	Status to Host
IN 01h	OUT Bbase+01h	Data from Host
OUT 01h	IN Bbase+01h	Data to Host
IN 02h	---	OCTART Flags
OUT 02h	---	OCTART Control
OUT 03h	---	Interrupt Vector To Host
1Xh	---	2681 DUART #2 (Channel 2 & 3)
2Xh	---	2681 DUART #3 (Channel 4 & 5)
3Xh	---	2681 DUART #4 (Channel 6 & 7)
OUT 4Xh	---	ROM/RAM Bank Switch
5Xh	---	2681 DUART #1 (Channel 0 & 1)
IN 6Xh	---	Parallel Input (handshaking)
OUT 6Xh	---	Parallel Output (handshaking)

Table 2-2 lists the 2681 DUART registers addressed by the lower four bits of Octart registers 1Xh, 2Xh, 3Xh, and 5Xh. In all DUART register descriptions in this manual, Channel A refers to the even-numbered channels: Channel 0 (port 5Xh), Channel 2 (port 1Xh), Channel 4 (port 2Xh), and Channel 6 (port 3Xh); Channel B refers to the odd-numbered channels: Channel 1 (port 5Xh), Channel 3 (port 1Xh), Channel 5 (port 2Xh), and Channel 7 (port 3Xh). Appendix B contains detailed register descriptions and programming information for the four 2681 DUARTs.

Table 2-2: 2681 DUART REGISTER ADDRESSING

OCTART PORT	READ (RDN=0)	WRITE (WRN=0)
X0h	Mode Register A (MR1A,MR2A)	Mode Register A (MR1A,MR2A)
X1h	Status Register A (SRA)	Clock Select Reg. A (CSRA)
X2h	*Reserved*	Command Register A (CRA)
X3h	RX Holding Register A (RHRA)	TX Holding Register A (THRA)
X4h	Not Connected	Aux. Control Register (ACR)
X5h	Interrupt Status Reg. (ISR)	Interrupt Mask Reg. (IMR)
X6h	Counter/Timer Upper (CTU)	C/T Upper Register (CTUR)
X7h	Counter/Timer Lower (CTL)	C/T Lower Register (CTLR)
X8h	Mode Register B (MR1B,MR2B)	Mode Register B (MR1B,MR2B)
X9h	Status Register B (SRB)	Clock Select Reg. B(CSRB)
XAh	*Reserved*	Command Register B (CRB)
XBh	RX Holding Register B (RHRB)	TX Holding Register B (THRB)
XCh	*Reserved*	*Reserved*
XDh	Not Connected	Not Connected
XEh	Start Counter Command	Not Connected
XFh	Stop Counter Command	Not Connected

OCTART RESET

Several events can reset the Octart board. They are:

1. Applying power to the Octart board. Octart Power On Clear (POC) circuitry generates a momentary active low pulse on line RESET* (IC22 pin 8) whenever S-100 bus lines 1 and 51 go from 0 VDC to +8 VDC.

2. An S-100 bus reset. This occurs whenever S-100 bus line 75, pRESET*, pulses active low. The reset condition persists as long as pRESET* is held low.

3. Forcing the RESET* pin of the Octart RESET connector active low. This can be done by either shorting pins RESET* and GROUND together with a normally-open pushbutton switch, or by driving pin RESET* active low with any output capable of sinking 2 mA @ +0.4 VDC or less. Pin RESET* must go inactive high again to remove the reset condition.

4. A software-controlled reset. If Octart switch SW-1, section 8 is ON (see Figure 1-1), then the host can reset the Octart board by sending the following six bytes to register **Commands From Host:** 7Eh, 55h, 0Fh, 70h, 2Ah, and 7Eh. These bytes cause a momentary reset condition. If this command sequence is altered in any way, no Octart reset occurs. To disable the software feature, throw section 8 of switch SW-1 OFF. Only D0 through D6 are used. D7 is not used for software reset.

All four of these events force line RESET* on the internal Octart control bus to go active low. This event is called an **Octart reset** throughout this manual. An Octart reset initializes the Octart board as follows:

1. The Octart memory configuration is unconditionally switched to 16K bytes of ROM from 0000h - 3FFFh, and 32K bytes of RAM from 4000h - BFFFh.

2. The Octart Z80A is reset. This means Z80A maskable interrupts are disabled, the I-register is initialized to 00h, the R-register is initialized to 00h, interrupt mode IM0 is selected, and the Z80A automatically starts executing program code at 0000h as soon as the reset condition is removed.

3. Selected Octart register bits are forced either set (logic 1) or reset (logic 0), as shown in Table 2-3. All other Octart register bits are unaffected.

Table 2-3: REGISTERS AFTER OCTART RESET

OCTART PORT	OUT 4Xh	OUT 00h	IN 02h	OUT 02h
D7	---	1	1	---
D6	---	0	0	---
D5	---	0	---	---
D4	---	0	---	---
D3	---	0	---	---
D2	---	0	---	0
D1	---	1	1	---
D0	1	RESET*	0	0

OCTART INTERRUPTS

For maximum system throughput, all I/O (both between the host and Octart, and between the Octart and its peripherals) should be interrupt-driven. There are two categories of Octart interrupts: internal Octart interrupts and host interrupts. The following paragraphs discuss each category.

Internal Octart Interrupts

The Octart's Z80A processor can be interrupted by a variety of sources on the board itself. The maskable interrupts issued by these sources, and directed to the Octart's Z80A (in contrast to those directed to the host processor) are collectively termed **internal Octart interrupts.** All internal Octart interrupt sources have the same priority.

DUART Interrupt Requests - The DUART can be programmed to issue interrupt requests to the Octart Z80A on a variety of channel conditions. These conditions include Tx Buffer Empty, Rx Character Available, Break Condition, and Timer Ready. (Refer to the section Octart Hardware Specifications for interrupt-vector information.)

Host I/O with the Octart Registers - If bit **Enable Octart Interrupts** of register **Octart Control** is set, then a maskable interrupt request is automatically issued to the Octart Z80A whenever the host processor either reads from, or writes to, the four Octart registers to which it has access (**Data To Host, Data From Host, Status To Host,** and **Commands From Host**). When the Z80A acknowledges the interrupt request, on-board circuitry automatically supplies interrupt vector **FCh** to the processor. (Refer to the section Octart Hardware Specifications for interrupt-vector information.) This feature provides a convenient means to alert the Octart Z80A that the host has either read data or status, or has written data or a command to the Octart. The interrupt service routine for this interrupt source should read register **Octart Flags** to determine which of these events has occurred, and take appropriate action. If bit **Enable Octart Interrupts** is reset, then this feature is disabled.

Host Interrupts

The Octart can quickly alert the host processor that it has status or receive data available by sending it a maskable interrupt request. The Octart does this by setting bit **Enable Host Interrupts** in register **Octart Control**, and then writing an interrupt vector to register **Interrupt Vector To Host.** When the vector is written to this register, on-board circuitry automatically drives S-100 bus line pINT* active low, which, in turn, relays the interrupt request to the host processor. When the host acknowledges the request, the

contents of register **Interrupt Vector To Host** are automatically placed on the S-100 bus Data In lines to vector the host processor to an appropriate service routine, and the interrupt request is removed. If bit **Enable Host Interrupts** is reset, then writing to register **Interrupt Vector To Host** is a null operation.

OCTART/HOST COMMUNICATIONS

The Octart Z80A and the host processor communicate through two bi-directional S-100 bus ports, normally using interrupt-driven I/O. This communication falls into four categories: host commands to the Octart, Octart status to the host, host data to the Octart, and Octart data to the host.

Host Commands to the Octart

The host sends an 8-bit command to the Octart by first polling bit **Command From Host Empty** of register **Status To Host.** If this bit is set, then the host can output a command byte to register **Commands From Host.** If status bit **Commands From Host Empty** is reset, this implies that the Octart has not read the previous command, and the host processor should hold off writing a new one; otherwise, the previous command byte will be overwritten.

The Octart determines that a new command byte from the host is available in register **Commands From Host** by polling bit **Command From Host Available** of register **Octart Flags.** This bit is set whenever the host writes a command to register **Commands From Host.** After reading a command from this register, the Octart must write a logic 1 to bit **Command From Host Empty** of register **Status To Host**; it is **not** automatically reset when the Octart reads a command. This action both sets status bit **Command From Host Empty** and resets flag bit **Command From Host Available.**

The meaning of individual command bits is completely defined by the software design, with the sole exception of the command sequence used to reset the Octart hardware.

Octart Status to the Host

The Octart sends four status bits to the host by first polling bit **Status To Host Empty** of register **Octart Flags.** If this bit is set, then the host can output a status byte to register **Status To Host.** Only bits D5 through D2 (**Status 5** through **Status 2**, respectively) are passed to the host; except for bit **Command From Host Empty** (see above), all other bit states written are irrelevant (they are hardware controlled). If status bit **Status To Host Empty** is reset, the host has not read the previous status word, and the Octart should hold off writing a new one; otherwise, the previous status word will be overwritten.

There is no fixed status bit available to alert the host that a new status word from the Octart is available. Normally, one of the four status bits is set aside for this handshaking function. Again, the meaning of individual status bits is completely defined by the software design. When the host reads register **Status To Host**, flag bit **Status To Host Empty** is automatically set for the next status exchange.

Host Data to the Octart

The host sends eight bits of data to the Octart for serial or parallel transmission by first polling bit **Data From Host Empty.** If this bit is set, then the host can output a data byte to register **Data From Host.** If status bit **Data From Host Empty** is reset, this implies that the Octart has not read the previous data byte, and the host processor should hold off writing a new one; otherwise, the previous data byte will be overwritten.

The Octart determines that a new data byte from the host is available in register **Data From Host** by polling bit **Data From Host Available** of register **Octart Flags.** A set bit implies a data byte is available; a reset bit implies the opposite. Reading the data byte from register **Data From Host** sets status bit **Data From Host Empty** for the next data transfer.

Octart Data to the Host

The Octart sends eight bits of received data to the host by first polling bit **Data To Host Empty** of register **Octart Flags.** If this bit is set, then the Octart can output a data byte to register **Data To Host.** If status bit **Data To Host Empty** is reset, this implies that the host has not read the previous data byte, and the Octart should hold off writing a new one; otherwise, the previous data byte will be overwritten.

The host determines that a new data byte from the Octart is available in register **Data to Host** by polling bit **Data to Host Available** of register **Status to Host.** A set bit implies a data byte is available; a reset bit implies the opposite. Reading the data byte from register **Data to Host** sets flag bit **Data to Host Empty** for the next data transfer.

Appendix A

OCTART REGISTER DESCRIPTIONS

Register
COMMANDS FROM HOST

Octart: **IN 00h**
Host: **OUT Bbase+00h**

D7	Command Bit 7 (MSB)
D6	Command Bit 6
D5	Command Bit 5
D4	Command Bit 4
D3	Command Bit 3
D2	Command Bit 2
D1	Command Bit 1
D0	Command Bit 0 (LSB)

The host passes commands to the Octart through this port. The bits are user-defined, so their interpretations depend on the Octart/Host software design. (The command sequence 7Eh, 55h, 0Fh, 70h, 2Ah, 7Eh is reserved for software reset--refer to the section Octart Reset for details.) These bits are not affected by an Octart reset, and should be assumed to be random until a command is written to this port for the first time. Bit **Command From Host Available** of register **Octart Flags** is set to alert the Octart that a command from the host is available in this register for reading. A host write to this port may be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control.** Refer to the section Octart Reset in Chapter 2.

Register
STATUS TO HOST

Octart: **OUT 00h**
Host: **IN Bbase+00h**

D7	Data From Host Empty
D6	Data To Host Available
D5	Status 5
D4	Status 4
D3	Status 3
D2	Status 2
D1	Command From Host Empty
D0	Octart Reset*

This register supplies the host processor with handshake lines for exchanging data with the Octart, and also allows the Octart to supply four software driven status bits to the host. A host read from this port may be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control.**

D7 Data From Host Empty

This bit is hardware-controlled; data output by the Octart to this bit position is ignored. This bit is reset immediately after the host processor writes a data byte to register **Data From Host.** This bit is set (signifying that the host may write another data byte to the Octart) after the Octart reads register **Data From Host** or by an Octart reset.

D6 Data To Host Available

This bit is hardware-controlled; data output by the Octart to this bit position is ignored. This bit is set (signifying that an Octart written data byte is available for host reading) when the Octart writes a data byte to register **Data To Host.** This bit is reset after the host processor reads register **Data To Host**, or by an Octart reset.

D5 Status 5

This bit function is defined by the Octart software. The bit is reset by an Octart reset.

D4 Status 4

This bit function is defined by the Octart software. The bit is reset by an Octart reset.

D3 Status 3

This bit function is defined by the Octart software. The bit is reset by an Octart reset.

D2 Status 2

This bit function is defined by the Octart software. The bit is reset by an Octart reset.

D1 Command From Host Empty

This bit is set (signifying that the host may write a new command byte to the Octart) when the Octart outputs a logic 1 to this bit position or by an Octart reset. This bit is reset as the host processor writes a byte to register **Command From Host.** The Octart must set this bit for the host under program control since the bit is **not** automatically set when the Octart reads register **Command From Host.** Outputting a logic 0 will reset this bit.

D0 Octart Reset*

This bit monitors the state of the line RESET* on the internal Octart control bus. If the host reads this bit reset, the RESET* line is active low (the Octart is in the middle of a reset operation). If the host reads this bit set, the RESET* line is inactive high (there is no reset operation in progress). Outputs to this bit position are null operations.

Register
DATA FROM HOST

Octart: **IN 01h**
Host: **OUT Bbase+01h**

D7	Data Bit 7 (MSB)
D6	Data Bit 6
D5	Data Bit 5
D4	Data Bit 4
D3	Data Bit 3
D2	Data Bit 2
D1	Data Bit 1
D0	Data Bit 0 (LSB)

The host passes eight bits of parallel data to the Octart through this register. These bits are not affected by an Octart reset, and should be assumed to be random until data is written to this register for the first time. Bit **Data From Host Available** of register **Octart Flags** is set to alert the Octart that a data byte from the host is available in this register for reading. When the Octart reads register **Data From Host**, bit **Data From Host Empty** of register **Status To Host** is set to alert the host that it may output a new data byte. If the host writes data to this port before the Octart has read the previous byte, the new data merely overwrites the old. If the Octart reads register **Input Data** before new data is available, the previous byte is merely re-read. A host write to this port may be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control**.

Register
DATA TO HOST

Octart: **OUT 01h**
Host: **IN Bbase+01h**

D7	Data Bit 7 (MSB)
D6	Data Bit 6
D5	Data Bit 5
D4	Data Bit 4
D3	Data Bit 3
D2	Data Bit 2
D1	Data Bit 1
D0	Data Bit 0 (LSB)

The Octart passes eight bits of parallel data to the host processor through this register. These bits are unaffected by an Octart reset, and should be assumed to be random until data is written to this port for the first time. **Status To Host** bit **Data To Host Available** is set to alert the host that a data byte from the Octart is available in this register for reading. When the host reads register **Data To Host**, bit **Data To Host Empty** of register **Octart Flags** is set to alert the Octart that it may output a new data byte. If the Octart writes data to this port before the host processor has read the previous byte, the new data merely overwrites the old. If the host reads register **Data To Host** before new data is available, the previous byte is merely re-read. A host read from this port may be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control.** The Octart may issue a host interrupt request after writing data to the host by writing a vector byte to register **Interrupt Vector To Host** while bit **Enable Host Interrupt Request** of register **Octart Control** is set.

Register
OCTART FLAGS

Octart: **IN 02h**
Host: **No Access**

D7	Data To Host Empty
D6	Data From Host Available
D5	Logic 1
D4	Logic 1
D3	Host Interrupt Pending
D2	Logic 1
D1	Status To Host Empty
D0	Command From Host Available

The Octart reads this register to determine the hardware status of its I/O registers and S-100 bus line pINT*. All **Flag Register** bits are controlled by Octart hardware.

D7 Data To Host Empty

This bit is reset when the Octart writes a data byte to register **Data To Host.** This bit is set (signifying that the Octart may write another data byte to the host processor) immediately after the host reads the **Data To Host** register. This event may, optionally, be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control.** An Octart reset forces this bit set.

D6 Data From Host Available

This bit is set (signifying that a host written data byte is available for Octart reading) immediately after the host has loaded a data byte into register **Data From Host.** This event may, optionally, be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control.** This bit is reset when the Octart reads the **Data From Host** register. An Octart reset forces this bit reset.

D3 Host Interrupt Pending

This bit is set when the Octart is driving S-100 bus line pINT* active low, awaiting maskable interrupt servicing from the host processor. This bit is reset after the host acknowledges the Octart interrupt request.

D1 Status From Host Empty

This bit is set immediately after the host reads register **Status To Host.** This event may optionally be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control.** This bit is reset when the Octart outputs a byte to register **Status To Host.** An Octart reset forces this bit set.

D0 Command From Host Available

This bit is set as the host processor writes a byte to register **Command From Host.** This event may optionally be programmed to generate an internal Octart interrupt by setting bit **Enable Octart Interrupts** of register **Octart Control.** This bit is reset when the Octart sets bit **Command From Host Empty** in register **Status To Host.** Note that the bit is **not** automatically reset when the Octart reads the **Command From Host** register. An Octart reset forces this bit reset.

Register
OCTART CONTROL

Octart: **OUT 02h**
Host: **No Access**

D7	Not Used
D6	Not Used
D5	Not Used
D4	Not Used
D3	Not Used
D2	Enable Octart Interrupts
D1	Not Used
D0	Enable Host Interrupts

D2 Enable Octart Interrupts

If this bit is set, a maskable interrupt request is issued to the Octart Z80A when the host exchanges I/O data with the Octart registers (port Bbase+00h or port Bbase+01h). Interrupt vector FCh, which anticipates either a Z80A interrupt mode IM1 or IM2 response, is automatically placed on the internal Octart data bus during Interrupt Acknowledge. (Refer to the section Octart Hardware Specifications for interrupt-vector information.) Resetting this bit inhibits this feature. This bit is automatically reset by an Octart reset.

D0 Enable Host Interrupts

When this bit is set, S-100 Bus line *pINT is forced active low, thereby interrupting the host processor, each time a vector is loaded into register **Interrupt Vector To Host.** When the host processor acknowledges the interrupt, the contents of register **Interrupt Vector To Host** are placed on the S-100 Data In bus, and the Octart releases line pINT*. Resetting this bit inhibits subsequent interrupts to the host, but it does **not** remove any interrupt request which is pending at the time the bit is reset. An Octart reset both resets this bit and removes any pending interrupt request to the host from the Octart.

Register
INTERRUPT VECTOR TO HOST

Octart: **OUT 03h**
Host: **No Access**

D7	Vector Bit V7 (MSB)
D6	Vector Bit V6
D5	Vector Bit V5
D4	Vector Bit V4
D3	Vector Bit V3
D2	Vector Bit V2
D1	Vector Bit V1
D0	Vector Bit V0 (LSB)

If bit **Enable Host Interrupts** of register **Octart Control** is set, loading a vector into this register issues a maskable interrupt request to the host processor. When the host processor acknowledges the request, the Octart places the contents of this register on the S-100 Data In bus to vector the host processor to an appropriate Octart service routine.

If bit **Enable Host Interrupts** is reset, then Octart outputs to this register are null operations. If there is more than one S-100 interrupt source, Octart interrupt requests are prioritized among the others with S-100 daisy chain interrupt cabling (Octart connector J3). If the host processor is operating in interrupt mode IM0, the Octart would typically output the opcode corresponding to one of eight RST (restart) instructions to this register.

If operating in IM1, the value output is irrelevant since the Z80A defaults to host memory address 0038h for interrupt servicing. If operating in IM2, the Octart would output a byte which, when concatenated with the contents of the host's Z80A I-register, yields the indirect jump address of the service routine in host memory. The contents of this register are unaffected by an Octart reset and should be assumed to be random until written to for the first time.

Register
2681 DUART #2 (Channels 2 and 3)

Octart: 1Xh
Host: No Access

This register controls Channel 2 (A) and Channel 3 (B). Table B-1 lists the 2681 DUART registers addressed by the lower four bits of Octart register 1Xh.

Refer to Appendix B for detailed 2681 DUART register descriptions and programming information.

Register
2681 DUART #3 (Channels 4 and 5)

Octart: 2Xh
Host No Access

This register controls Channel 4 (A) and Channel 5 (B). Table B-1 lists the 2681 DUART registers addressed by the lower four bits of Octart registers 2Xh.

Refer to Appendix B for detailed 2681 DUART register descriptions and programming information.

Register
2681 DUART #4 (Channels 6 and 7)

Octart: 3Xh
Host: No Access

This register controls Channel 6 (A) and Channel 7 (B). Table B-1 lists the 2681 DUART registers addressed by the lower four bits of Octart registers 3Xh.

Refer to Appendix B for detailed 2681 DUART register descriptions and programming information.

Register
CONFIGURE MEMORY

Octart: **OUT 4Xh**
Host: **No Access**

D7	Not Used
D6	Not Used
D5	Not Used
D4	Not Used
D3	Not Used
D2	Not Used
D1	Not Used
D0	RAM/ROM*

Bit D0 of this register is used to control the Octart memory configuration. The port address of register **Configure Memory** is 4Xh; that is, the most significant hex digit must be a 4, while the least significant hex digit may be any value (0h through Fh).

D0 RAM/ROM*

An Octart reset forces this bit set. If this bit is set, Octart on-board memory is configured as follows:

C000h - FFFFh (16K bytes): ROM
4000h - BFFFh (32K bytes): RAM
0000h - 3FFFh (16K bytes): ROM

Note: These two 16K byte blocks of ROM are **not** independent. The same 16K byte ROM chip is mapped into both low memory (0000h - 3FFFh) and high memory (C000h - FFFFh).

If bit RAM/ROM* is reset, Octart on-board memory is configured as follows:

0000h - FFFFh (64K bytes): RAM

Register
2681 DUART #1 (Channel 0 and 1)

Octart: 5Xh
Host: No Access

This register controls Channel 0 (A) and Channel 1 (B). Table B-1 lists the 2681 DUART registers addressed by the lower four bits of Octart registers 5Xh.

Refer to Appendix B for detailed 2681 DUART register descriptions and programming information.

Register
PARALLEL INPUT
(RS-232 levels)

Octart: **IN 6Xh**
Host: **No Access**

D7	Bit 7 In, CTS7
D6	Bit 6 In, CTS6
D5	Bit 5 In, CTS5
D4	Bit 4 In, CTS4
D3	Bit 3 In, CTS3
D2	Bit 2 In, CTS2
D1	Bit 1 In, CTS1
D0	Bit 0 In, CTS0

This register reads eight parallel input bits from Octart connectors J1 and J2 in inverted form. The port address of register **Parallel Input** is 6Xh; that is, the most significant hex digit must be a 6, while the least significant hex digit may be any value (0h through Fh).

These bits are used to perform handshaking for data exchange under software control.

Register
PARALLEL OUTPUT
(RS-232 levels)

Octart: **OUT 6Xh**
Host: **No Access**

D7	Bit 7 Out, RTS7
D6	Bit 6 Out, RTS6
D5	Bit 5 Out, RTS5
D4	Bit 4 Out, RTS4
D3	Bit 3 Out, RTS3
D2	Bit 2 Out, RTS2
D1	Bit 1 Out, RTS1
D0	Bit 0 Out, RTS0

This register writes eight parallel output bits to Octart connectors J1 and J2 in inverted form. The digits written to register **Parallel Output** are latched, and are not affected by an Octart reset. The port address of register **Parallel Output** is 6Xh; that is, the most significant hex digit must be a 6, while the least significant hex digit may be any value (0h through Fh).

These bits are used to perform handshaking for data exchange under software control.

Appendix B

2681 DUART REGISTER DESCRIPTIONS

Table B-1 summarizes the 2681 DUART registers addressed by the lower four bits of Octart registers 1Xh, 2Xh, 3Xh, and 5Xh. For each of the four DUART registers, Channel A refers to the even-numbered channel: Channel 0 (port 5Xh), Channel 2 (port 2Xh), Channel 4 (port 3Xh), and Channel 6 (port 4Xh); Channel B refers to the odd-numbered channel: Channel 1 (port 5Xh), Channel 3 (port 2Xh), Channel 5 (port 3Xh), and Channel 7 (port 4Xh).

Table B-1: 2681 DUART REGISTER ADDRESSING

OCTART PORT	READ (RDN=0)	WRITE (WRN=0)
X0h	Mode Register A (MR1A,MR2A)	Mode Register A (MR1A,MR2A)
X1h	Status Register A (SRA)	Clock Select Reg. A (CSRA)
X2h	*Reserved*	Command Register A (CRA)
X3h	RX Holding Register A (RHRA)	TX Holding Register A (THRA)
X4h	Not Connected	Aux. Control Register (ACR)
X5h	Interrupt Status Reg. (ISR)	Interrupt Mask Reg. (IMR)
X6h	Counter/Timer Upper (CTU)	C/T Upper Register (CTUR)
X7h	Counter/Timer Lower (CTL)	C/T Lower Register (CTLR)
X8h	Mode Register B (MR1B,MR2B)	Mode Register B (MR1B,MR2B)
X9h	Status Register B (SRB)	Clock Select Reg. B(CSRB)
XAh	*Reserved*	Command Register B (CRB)
XBh	RX Holding Register B (RHRB)	TX Holding Register B (THRB)
XCh	*Reserved*	*Reserved*
XDh	Not Connected	Not Connected
XEh	Start Counter Command	Not Connected
XFh	Stop Counter Command	Not Connected

PROGRAMMING CONSIDERATIONS

The operation of the DUART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The addressing of the registers is described in Table B-1.

The contents of certain control registers are initialized to zero on RESET. Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems. For example, changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect character. In general, the contents of the MR, the CSR, and the OPCR should only be changed while the receiver(s) and transmitter(s) are not enabled, and certain changes to the ACR should only be made while the C/T is stopped.

Mode registers 1 and 2 of each channel are accessed via independent auxiliary pointers. The pointer is set to MR1x by RESET or by issuing a 'reset pointer' command via the corresponding command register. Any read or write of the mode register while the pointer is at MR1x switches the pointer to MR2x. The pointer then remains at MR2x, so that subsequent accesses are always to MR2x unless the pointer is reset to MR1x as described above.

Mode, command, clock select, and status registers are duplicated for each channel to provide totally independent operation and control.

DUART REGISTER DESCRIPTION

MR1A - CHANNEL A MODE REGISTER 1

MR1A is accessed when the channel A MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRA. After reading or writing MR1A, the pointer will point to MR2A.

MR1A[7] - Channel A Receiver Request-to-Send Control

This bit controls the deactivation of the RTSAN output (OP0) by the receiver. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR1A[7]=1 causes RTSAN to be negated upon receipt of a valid start bit if the channel A FIFO is full. However, OPR[0] is not reset and RTSAN will be asserted again when an empty FIFO position is available. This feature can be used for flow control to prevent overrun in the receiver by using the RTSAN output signal to control the CTSN input of the transmitting device.

MR1A[6] - Channel A Receiver Interrupt Select

This bit selects either the channel A receiver ready status (RXRDY) or the channel A FIFO full status (FFULL) to be used for CPU interrupts. It also causes the selected bit to be output on OP4 if it is programmed as an interrupt output via the OPCR.

MR1A[5] - Channel A Error Mode Select

This bit selects the operating mode of the three FIFOed status bits (FE, PE, received break) for channel A. In the character mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these bits is the accumulation (logical OR) of the status for all characters coming to the top of the FIFO since the last 'reset error' command for channel A was issued.

MR1A[4:3] - Channel A Parity Mode Select

If 'with parity' or 'force parity' is selected, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR1A[4:3]=11, selects channel A to operate in the special multidrop mode described in the Operation section.

MR1A[2] - Channel A Parity Type Select

This bit selects the parity type (odd or even) if the 'with parity' mode is programmed by MR1A[4:3], and the polarity of the forced parity bit if the 'force parity' mode is programmed. It has no effect if the 'no parity' mode is programmed. In the special multidrop mode it selects the polarity of the A/D bit.

MR1A[1:0] - Channel A Bits per Character Select

This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

MR2A - CHANNEL A MODE
REGISTER 2

MR2A is accessed when the channel A MR pointer points to MR2, which occurs after any access to MR1A. Accesses to MR2A do not change the pointer.

MR2A[7:6] - Channel A Mode Select

Each channel of the DUART can operate in one of four modes. MR2A[7:6]=00 is the normal mode, with the transmitter and receiver operating independently. MR2A[7:6]=01 places the channel in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TxDA output.

2. The receive clock is used for the transmitter.

3. The receiver must be enabled, but the transmitter need not be enabled.

4. The channel A TxRDY and TxEMT status bits are inactive.

5. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.

6. Character framing is checked, but the stop bits are retransmitted as received.

7. A received break is echoed as received until the next valid start bit is detected.

8. CPU-to-receiver communication continues normally, but the CPU-to-transmitter link is disabled.

Two diagnostic modes can also be configured, MR2A[7:6]=10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.

2. The transmit clock is used for the receiver.

3. The TxDA output is held high.

4. The RxDA input is ignored.

5. The transmitter must be enabled, but the receiver need not be enabled.

6. CPU-to-transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2A[7:6]=11. In this mode:

1. Received data is relocked and retransmitted on the TxDA output.

2. The receive clock is used for the transmitter.

3. Received data is not sent to the local CPU, and the error status conditions are inactive.

4. The received parity is not checked and is not regenerated for transmission, i.e., transmitted parity bit is as received.

5. The receiver must be enabled.

6. Character framing is not checked, and the stop bits are transmitted as received.

7. A received break is echoed as received until the next valid start bit is detected.

The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character. Likewise, if a mode is deselected, the device will switch out of the mode immediately. An exception to this is switching out of autoecho or remote loopback modes: if the de-selection occurs just after the receiver has sampled the stop bit (indicated in autoecho by assertion of RxRDY), and the transmitter is enabled, the transmitter will remain in autoecho mode until the entire stop bit has been retransmitted.

MR2A[5] - Channel A Transmitter Request-to-Send Control

This bit controls the deactivation of the RTSAN output (OPO) by the transmitter. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR2A[5]=1 causes OPR[0] to be reset automatically one bit time after the characters in the channel A transmit shift register and in the THR, if any, are completely transmitted, including the programmed number of stop bits, if the transmitter is

not enabled. This feature can be used to automatically terminate the transmission of a message as follows:

1. Program auto-reset mode: MR2A[5]=1.

2. Enable transmitter.

3. Assert RTSAN: OPR[0]=1.

4. Send message.

5. Disable transmitter after the last character is loaded into the channel A THR.

6. The last character will be transmitted and OPR[0] will be reset one bit time after the last stop bit, causing RTSAN to be negated.

MR2A[4] - Channel A Clear-to-Send Control

If this bit is 0, CTSAN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSAN (IPO) each time it is ready to send a character. if IPO is asserted (low), the character is transmitted. If it is negated (high), the TxDA output remains in the marking state and the transmission is delayed until CTSAN goes low. Changes in CTSAN while a character is being transmitted do not affect the transmission of that character.

MR2A[3:0] - Channel A Stop Bit Length Select

This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of 9/16 to 1 and 1-9/16 to two bits, in increments of 1/16 bit, can be programmed for character lengths of six, seven, and eight bits. For a character length of five bits, 1-1/16 to two stop bits can be programmed in increments of 1/16 bit. The receiver only checks for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled) in all cases.

If an external 1X clock is used for the transmitter, MR2A[3]=0 selects one stop bit and MR2A[3]=1 selects two stop bits to be transmitted.

MR1B - CHANNEL B MODE REGISTER 1

MR1B is accessed when the channel B MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRB. After reading or writing MR1B, the pointer will point to MR2B.

The bit definitions for this register are identical to the bit definitions for MR1A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

MR2B - CHANNEL B MODE REGISTER 2

MR2B is accessed when the channel B MR pointer points to MR2, which occurs after any access to MR1B. Accesses to MR2B do not change the pointer.

The bit definitions for this register are identical to the bit definitions for MR2A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

CSRA - CHANNEL A CLOCK SELECT REGISTER

CSRA[7:4] - Channel A Receiver Clock Select

This field selects the baud rate clock for the channel A receiver as follows:

CSRA[7:4]	Baud Rate CLOCK=3.6864MHz ACR[7]=0	ACR[7]=1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1,200	1,200
0 1 1 1	1,050	2,000
1 0 0 0	2,400	2,400
1 0 0 1	4,800	4,800
1 0 1 0	7,200	1,800
1 0 1 1	9,600	9,600
1 1 0 0	38.4K	19.2K
1 1 0 1	Timer	Timer
1 1 1 0	not used	not used
1 1 1 1	not used	not used

The receiver clock is always a 16X clock.

CSRA[3:0] - Channel A Transmitter Clock Select

This field selects the baud rate clock for the channel A transmitter. The field definition is as per CSRA[7:4].

The transmitter clock is always a 16X clock.

CSRB - CHANNEL B CLOCK SELECT REGISTER

CSRB[7:4] - Channel B Receiver Clock Select

This field selects the baud rate clock for the channel B receiver. The field definition is as per CSRA[7:4].

The receiver clock is always a 16X clock.

CSRB[3:0] - Channel B Transmitter Clock Select

This field selects the baud rate clock for the channel B transmitter. The field definition is the same as CSRA[7:4].

The transmitter clock is always a 16X clock.

CRA - CHANNEL A COMMAND REGISTER

CRA is a register used to supply commands to channel A. Multiple commands can be specified in a single write to CRA as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

CRA[6:4] - Channel A Miscellaneous Commands

The encoded value of this field may be used to specify a single command as follows:

CRA[6:4]	COMMAND
0 0 0	No command.
0 0 1	Reset MR pointer. Causes the channel A MR pointer to point to MR1.
0 1 0	Reset receiver. Resets the channel A receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.
0 1 1	Reset transmitter. Resets the channel A transmitter as if a hardware reset had been applied.
1 0 0	Reset error status. Clears the channel A Received Break, Parity Error, Framing Error, and Overrun Error bits in the status register (SRA[7:4]). Used in character mode to clear OE status (although RB, PE, and FE bits will also be cleared) and in block mode to clear all error status after a block of data has been received.
1 0 1	Reset channel A break change interrupt. Causes the channel A break detect change bit in the interrupt status register (ISR[2]) to be cleared to zero.
1 1 0	Start break. Forces the TXDA output low (spacing). If the transmitter is empty the start of the break condition will be delayed up to two bit times. If the transmitter is active the break begins when transmission of the character is completed. If a character is

in the THR, the start of the break will be delayed until that character, or any others loaded subsequently are transmitted. The transmitter must be enabled for this command to be accepted.

1 1 1 Stop Break. The TxDA line will go high (marking) within two bit times. TxDA will remain high for one bit time before the next character, if any, is transmitted.

CRA[3] - Disable Channel A Transmitter

This command terminates transmitter operation and resets the TxRDY and TxEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state.

CRA[2] - Enable Channel A Transmitter

Enables operation of the channel A transmitter. The TxRDY status bit will be asserted.

CRA[1] - Disable Channel A Receiver

This command terminates operation of the receiver immediately--a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special multidrop mode is programmed, the receiver operates even if it is disabled. Refer to the Operation section.

CRA[0] - Enable Channel A Receiver

This command enables operation of the channel A receiver. If not in the special wakeup mode, this also forces the receiver into the search for start bit state.

CRB - CHANNEL B COMMAND REGISTER

CRB is a register used to supply commands to channel B. Multiple commands can be specified in a single write to CRB as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

The bit definitions for this register are identical to the bit definitions for CRA, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

SRA - CHANNEL A STATUS REGISTER

SRA[7] - Channel A Received Break

This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received: further entries to the FIFO are inhibited until the RxDA line returns to the marking state for at least one-half a bit time (two successive edges of the internal or external 1x clock).

When this bit is set, the channel A 'change in break' bit in the ISR (ISR[2]) is set. ISR[2] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry can detect breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must persist until at least the end of the next character time in order for it to be detected.

SRA[6] - Channel A Framing Error

This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first stop bit position.

SRA[5] - Channel A Parity Error

This bit is set when the 'with parity' or 'force parity' mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special multidrop mode the parity error bit stores the received A/D bit.

SRA[4] - Channel A Overrun Error

This bit, when set, indicates that one or more characters in the received data stream have been lost. It is set upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a 'reset error status' command.

SRA[3] - Channel A Transmitter Empty (TxEMTA)

This bit will be set when the channel A transmitter underruns, i.e., both the transmit holding register (THR) and the transmit shift register are empty. It is set after transmission of the last stop bit of a character if no character is in the THR awaiting transmission. It is reset when the THR is loaded by the CPU or when the transmitter is disabled.

SRA[2] - Channel A Transmitter Ready (TxRDYA)

This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TxRDY is reset when the transmitter is disabled and is set when the transmitter is first enabled. Characters loaded into the THR while the transmitter is disabled will not be transmitted.

SRA[1] - Channel A FIFO Full (FFULLA)

This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, FFULL will not be reset when the CPU reads the RHR.

SRA[0] - Channel A Receiver Ready (RxRDYA)

This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive

shift register to the FIFO and reset when the CPU reads the RHR, if after this read there are no more characters still in the FIFO.

SRB - Channel B Status Register

The bit definitions for this register are identical to the bit definitions for SRA, except that all status applies to the channel B receiver and transmitter and the corresponding inputs and outputs.

ACR - AUXILIARY CONTROL REGISTER

ACR[7] - Baud Rate Generator Set Select

This bit selects one of two sets of baud rates to be generated by the BRG:

Set 1: 50, 110, 134.5, 200, 300, 600, 1050, 1200, 2400, 4800, 7200, 9600, and 38400 baud.

Set 2: 75, 110, 134.5 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, and 19200 baud.

The selected set of rates is available for use by the channel A and B receivers and transmitters as described in CSRA and CSRB. Baud rate generator characteristics are given in Table B-2.

Table B-2: BAUD RATE GENERATOR CHARACTERISTICS
CRYSTAL OR CLOCK - 3.6864 MHz

NOMINAL RATE (BAUD)	ACTUAL 16X CLOCK (KHz)	ERROR (PERCENT)
50	0.8	0
75	1.2	0
110	1.759	-0.069
134.5	2.153	0.059
150	2.4	0
200	3.2	0
300	4.8	0
600	9.6	0
1050	16.756	-0.260
1200	19.2	0
1800	28.8	0
2000	32.056	0.175
2400	38.4	0
4800	76.8	0
7200	115.2	0
9600	153.6	0
19200	307.2	0
38400	614.4	0

ACR[6:4] - Counter/Timer Mode and Clock Source Select

This field selects the operating mode of the counter/timer and its clock source as shown in Table B-3.

Table B-3: ACR [6:4] FIELD DEFINITION

ACR[6:4]	MODE	CLOCK SOURCE
0 0 0	Counter	External (IP2)
0 0 1	Counter	TXCA - 1X clock of channel A transmitter
0 1 0	Counter	TXCB - 1X clock of channel B transmitter
0 1 1	Counter	Crystal or external clock (X1/CLK) divided by 16
1 0 0	Timer	External (IP2)
1 0 1	Timer	External (IP2) divided by 16
1 1 0	Timer	Crystal or external clock (X1/CLK)
1 1 1	Timer	Crystal or external clock (X1/CLK) divided by 16

ACR[3:0] - Not Used

ISR - INTERRUPT STATUS REGISTER

This register provides the status of all potential interrupt sources. The contents of this register are masked by the interrupt mask register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If

the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the reading of the ISR--the true status will be provided regardless of the contents of the IMR. The contents of this register are initialized to 00 16 when the DUART is reset.

ISR[7] - Not Used

ISR[6] - Channel B Change in Break

This bit, when set, indicates that the channel B receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel B 'reset break change interrupt' command.

ISR[5] - Channel B Receiver Ready or FIFO Full

The function of this bit is programmed by MR1B[6]. If programmed as receiver ready, it indicates that a character has been received in channel B and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel B FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[4] - Channel B Transmitter Ready

This bit is a duplicate of TxRDYB (SRB[2]).

ISR[3] - Counter Ready

In the counter mode, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command.

In the timer mode, this bit is set once each cycle of the generated square wave (every other time that the counter/timer reaches zero count). The bit is reset by a stop counter command. The command, however, does not stop the counter/timer.

ISR[2] - Channel A Change in Break

This bit, when set, indicates that the channel A receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel A 'reset break change interrupt' command.

ISR[1] - Channel A Receiver Ready or FIFO Full

The function of this bit is programmed by MR1A[6]. If programmed as receiver ready, it indicates that a character has been received in channel A and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel A FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[0] - Channel A Transmitter Ready

This bit is a duplicate of TxRDYA (SRA[2]).

IMR - INTERRUPT MASK REGISTER

The programming of this register selects which bits in the ISR cause an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output.

CTUR and CTLR - Counter/Timer Registers

The CTUR and CTLR hold the eight MSBs and eight LSBs, respectively, of the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded into the CTUR/CTLR registers is 0002 16. Note that these registers are write-only and cannot be read by the CPU.

In the timer (programmable divider) mode, the C/T generates a square wave with a period of twice the value (in clock periods) of the CTUR and CTLR. If the value in CTUR or CTLR is changed, the current half-period will not be affected, but subsequent half periods will be. In this mode the C/T runs continuously. Receipt of a start counter command (read with A3-A0=1110) causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR.

The counter ready status bit (ISR[3]) is set once each cycle of the square wave. The bit is reset by a stop counter command (read with A3-A0=1111). The command, however, does not stop the C/T.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR by the CPU. Counting begins upon receipt of a start counter command. Upon reaching terminal count (0000 16), the counter ready interrupt bit (ISR[3] is set. The counter continues counting past the terminal count until stopped by the CPU. The output returns to the high state and ISR[3] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower eight bits of the counter (CTU, CTL) may be read by the CPU. It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower eight bits to the upper eight bits occurs between the times that both halves of the counter are read. However, note that a subsequent start counter command will cause the counter to begin a new count cycle using the values in CTUR and CTLR.

Appendix C

SIX-BIT TRANSCODE

00h	SOH	20h	-
01h	A	21h	/
02h	B	22h	S
03h	C	23h	T
04h	D	24h	U
05h	E	25h	V
06h	F	26h	W
07h	G	27h	X
08h	H	28h	Y
09h	I	29h	Z
0Ah	STX	2Ah	ESC
0Bh	.	2Bh	,
0Ch	<	2Ch	%
0Dh	BEL	2Dh	ENQ
0Eh	SUB	2Eh	ETX
0Fh	ETB	2Fh	HT
10h	&	30h	0
11h	J	31h	1
12h	K	32h	2
13h	L	33h	3
14h	M	34h	4
15h	N	35h	5
16h	O	36h	6
17h	P	37h	7
18h	Q	38h	8
19h	R	39h	9
1Ah	SPACE	3Ah	SYN
1Bh	$	3Bh	#
1Ch	*	3Ch	@
1Dh	US	3Dh	NAK
1Eh	EOT	3Eh	EM
1Fh	DLE	3Fh	DEL

Appendix D

ASCII CHARACTER CODES

00h NUL (CONTROL-@)	20h SPACE	40h @	60h '
01h SOH (CONTROL-A)	21h !	41h A	61h a
02h STX (CONTROL-B)	22h "	42h B	62h b
03h ETX (CONTROL-C)	23h #	43h C	63h c
04h EOT (CONTROL-D)	24h $	44h D	64h d
05h ENQ (CONTROL-E)	25h %	45h E	65h e
06h ACK (CONTROL-F)	26h &	46h F	66h f
07h BEL (CONTROL-G)	27h '	47h G	67h g
08h BS (CONTROL-H)	28h (	48h H	68h h
09h HT (CONTROL-I)	29h)	49h I	69h i
0Ah LF (CONTROL-J)	2Ah *	4Ah J	6Ah j
0Bh VT (CONTROL-K)	2Bh +	4Bh K	6Bh k
0Ch FF (CONTROL-L)	2Ch ,	4Ch L	6Ch l
0Dh CR (CONTROL-M)	2Dh -	4Dh M	6Dh m
0Eh SO (CONTROL-N)	2Eh .	4Eh N	6Eh n
0Fh SI (CONTROL-O)	2Fh /	4Fh O	6Fh o
10h DLE (CONTROL-P)	30h 0	50h P	70h p
11h DC1 (CONTROL-Q)	31h 1	51h Q	71h q
12h DC2 (CONTROL-R)	32h 2	52h R	72h r
13h DC3 (CONTROL-S)	33h 3	53h S	73h s
14h DC4 (CONTROL-T)	34h 4	54h T	74h t
15h NAK (CONTROL-U)	35h 5	55h U	75h u
16h SYN (CONTROL-V)	36h 6	56h V	76h v
17h ETB (CONTROL-W)	37h 7	57h W	77h w
18h CAN (CONTROL-X)	38h 8	58h X	78h x
19h EM (CONTROL-Y)	39h 9	59h Y	79h y
1Ah SUB (CONTROL-Z)	3Ah :	5Ah Z	7Ah z
1Bh ESC (CONTROL-[)	3Bh ;	5Bh [	7Bh {
1Ch FS (CONTROL-\)	3Ch <	5Ch \	7Ch ¦
1Dh GS (CONTROL-])	3Dh =	5Dh]	7Dh }
IEh RS (CONTROL-^)	3Eh >	5Eh ^	7Eh ~
1Fh US (CONTROL-_)	3Fh ?	5Fh _	7Fh DEL

Appendix E

EBCDIC CHARACTER CODES

00h	NUL	2Ah	SM	7Bh	#	C8h	H
01h	SOH	2Bh	CV2	7Ch	@	C9h	I
02h	STX	2Dh	ENQ	7Dh	'	D0h	}
03h	ETX	2Eh	ACK	7Eh	=	D1h	J
04h	PF	2Fh	BEL	7Fh	"	D2h	K
05h	HT	32h	SYN	81h	a	D3h	L
06h	LC	34h	PN	82h	b	D4h	M
07h	DEL	35h	RS	83h	c	D5h	N
09h	RLF	36h	UC	84h	d	D6h	O
0Ah	SMM	37h	EOT	85h	e	D7h	P
0Bh	VT	3Bh	CV3	86h	f	D8h	Q
0Ch	FF	3Ch	DC4	87h	g	D9h	R
0Dh	CR	3Dh	NAK	88h	h	E0h	\
0Eh	SO	3Fh	SUB	92h	k	E2h	S
0Fh	SI	40H	SPACE	93h	l	E3h	T
10h	DLE	4Ah	c	94h	m	E4h	U
11h	DC1	4Bh	.	95h	n	E5h	V
12h	DC2	4Ch	<	96h	o	E6h	W
13h	DC3	4Dh	(	97h	p	E7h	X
14h	RES	4Eh	+	98h	q	E8h	Y
15h	NL	4Fh	|	99h	r	E9h	Z
16h	BS	50h	&	A1h	~	F0h	0
17h	IL	5Ah	!	A2h	s	F1h	1
18h	CAN	5Bh	$	A3h	t	F2h	2
19h	EM	5Eh	;	A4h	u	F3h	3
1Ah	CC	5Ch	*	A5h	v	F4h	4
1Bh	CV1	5Dh	)	A6h	w	F5h	5
1Ch	IFS	5Eh	;	A7h	x	F6h	6
1Dh	IGS	5Fh	^	A8h	y	F7h	7
1Eh	IRS	60h	-	A9h	z	F8h	8
1Fh	IUS	61h	/	C0h	{	F9h	9
20h	DS	6Ah	|	C1h	A		
21h	SOS	6Bh	,	C2h	B		
22h	FS	6Ch	* %	C3h	C		
24h	BYP	6Eh	>	C4h	D		
25h	LF	6Fh	?	C5h	E		
26h	EOB/ETB	79h	`	C6h	F		
27h	PRE/ESC	7Ah	:	C7h	G		

Appendix F

PARTS LIST

Integrated Circuits

Designation	Cromemco Description	Part No.
IC1	1489	010-0077
IC2	1488	010-0076
IC3	1489	010-0077
IC4-6	1488	010-0076
IC7-8	1489	010-0077
IC9	CPU & Memory Support (version 2.0)	011-0095
	2 sockets, 40-pin	017-0006
IC10	27128 Rom	502-0087
	1 socket, 28-pin	017-0071
IC11	74LS157	010-0046
IC12-13	TMS4164 NL-15	011-0079
IC14	74LS374	010-0133
IC15-18	SC2681	011-0106
	4 sockets, 24-pin (.400 wide)	017-0363
IC19	74LS04	010-0066
IC20	74LS32	010-0058
IC21	7407	010-0104
IC22	7404	010-0030
IC23	74LS74	010-0055
IC24	74LS244	010-0100
IC25	74LS138	010-0096
IC26	Z80 CPU	011-0010
	2 sockets, 40-pin	017-0006
IC27	74LS157	010-0046
IC28-29	TMS4164 NL-15	011-0079
IC30	74LS367	010-0108
IC31	7407	010-0104
IC32	74LS00	010-0069
IC33-34	74LS74	010-0055
IC35	74ALS175	010-0358
IC36	74LS138	010-0096
IC37	74LS21	010-0060
IC38	74LS04	010-0066
IC39	74LS74	010-0055
IC40	74ALS74	010-0357
IC41	74LS244	010-0100

Integrated Circuits (continued)

Designation	Cromemco Description	Part No.
IC42	74LS174	010-0097
IC43	74LS32	010-0058
IC44	74LS244	010-0100
IC45-46	TMS4164 NL-15	011-0079
IC47-48	7805/340T-5	012-0001
IC49	7912	012-0124
IC50	7812/340T-12	012-0002
IC51	AMZ8121	010-0328
IC52	82S159	010-NEW
		502-0072
	1 socket, 20-pin	017-0004
IC53-56	74LS373	010-0102
IC57-58	74LS244	010-0100
IC59-60	TMS4164 NL-15	011-0079

Diodes/ Transistors

Designation	Cromemco Description	Part No.
D1,2	1N4148	008-0002
Q1	2N3906	009-0002

Capacitors

Designation	Cromemco Description	Part No.
C1	.005/100V	004-0025
C2,3	.001 CRDC	004-0022
C4-6	10 UF/20V	004-0032
C7	33 CRDC	004-0048
C8	.047/50V	004-0061
C9,10	47 MONO	004-0000
C11-13	10 UF/20V	004-0032
C14,15	6.8 UF/35V	004-0034

Capacitor Networks

Designation	Cromemco Description	Part No.
CN1,2	47 PF SIP 8-pin	005-0000

Resistors

Designation	Cromemco Description	Part No.
R1,2	100 ohm 1/4W	001-0007
R3,4	1 kohm 1/4W	001-0018
R6	270 ohm 1/4W	001-0011
R7	47 ohm 1/4W	001-0003
R8,9	4.7 kohm 1/4W	001-0024
R10	1.2 kohm 1/4W	001-0019
R11	470 ohm 1/4W	001-0014
R12	220 ohm 1/4W	001-0010
R13	22 ohm 1/4W	001-0001
R14	4.7 kohm 1/4W	001-0024

Resistor Networks

Designation	Cromemco Description	Part No.
RN1	47 kohm 9R 10P	003-0079
RN2	100 ohm 4R 8P	003-0001
RN3	75 ohm 4R 8P	003-0080
RN4	100 ohm 4R 8P	003-0001
RN5	4.7 kohm 7R 8P	003-0009
RN6	10 kohm 7R 8P	003-0025
RN7	4.7 kohm 7R 8P	003-0009
RN8	100 ohm 7R 8P	003-0036
RN9,10	330 ohm 4R 8P	003-0004
RN11	4.7 kohm 9R 10P	003-0014
RN12,13	330 ohm 4R 8P	003-0004

Miscellaneous

Designation	Cromemco Description	Part No.
SW1	1 DIP switch, 8-pos	013-0002
	2 #6 lock washer cad	015-0020
	2 #4 lock washer cad	015-0139
	2 spacer 1/4 hex 6-32 x 1/4 al	015-0169
	2 4-40 x 1/4 hex standoff	015-0173
J1,2	2 header, 26 pin straight	017-0030
J3	1 socket, 2 pin	017-0009
	1 PCB Rev. A	020-0135
Y1	1 crystal 7.3728 MHZ	026-0034

Page 0001

Appendix G

THE OCTART MONITOR

```
CROMEMCO Z80 Macro Assembler version 03.08 -                    Feb 21, 1984  15:28:53
*** THE CROMEMCO OCTART MONITOR ***

                ##########################################################################
                ##########################################################################
                ##########################################################################
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0002
*** THE CROMEMCO OCTART MONITOR ***
*** VERSION & SPECIAL MACROS ***

                0024  ;;      Release date: September 15, 1983
                0025  ;;      (Modified from IOP monitor Version 03.00)
                0026
                0027
(0001)          0028  Version:defv    01              ;Version number
(0003)          0029  Release:defv    03              ;Release number
                0030  ;
(FFFF)          0031  True:   Equ     -1              ;True for conditionals
(0000)          0032  False:  Equ     0               ;False for conditionals
(0000)          0033  Test:   defv    False           ;True  = Runs in OCTART RAM
                0034  ;                               ;False = Runs in OCTART ROM
                0035  ;
                0036  ;###############################################################################
                0037  ; Define what physical device will be our AUXiliary console port
                0038  ;###############################################################################
                0039  ;
(0000)          0040  Csp:    defv    False           ;True = Value for AUXdev to be the CSP board
(FFFF)          0041  Spare:  defv    True            ;True = Value for AUXdev to be a spare board
                0042  ;
                0043  ;###############################################################################
                0044  ; Special macro for developmental testing of the monitor in OCTART RAM
                0045  ;###############################################################################
                0046
(0000)          0047      If Test
                0048
                0049  JSys:   OMacro  #Arg            ;Create special JSys macro for testing
                0050          Call    #Arg(2,-1)      ;Execute direct call (No JSys)
                0051          Mend
                0052
                0053  Jr:     OMacro  #Arg1,#Arg2     ;Create special Jr macro for testing
                0054      If "#Arg2" > ""
                0055          Jp      #Arg1,#Arg2     ;Execute Jump instead of Jump relative
                0056      Else
                0057          Jp      #Arg1           ;Execute Jump instead of Jump relative
                0058      EndIf
                0059          Mend
                0060
                0061  TopRam: defl    0bfffh          ;Set pointer to top of RAM below debugger
                0062  Origin: defl    4000h           ;Set origin address for OCTART RAM space
                0063  Default:defl    4000h           ;Default parameter value
                0064
                0065      Else
                0066
(BFFF)          0067  TopRam: defl    0bfffh          ;Set pointer to top of RAM
(0000)          0068  Origin: defl    0000h           ;Set origin address for OCTART ROM space
(4000)          0069  Default:defl    4000h           ;Default parameter value
                0070                                  ;(Also used for the internal RAM test)
                0071      EndIf
                0072
                0073
                0074
                0075
                0076
                0077  ;###############################################################################
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0003
*** THE CROMEMCO OCTART MONITOR ***
*** VERSION & SPECIAL MACROS ***

0078  ;Special macro for filling unused spaces in PROM with OFFh (Rst 38)
0079  ;##############################################################################
0080
0081  Origin:OMacro   #Address
0082  Swath:  defv    [Origin + #Address - $]
0083    If Swath > 0
0084          Rept    Swath
0085          Defb    OFFh            ;Fill in empty spaces
0086          Mend
0087    EndIf
0088    If Swath < 0
0089          MNote Program Origin Address Overlap @ #Address
0090    EndIf
0091          Mend
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21. 1984  15:28:53          Page 0004
*** THE CROMEMCO OCTART MONITOR ***
*** STARTUP & RESTART CONTROL ***

      (0000)      0093          Org     Origin
                  0094  ;#########################################################################
                  0095  OCTMon:; This is the Power-On-Clear startup entry point for the monitor.
                  0096          ; It may also be entered using a RST 0 (NOT JSYS 0) command.
                  0097  ;#########################################################################
                  0098
0000  1846        0099          Jr      StartUp         ;Skip around version info & RST xxh
                  0100
0002  0103        0101          defb    Version,Release ;Version number for user reference
0004  07BF        0102          defw    ConID           ;Pointer to console data
0006  EDBF        0103          defw    UserRam         ;Free space pointer
                  0104
      (FFFF)      0105     If not test
0008              0106          Origin 08h
      (0000)      0107+ Swath:  defv    [Origin + 08h - $]
      (0000)      0108+   If Swath > 0
                  0109+         Rept    Swath
                  0110+         Defb    0FFh            ;Fill in empty spaces
                  0111+         Mend
                  0112+   EndIf
      (0000)      0113+   If Swath < 0
                  0114+         MNote Program Origin Address Overlap @ 08h
                  0115+   EndIf
                  0116     endif
                  0117  ;#########################################################################
                  0118  Rst.08: ; This is a user-definable restart entry point.  When using the
                  0119          ; OCTARTMonitor program this address is used by the monitor for internal
                  0120          ; System calls (JSYS xxh).  If access to the monitor system calls is
                  0121          ; required by the user program , this entry point should not be
                  0122          ; redefined.
                  0123          ;
                  0124          ; To redefine this restart, put the address of the user subroutine
                  0125          ; at (UserJsys) [BFDFh].
                  0126  ;#########################################################################
                  0127
0008  E5          0128          Push    HL              ;Save HL
0009  2ADFBF      0129          Ld      HL,(UserJsys)   ;HL --> User Jsys subroutine
000C  E3          0130          Ex      (SP),HL         ;Put routine address on stack & restore HL
000D  C9          0131          Ret                     ;END Rst.10
                  0132
000E  FF          0133          defb    Offh            ;**** Place to put the checksum byte *****
                  0134
      (FFFF)      0135     if not test
000F              0136          Origin  10h
      (0001)      0137+ Swath:  defv    [Origin + 10h - $]
      (FFFF)      0138+   If Swath > 0
000F  (0001)      0139+         Rept    Swath
                  0140+         Defb    0FFh            ;Fill in empty spaces
                  0141+         Mend
000F  FF          0142+         Defb    0FFh            ;Fill in empty spaces
                  0143+   EndIf
      (0000)      0144+   If Swath < 0
                  0145+         MNote Program Origin Address Overlap @ 10h
                  0146+   EndIf
```

```
CROMEMCO Z80 Macro Assembler version 03.08                    Feb 21. 1984  15:28:53                    Page 0005
*** THE CROMEMCO OCTART MONITOR ***
*** STARTUP & RESTART CONTROL ***

                        endif
                    ;##########################################################################
                    Rst.10: ; This is a user-definable restart entry point.  To define this
                            ; restart put the address of the user subroutine at (User1) [BFE1h]
                    ;##########################################################################

0010  E5                    Push    HL              ;Save HL
0011  2AE1BF                Ld      HL,(User1)      ;HL --> User subroutine
0014  E3                    Ex      (SP),HL         ;Put routine address on stack & restore HL
0015  C9                    Ret                     ;END Rst.10

      (FFFF)          if not test
0016                        Origin  18h
      (0002)        Swath:  defv    [Origin + 18h - $]
      (FFFF)          If Swath > 0
0016  (0002)                Rept    Swath
                            Defb    0FFh            ;Fill in empty spaces
                            Mend
0016  FF                    Defb    0FFh            ;Fill in empty spaces
0017  FF                    Defb    0FFh            ;Fill in empty spaces
                        EndIf
      (0000)          If Swath < 0
                            MNote Program Origin Address Overlap @ 18h
                        EndIf
                        endif
                    ;##########################################################################
                    Rst.18: ; This is a user-definable restart entry point.  To define this
                            ; restart put the address of the user subroutine at (User2) [BFE3h]
                    ;##########################################################################

0018  E5                    Push    HL              ;Save HL
0019  2AE3BF                Ld      HL,(User2)      ;HL --> User subroutine
001C  E3                    Ex      (SP),HL         ;Put routine address on stack & restore HL
001D  C9                    Ret                     ;END Rst.18
```

```
                    0182
      (FFFF)        0183      if not test
001E                0184           Origin  20h
      (0002)        0185+ Swath:   defv    [Origin + 20h - $]
      (FFFF)        0186+    If Swath > 0
001E  (0002)        0187+           Rept    Swath
                    0188+           Defb    0FFh            ;Fill in empty spaces
                    0189+           Mend
001E  FF            0190+           Defb    0FFh            ;Fill in empty spaces
001F  FF            0191+           Defb    0FFh            ;Fill in empty spaces
                    0192+    EndIf
      (0000)        0193+    If Swath < 0
                    0194+           MNote Program Origin Address Overlap @ 20h
                    0195+    EndIf
                    0196      endif
                    0197  ;#######################################################################
                    0198  Rst.20: ; This is a user-definable restart entry point.  To define this
                    0199          ; restart put the address of the user subroutine at (User3) [BFE5h]
                    0200  ;#######################################################################
                    0201
0020  E5            0202           Push    HL              ;Save HL
0021  2AE5BF        0203           Ld      HL,(User3)      ;HL --> User subroutine
0024  E3            0204           Ex      (SP),HL         ;Put routine address on stack & restore HL
0025  C9            0205           Ret                     ;END Rst.20
                    0206
      (FFFF)        0207      if not test
0026                0208           Origin  28h
      (0002)        0209+ Swath:   defv    [Origin + 28h - $]
      (FFFF)        0210+    If Swath > 0
0026  (0002)        0211+           Rept    Swath
                    0212+           Defb    0FFh            ;Fill in empty spaces
                    0213+           Mend
0026  FF            0214+           Defb    0FFh            ;Fill in empty spaces
0027  FF            0215+           Defb    0FFh            ;Fill in empty spaces
                    0216+    EndIf
      (0000)        0217+    If Swath < 0
                    0218+           MNote Program Origin Address Overlap @ 28h
                    0219+    EndIf
                    0220      endif
                    0221  ;#######################################################################
                    0222  Rst.28: ; This is a user-definable restart entry point.  To define this
                    0223          ; restart put the address of the user subroutine at (User4) [BFE7h]
                    0224  ;#######################################################################
                    0225
0028  E5            0226           Push    HL              ;Save HL
0029  2AE7BF        0227           Ld      HL,(User4)      ;HL --> User subroutine
002C  E3            0228           Ex      (SP),HL         ;Put routine address on stack & restore HL
002D  C9            0229           Ret                     ;END Rst.28
                    0230
      (FFFF)        0231      if not test
002E                0232           Origin  30h
      (0002)        0233+ Swath:   defv    [Origin + 30h - $]
      (FFFF)        0234+    If Swath > 0
002E  (0002)        0235+           Rept    Swath
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21. 1984  15:28:53          Page 0007
*** THE CROMEMCO OCTART MONITOR ***
*** STARTUP & RESTART CONTROL ***

                    0236+          Defb    0FFh            ;Fill in empty spaces
                    0237+          Mend
002E  FF            0238+          Defb    0FFh            ;Fill in empty spaces
002F  FF            0239+          Defb    0FFh            ;Fill in empty spaces
                    0240+    EndIf
      (0000)        0241+    If Swath < 0
                    0242+          MNote Program Origin Address Overlap @ 30h
                    0243+    EndIf
                    0244      endif
                    0245  ;#########################################################################
                    0246  Rst.30: ; This is a user-definable restart entry point.  To define this
                    0247          ; restart put the address of the user subroutine at Break+1
                    0248  ;#########################################################################
                    0249
0030  C3FFBE        0250          Jp      Break           ;Goto debugger break point
                    0251
0033  6D02          0252          defw    SM.20           ;Pointer to sm command (for OCTARTDEBUG)
0035  5201          0253          defw    Init.20         ;Pointer to init single console
                    0254
      (FFFF)        0255      if not test
0037                0256          Origin  38h
      (0001)        0257+ Swath:  defv    [Origin + 38h - $]
      (FFFF)        0258+    If Swath > 0
0037  (0001)        0259+          Rept    Swath
                    0260+          Defb    0FFh            ;Fill in empty spaces
                    0261+          Mend
0037  FF            0262+          Defb    0FFh            ;Fill in empty spaces
                    0263+    EndIf
      (0000)        0264+    If Swath < 0
                    0265+          MNote Program Origin Address Overlap @ 38h
                    0266+    EndIf
                    0267      endif
                    0268  ;#########################################################################
                    0269  Crash:  ; This is the invalid-jump entry point (RST 38h instruction).
                    0270          ; The following subroutine will display the Crash error message with
                    0271          ; the crash address (top of stack) and restart the monitor.
                    0272  ;#########################################################################
                    0273
0038  E1            0274          Pop     HL              ;Get crash program counter
0039  31DFBF        0275          Ld      SP,Stack        ;Reload stack pointer
003C  CF09          0276          Jsys    .WrMsg          ;Print error message following
003E  0D437261      0277          defm    CR,'Crash '     ;Crash error message
0045  2B            0278          Dec     HL              ;Adjust the program counter
0046  CF10          0279          Jsys    .Prt2Hex        ;Print crash address
                    0280
```

```
                    0282  ;#############################################################################
                    0283  StartUp:; This section clears the monitor variables to zero, prints the signon
                    0284          ; message and initializes and establishes communication with one of
                    0285          ; the console channels available to the OCTART.
                    0286          ; In addition, a RAM test and PROM checksum test are performed.
                    0287  ;#############################################################################
                    0288
0048  F3            0289          di                      ; No interrupts, for a smooth start
                    0290
                    0291          ;***********************************************
                    0292          ;Check OCTART RAM For Errors
                    0293          ;***********************************************
                    0294
      (0000)        0295     If test
                    0296  ;       The OCTART RAM test is not used in RAM based assemblies
                    0297     Else
0049  210040        0298          Ld      HL,Default      ;Get start of RAM
004C  56            0299  SU.20:  Ld      D,(HL)          ;D = Data byte from (HL)
004D  7A            0300          Ld      A,D             ;A = Data byte
004E  2F            0301          Cpl                     ;Toggle all bits
004F  77            0302          Ld      (HL),A          ;Change bits in (HL)
0050  7E            0303          Ld      A,(HL)          ;A = Inverted data byte
0051  72            0304          Ld      (HL),D          ;Restore original byte
0052  2F            0305          Cpl                     ;Toggle all bits
0053  92            0306          Sub     A,D             ;Do they match ?
0054  2020          0307          Jr      NZ,MonErr1      ;No: skip next
0056  23            0308          Inc     HL              ;Bump RAM pointer
0057  B4            0309          Or      A,H             ;A = High byte of address
0058  F24C00        0310          Jp      P,SU.20         ;Loop till H = 80h
                    0311     Endif
005B  97            0312          sub     a
005C  32EBBF        0313          ld      (errbit),a      ;clear error byte
005F  C37B00    R   0314          Jp      su.25
                    0315
      (FFFF)        0316     if not test
0062                0317          Origin  66h
      (0007)        0318+ Swath:  defv    [Origin + 66h - $]
      (FFFF)        0319+   If Swath > 0
0062  (0004)        0320+         Rept    Swath
                    0321+         Defb    0FFh            ;Fill in empty spaces
                    0322+         Mend
0062  FF            0323+         Defb    0FFh            ;Fill in empty spaces
0063  FF            0324+         Defb    0FFh            ;Fill in empty spaces
0064  FF            0325+         Defb    0FFh            ;Fill in empty spaces
0065  FF            0326+         Defb    0FFh            ;Fill in empty spaces
                    0327+   EndIf
      (0000)        0328+   If Swath < 0
                    0329+         MNote Program Origin Address Overlap @ 66h
                    0330+   EndIf
                    0331     endif
                    0332  ;#############################################################################
                    0333  ;
                    0334  ;       N O N - M A S K A B L E   I N T E R R U P T   V E C T O R
                    0335  ;
```

```
                    0336  ; This interrupt vector is user definable by changing the pointer located at
                    0337  ; (UserNMI) [BFE9h]
                    0338  ;##############################################################################
                    0339
0066  E5             0340          Push    HL              ;Save HL
0067  2AE9BF         0341          Ld      HL,(UserNMI)    ;Get user Non Maskable Interrupt vector
006A  E3             0342          Ex      (Sp),HL         ;Put vector on stack & restore HL
006B  C9             0343          Ret
                    0344
```

```
                    0346  ;#############################################################################
                    0347          ;This routine is jumped to if an error is detected in the OCTART RAM
                    0348          ;test or the PROM checksum test.
                    0349  ;#############################################################################
                    0350
006C  21EBBF        0351  MonErr2:Ld      hl,errbit       ; Bit 4 (10h) = ROM memory checksum error
006F  CBE6          0352          set     4,(hl)
0071  210008        0353          ld      hl,800h
0074  1863          0354          jr      su.60
                    0355
0076  3E08          0356  MonErr1:Ld      A,Ramerr        ; Bit 3 (08h) = RAM memory test error
0078  32EBBF        0357          ld      (errbit),a      ;
                    0358
                    0359          ;***********************************************
                    0360          ; Miscellaneous initialization
                    0361          ;***********************************************
                    0362
007B  31DFBF        0363  su.25:  Ld      SP,Stack        ;Set program stack pointer
007E  217904        0364          Ld      HL,Jsys         ;HL --> Jsys entry point
0081  22DFBF        0365          Ld      (UserJsys),HL   ;Setup for use by monitor
                    0366
                    0367  Arnd66: ;***********************************************
                    0368          ; Startup continued
                    0369          ;***********************************************
                    0370
0084  CDBA01        0371          Call    inisc           ;init SC2681
0087  21FFBE        0372          Ld      HL,VarTbl       ;HL --> OCTARTMonitor variable table
008A  01DC00        0373          Ld      BC,Stack-VarTbl-4;BC = Number of bytes to fill
008D  CD5B03        0374          Call    Zap.05          ;Fill variable table with zeros
0090  210040        0375          Ld      HL.Default      ;HL = Initial default value for pointers
0093  2205BF        0376          Ld      (SmPtr),HL      ;For substitute memory command
0096  2203BF        0377          Ld      (DmPtr),HL      ;Set display memory default pointer
0099  3EC3          0378          Ld      A,0C3h          ;Set up JP instruction
009B  32FFBE        0379          Ld      (Break),A       ;At break to abort
009E  213800        0380          Ld      HL,Crash        ;HL --> Crash routine
00A1  2200BF        0381          Ld      (Break+1),HL    ;Set initial address of OCTARTDEBUG break point
00A4  22E1BF        0382          Ld      (User1),HL      ;Set initial address of user restart #1
00A7  22E3BF        0383          Ld      (User2),HL      ;Set initial address of user restart #2
00AA  22E5BF        0384          Ld      (User3),HL      ;Set initial address of user restart #3
00AD  22E7BF        0385          Ld      (User4),HL      ;Set initial address of user restart #4
00B0  22E9BF        0386          Ld      (UserNMI),HL    ;Set initial NMI interrupt vector
                    0387
                    0388          ;***********************************************
                    0389          ; Check Monitor PROM For Errors (XOR checksum)
                    0390          ;***********************************************
                    0391
00B3  210000        0392          Ld      HL.Origin       ;HL --> Beginning of monitor PROM
00B6  010008        0393          Ld      BC,0800h        ;BC = Number of bytes to test
00B9  AF            0394          xor     a,a             ;Clear accumulator
00BA  AE            0395  SU.30:  Xor     A,(HL)          ;Test PROM byte
00BB  EDA1          0396          CPI                     ;Adjust counters
00BD  EABA00        0397          Jp      PE,SU.30        ;Loop till BC = 0
      (0000)        0398    If Test
                    0399          Xor     A,A             ;Clear accumulator
```

```
CROMEMCO Z80 Macro Assembler version 03.08                    Feb 21, 1984  15:28:53                    Page 0011
*** THE CROMEMCO OCTART MONITOR ***
*** MORE INITS & ROM TEST ***

                    0400    Else
00C0  B7            0401            Or      A,A             ;Test accumulator for all bits zero
                    0402    EndIf
                    0403
00C1  20A9          0404            Jr      NZ,MonErr2      ;Skip next if error
00C3  1814          0405            jr      su.60
                    0406
```

```
                0408          ;***********************************************
                0409          ; Check If User PROM In System (HL = 800h)
                0410          ;***********************************************
                0411
      (0000)    0412    If test
                0413          Ld      HL,800h         ;HL --> First USER PROM socket
                0414    Endif
00C5  7E        0415  SU.40:  Ld      A,(HL)          ;A = First byte of PROM
00C6  3C        0416          Inc     A               ;Is there a PROM there ?
00C7  7C        0417          Ld      A,H             ;A = High address byte of user PROM
00C8  2808      0418          Jr      Z,SU.50         ;No: skip next
                0419
                0420          ;***********************************************
                0421          ; Branch To Routines In User PROM
                0422          ; All OCTART initialization has been completed for
                0423          ; use by the user routines.
                0424          ;***********************************************
                0425
00CA  21D900    0426          Ld      HL,SU.60        ;HL --> Re-entry address
00CD  E5        0427          Push    HL              ;Put re-entry address on stack
00CE  67        0428          Ld      H,A             ;Put address in HL
00CF  2E00      0429          Ld      L,0             ;Force to even boundary
00D1  E9        0430          Jp      (HL)            ;Execute User routines in PROM
                0431
00D2  C608      0432  su.50:  Add     A,8             ;A = High address byte of next PROM to test
00D4  67        0433          Ld      H,A             ;Put high address in H
00D5  FE40      0434          Cp      A,40h           ;Are we finished ?
00D7  20EC      0435          Jr      NZ,SU.40        ;No: Loop
                0436
                0437          ;***********************************************
                0438          ; Select the console & print signon prompt
                0439          ;***********************************************
                0440
00D9  CD5201    0441  SU.60:  Call    Init.20         ;Select 1 of N consoles
00DC  CD1A02    0442          call    ckerror         ;check RAM ROM error
00DF  CF09      0443          Jsys    .WrMsg          ;Print following message on current console
                0444
00E1  4F435441  0445          defb    'OCTART Monitor '
00F0  30312E    0446          defb    Version/10+'0',Version%10+'0','.'
00F3  30338D    0447          defm    Release/10+'0',Release%10+'0',CR
                0448
```

```
                    0450  ;#############################################################################
                    0451  ReEntry:; Monitor main command loop entry point
                    0452  Exit:   ; Jsys .Exit command entry point
                    0453          ;
                    0454          ; If Batch is in progress, then continue with next batch command,
                    0455          ; else print monitor prompt and input command from console.
                    0456  ;#############################################################################
                    0457
00F6  31DFBF        0458          Ld      SP,Stack        ;Reload stack pointer
00F9  ED5B0FBF      0459          Ld      DE,(BatchPtr)   ;DE --> Next command in batch
00FD  13            0460          Inc     DE              ;
00FE  3A0DBF        0461          Ld      A,(Batch.F)     ;A = Batch mode flag
0101  B7            0462          Or      A,A             ;Is there an active batch ? (accumulator <> 0)
0102  200A          0463          Jr      NZ,RE.10        ;Yes: Continue without prompting
0104  CF09          0464          Jsys    .WrMsg          ;Print message following
      (0000)        0465    If test
                    0466          defm    '>'             ;The RAM-based prompt
                    0467    Else
0106  AE            0468          defm    '.'             ;The normal ROM-based prompt
                    0469    EndIf
0107  1112BF        0470          Ld      DE,Input$       ;DE --> Input buffer
010A  3E48          0471          Ld      A,Length        ;A = Maximum input line length
010C  CF04          0472          Jsys    .RdLine         ;Input line from console
010E  CDA304        0473  RE.10:  Call    Scan.10         ;Find first non-space
0111  2827          0474          Jr      Z,ReEntx        ;END of line, reset batch flags and return
0113  47            0475          Ld      B,A             ;B = Command letter
0114  219707        0476          Ld      HL,CmdTbl-2     ;Point to command jump table
0117  23            0477  RE.20:  Inc     HL              ;Bump pointer
0118  23            0478          Inc     HL              ;HL --> Command letter from table
0119  7E            0479          Ld      A,(HL)          ;A = Command letter from table
011A  23            0480          Inc     HL              ;HL --> Command address
011B  B7            0481          Or      A,A             ;Is this the end of the command table ?
011C  2818          0482          Jr      Z,CmdErr        ;Yes: Print error message and Return
011E  B8            0483          Cp      A,B             ;Is this the command ?
011F  20F6          0484          Jr      NZ,RE.20        ;NO: Loop till valid command or end of table
0121  CDFF04        0485          Call    Loadhh          ;Yes: Get the command routine address
0124  13            0486          Inc     DE              ;Bump past command letter
0125  CDA304        0487          Call    Scan.10         ;Get next character
0128  FE4D          0488          Cp      A,'M'           ;Is next character an 'M' ?
012A  2001          0489          Jr      NZ,RE.30        ;No: skip next
012C  13            0490          Inc     DE              ;Yes: Bump character pointer
012D  01F600        0491  RE.30:  Ld      BC,ReEntry      ;BC --> Re-entry location
0130  C5            0492          Push    BC              ;Force return address on stack
0131  E5            0493          Push    HL              ;Put execution routine address on stack
0132  210040        0494          Ld      HL,Default      ;Load default arg1 if needed
0135  C9            0495          Ret                     ;Go execute routine in HL
```

```
                    0497
                    0498  ;#########################################################################
                    0499  CmdErr: ; Come here if we get a command error
                    0500  ;#########################################################################
                    0501
0136  CF09          0502          Jsys    .WrMsg          ;Print error message following
0138  07BF          0503          defm    BEL,'?'
                    0504
                    0505  ;#########################################################################
                    0506  ReEntx: ; This routine will return to the address contained in (Return) only
                    0507          ; if the break jump transfer is not pointing to the monitor, else
                    0508          ; go back to main command level (no batch jobs will survive).
                    0509          ; If the console is not at the beginning of an output line then
                    0510          ; print a <cr><lf> sequence.
                    0511  ;#########################################################################
                    0512
013A  CD3A03        0513          Call    BM.Off          ;Turn off batch mode
013D  3A08BF        0514          Ld      A,(Column)      ;A = Console column number
0140  B7            0515          Or      A,A             ;Is it Zero ?
0141  C4D006        0516          Call    NZ,CRLF         ;No: print a CR,LF
0144  3A01BF        0517          Ld      A,(Break+2)     ;Get break's jump address
      (FFFF)        0518    If Not Test
0147  B7            0519          Or      A,A             ;Is OCTARTDEBUG running?
                    0520    Else
                    0521          Cp      A,40h           ;Is OCTARTDEBUG running ?
                    0522    EndIf
0148  28AC          0523          Jr      Z,ReEntry       ;No: Go back to command level
014A  2A0BBF        0524          Ld      HL,(Return)     ;Yes: Get execution return address
014D  E5            0525          Push    HL              ;Put address on stack
014E  C9            0526          Ret                     ;Return to calling routine
                    0527
```

```
                        0529  ;#############################################################################
                        0530  Init:   ; Command to initialize console I<cr>
                        0531          ; 1. This command is used to transfer console control to a different
                        0532          ;    terminal.
                        0533          ; 2. Main console initialization routines: scan system for active
                        0534          ;    console port, determine the baud rate, and define the current
                        0535          ;    console number (ConID).
                        0536          ; 3. If ConID is negative then the console is the Host system,
                        0537          ;    else the console is a Octart channel.
                        0538  ;#############################################################################
                        0539
014F  CD0F05            0540          Call    Eolchk          ;Are we at the end of the line ?
                        0541
                        0542          ;***********************************************
                        0543          ; Check if HOST channel is active
                        0544          ;***********************************************
                        0545
0152  AF                0546  Init.20:Xor     A,A             ;Clear accumulator
0153  10FE              0547  Init.25:Djnz    $               ;Wait for any characters to clear SIO output
0155  3D                0548          Dec     A               ;Decrement loop counter
0156  20FB              0549          Jr      NZ,Init.25      ;Loop till accumulator = 0
0158  DB02              0550          in      a,(HPORT)       ;get status
015A  1F                0551          rra
015B  3AEBBF            0552          ld      a,(errbit)      ;if error ,send it otherwise
015E  3802              0553          jr      c,init.27       ;it has command waiting
0160  CBCF              0554          set     1,a             ;set bit 0 (no command waiting)
                        0555  init.27:
0162  D300              0556          Out     HPort-2,A       ;Say we are working on the host
0164  3EFF              0557          ld      a,0ffh
0166  CF11              0558          Jsys    .Select         ;Select Host & check if character ready
0168  2805              0559          Jr      Z,Init.30       ;No character ready, check OCTART serial port
016A  CDAF01            0560          Call    CRTest          ;Check if carriage returns entered
016D  283D              0561          Jr      Z,Init.70       ;Yes: Echo it & return
                        0562
                        0563          ;***********************************************************
                        0564          ; Check to see if anyone of eight channels connect to OCTART
                        0565          ;***********************************************************
                        0566  Init.30:
016F  3E51              0567          ld      a,STAT_1A       ;chip #1 #A (chip 1 ,channel A)
0171  CD9901            0568          call    Init.50         ;q,connected & cr typed?
0174  3E59              0569          ld      a,STAT_1B       ;chip #1 #B (chip 1 ,channel B)
0176  CD9901            0570          call    Init.50         ;q,connected & cr typed?
                        0571   ;
0179  3E11              0572          ld      a,STAT_2A       ;chip #2 #A (chip 2 ,channel A)
017B  CD9901            0573          call    Init.50         ;q,connected & cr typed?
017E  3E19              0574          ld      a,STAT_2B       ;chip #2 #B (chip 2 ,channel B)
0180  CD9901            0575          call    Init.50         ;q,connected & cr typed?
                        0576  ;
0183  3E21              0577          ld      a,STAT_3A       ;chip #3 #A (chip 3 ,channel A)
0185  CD9901            0578          call    Init.50         ;q,connected & cr typed?
0188  3E29              0579          ld      a,STAT_3B       ;chip #3 #B (chip 3 ,channel B)
018A  CD9901            0580          call    Init.50         ;q,connected & cr typed?
                        0581  ;
018D  3E31              0582          ld      a,STAT_4A       ;chip #4 #A (chip 4 ,channel A)
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0016
*** THE CROMEMCO OCTART MONITOR ***
*** INITIALIZATION SUBROUTINES ***

018F  CD9901      0583          call    Init.50         ;q,connected & cr typed?
0192  3E39        0584          ld      a,STAT_4B       ;chip #4 #B (chip 4 ,channel B)
0194  CD9901      0585          call    Init.50         ;q,connected & cr typed?
                  0586
0197  18B9        0587          jr      Init.20         ;loop to scan all ports
                  0588
                  0589
                  0590  Init.50:
0199  CF11        0591          Jsys    .Select
019B  C8          0592          ret     z               ;character not ready
019C  CDAF01      0593          Call    CRTest          ;Check if carriage returns entered
019F  C0          0594          ret     nz              ;Not Carriage return
01A0  F1          0595          pop     af              ;stack adjustment
01A1  DB02        0596          in      a,(HPORT)       ;get status
01A3  1F          0597          rra
01A4  3E20        0598          Ld      A,QMode         ;Yes: Tell the host we have initialized
01A6  3802        0599          jr      c,init.55
01A8  CBCF        0600          set     1,a
                  0601  init.55:
                  0602
01AA  D300        0603          Out     Hport-2,A       ;one of the channels
01AC  C3D006      0604  Init.70:Jp      CRLF            ;Send a newline and return
                  0605
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0017
*** THE CROMEMCO OCTART MONITOR ***
*** INITIALIZATION SUBROUTINES ***

                    0607
                    0608            ;***********************************************
                    0609    CRTest: ; Test for carriage-returns from the console
                    0610            ;***********************************************
                    0611
01AF  CD8306        0612            call    Gettim          ;Get character from console
01B2  2803          0613            jr      z,crtto
01B4  CD8306        0614            call    Gettim          ;Get second character from console
01B7  FE0D          0615    crtto:  Cp      A,CR            ;Is it a carriage return ?
01B9  C9            0616            Ret
                    0617
```

```
                    0619  ;#############################################################################
                    0620  ;;OCTARTInit:; OCTART Initialization & Setup Routine
                    0621          ;
                    0622          ; This routine sets up the default baud rates and masks out all
                    0623          ; interrupts for all active devices connected to the OCTART.
                    0624  ;#############################################################################
                    0625  ;
                    0626  ; SC2681 init routine
                    0627  ;
                    0628  inisc:
01BA  211002        0629          ld      hl,ilist        ;pointer to init data
01BD  0E50          0630          ld      c,MR1A          ;mode register 1 channel A
01BF  CDF001        0631          call    outport         ;init register 1 (1.2.3.4 chips) &
                    0632                                  ; (cnannel A,B)
01C2  0E50          0633          ld      c,MR2A          ;Mode register 2
01C4  CDF001        0634          call    outport         ;
                    0635  ;
01C7  0E51          0636          ld      c,CSRA          ;Clock select register (9600 baud)
01C9  CDF001        0637          call    outport         ;
                    0638  ;
01CC  0E52          0639          ld      c,CRA           ;Command register
01CE  CDF001        0640          call    outport         ;
                    0641  ;
01D1  0E53          0642          ld      c,THRA          ;Tx holding register
01D3  CDF001        0643          call    outport         ;
                    0644  ;
01D6  0E54          0645          ld      c,ACR           ;Aux. control register
01D8  CDF001        0646          call    outport         ;
                    0647  ;
01DB  0E55          0648          ld      c,IMR           ;Interrupt mask register
01DD  CDF001        0649          call    outport         ;
                    0650  ;
01E0  0E56          0651          ld      c,CTUR          ;C/T upper register
01E2  CDF001        0652          call    outport         ;
                    0653  ;
01E5  0E57          0654          ld      c,CTLR          ;C/T lower register
01E7  CDF001        0655          call    outport         ;
                    0656  ;
01EA  0E52          0657          ld      c,CRA           ;Enable Tx.,Rx.
01EC  CDF001        0658          call    outport         ;
01EF  C9            0659          ret
                    0660  ;
                    0661  outport:
01F0  7E            0662          ld      a,(hl)          ;get init data from the data list
01F1  CD0102        0663          call    out20           ;init cnannel A (chip #1,#2,#3,#4)
                    0664  ;                               ;   /
01F4  CBA9          0665          res     5,c             ;  /
01F6  CBE1          0666          set     4,c             ; /
01F8  CBF1          0667          set     6,c             ;port = 5X
01FA  CBD9          0668          set     3,c             ;set bit 3 for B channel
01FC  CD0102        0669          call    out20           ;init channel B (chip #1,#2,#3,#4)
01FF  23            0670          inc     hl
0200  C9            0671          ret
                    0672  ;
```

```
                    0673  ;
                    0674  out20:
0201  ED79           0675          out     (c),a           ;chip #1 (5X)
0203  CBB1           0676          res     6,c             ;
0205  ED79           0677          out     (c),a           ;chip #2 (1X)
0207  CBE9           0678          set     5,c             ;
0209  ED79           0679          out     (c),a           ;chip #4 (3X)
020B  CBA1           0680          res     4,c             ;
020D  ED79           0681          out     (c),a           ;chip #3 (2X)
020F  C9             0682          ret
                     0683
                     0684  ;
                     0685
                     0686  ilist:
0210  13             0687          db      00010011b               ;MR1A,MR1B(no parity,8 bits/char)
0211  07             0688          db      00000111b               ;MR2A,MR2B(stop bit length 1)
0212  BB             0689          db      10111011b               ;CSRA,CSRB (9600 BAUD)
0213  0A             0690          db      00001010b               ;CRA,CRB
0214  0D             0691          db      00001101b               ;THRA,THRB (CARRIAGE RETURN)
0215  80             0692          db      10000000b               ;ACR
0216  00             0693          db      00000000b               ;IMR
0217  00             0694          db      00000000b               ;CTUR
0218  00             0695          db      00000000b               ;CTLR
0219  05             0696          db      00000101b               ;ACR (Enable Tx,Rx)
                     0697
                     0698  ;#########################################################################
                     0699  ; Check ROM & RAM error bit, if error then display error message.
                     0700  ;
                     0701  ;#########################################################################
                     0702
                     0703  ckerror:
021A  21EBBF         0704          ld      hl,errbit               ;pointer to error byte
021D  CB5E           0705          bit     3,(hl)                  ;q,RAM error(if bit set)
021F  280D           0706          jr      z,err10                 ;
0221  CF09           0707          Jsys    .WrMsg                  ;display error message
0223  52414D20       0708          defm    'RAM error',cr,lf       ;
022E  21EBBF         0709  err10:  ld      hl,errbit               ;pointer to error byte
0231  CB66           0710          bit     4,(hl)                  ;q,ROM error(if bit set)
0233  C8             0711          ret     z                       ;
0234  CF09           0712          Jsys    .WrMsg                  ;display error message
0236  524F4D20       0713          defm    'ROM error',cr,lf       ;
0241  C9             0714          ret
                     0715
                     0716
                     0717
                     0718
```

```
                    0720  ;#############################################################################
                    0721  Move:   ; Move memory command
                    0722          ; M source s swath dest <cr>
                    0723          ; M source finish dest <cr>
                    0724          ; This command moves swath bytes from source to destination
                    0725          ; (or from source through finish to destination)
                    0726  ;#############################################################################
                    0727
0242  D5            0728          Push    DE              ;Save command line pointer
0243  CD2005        0729          Call    Arg3q           ;Get arguments
0246  CD6203        0730          Call    Zap.20          ;Move BC characters from (HL) to (DE)
0249  D1            0731          Pop     DE              ;Restore command line pointer
                    0732
                    0733  ;#############################################################################
                    0734  Verify: ; Verify memory command
                    0735          ; Forms of this command:
                    0736          ; V source s swath dest <cr>
                    0737          ; V source finish dest <cr>
                    0738          ; This command compares (verifies) swath bytes from
                    0739          ; source to destination (or from source through finish
                    0740          ; to destination) and displays any discrepancies
                    0741  ;#############################################################################
                    0742
024A  CD2005        0743          Call    Arg3q           ;Get the arguments
024D  1A            0744  VM.10:  Ld      A,(DE)          ;A = Data from destination
024E  BE            0745          Cp      A,(HL)          ;Is it the same as source?
024F  2810          0746          Jr      Z,VM.20         ;Yes: Check next byte
0251  CF10          0747          Jsys    .Prt2Hex        ;No: Print source address
0253  7E            0748          Ld      A,(HL)          ;A = Source data
0254  CDFA06        0749          Call    HexBS           ;Print it
0257  1A            0750          Ld      A,(DE)          ;Get destination data
0258  CDFA06        0751          Call    HexBS           ;Print it
025B  EB            0752          Ex      DE,HL           ;HL = Destination address
025C  CF10          0753          Jsys    .Prt2Hex        ;Print it
025E  EB            0754          Ex      DE,HL           ;Restore it
025F  CF0A          0755          Jsys    .CRLF           ;New line
0261  13            0756  VM.20:  Inc     DE              ;Bump destination pointer
0262  EDA1          0757          CPI                     ;Bump HL & BC & set parity flag
0264  E0            0758          Ret     PO              ;EXIT if BC = 0
0265  18E6          0759          Jr      VM.10           ;No, keep checking
                    0760
```

```
                        0762  ;#############################################################################
                        0763  SetMem: ; Set (substitute) memory command
                        0764          ; Form of this command: SM addr <cr>
                        0765          ; This command will prompt for information to be stuffed into memory
                        0766          ; (this command kills batch jobs due to possible buffer conflicts).
                        0767          ; The letter M is optional
                        0768  ;#############################################################################
                        0769
0267  2A05BF            0770          Ld      HL,(SmPtr)      ;Get default argument
026A  CD0D05            0771          Call    Arg1d           ;Get starting address
026D  CD3A03            0772  SM.20:  Call    BM.Off          ;Turn off batch mode
0270  CF10              0773          Jsys    .Prt2Hex        ;Print set address
0272  7E                0774          Ld      A,(HL)          ;Get current contents
0273  CDFA06            0775          Call    HexBS           ;Print them
0276  1112BF            0776          Ld      DE,Input$       ;Point to line buffer
0279  3E48              0777          Ld      A,Length        ;Get line length
027B  CF04              0778          Jsys    .RdLine         ;Read input line from console
027D  CDA304            0779          Call    Scan.10         ;Scan for non-space
0280  2812              0780          Jr      Z,SM.40         ;Skip this byte if CR only entered
0282  FE2E              0781          Cp      A,'.'           ;Is this the end ?
0284  C8                0782          Ret     Z               ;Yes, return
0285  FE2D              0783          Cp      A,'-'           ;Is it to back up?
0287  2B                0784          Dec     HL              ;Back up memory pointer
0288  281B              0785          Jr      Z,SM.70         ;YES: Save it & try again
028A  23                0786          Inc     HL              ;Restore pointer
028B  CDC904            0787          Call    Instr           ;Convert input line to binary
028E  3807              0788          Jr      C,SM.50         ;Skip next if error
0290  AF                0789          Xor     A,A             ;Clear accumulator
0291  B0                0790          Or      A,B             ;Is count = 0 ?
0292  200A              0791          Jr      NZ,SM.60        ;No: Save the data
0294  23                0792  SM.40:  Inc     HL              ;Bump pointer
0295  180E              0793          Jr      SM.70           ;& Try again
                        0794
0297  CF09              0795  SM.50:  Jsys    .WrMsg          ;Print error message
0299  073F8D            0796          defm    BEL,'?\r'       ;
029C  18CF              0797          Jr      SM.20           ;Loop
                        0798
029E  48                0799  SM.60:  Ld      C,B             ;Load count in BC
029F  0600              0800          Ld      B,0             ;/
02A1  EB                0801          Ex      DE,HL           ;Swap pointers
02A2  EDB0              0802          Ldir                    ;Shove the data into memory
02A4  EB                0803          Ex      DE,HL           ;Restore pointers
02A5  2205BF            0804  SM.70:  Ld      (SmPtr),HL      ;Save pointer for later
02A8  18C3              0805          Jr      SM.20           ;& go for more
                        0806
```

```
                    0808  ;#############################################################################
                    0809  Query:  ; Find a string-of-bytes command
                    0810          ; Forms of this command:
                    0811          ; Q start s swath string-of-bytes <cr>
                    0812          ; Q start finish string-of-bytes <cr>
                    0813          ; This command searches the memory specified from
                    0814          ; start for swath bytes (or from start through
                    0815          ; finish) for the specified string-of-bytes;
                    0816          ; when found, the first 16 bytes are displayed
                    0817  ;#############################################################################
                    0818
02AA  CD2E05        0819          Call    Arg2s           ;Get addresses and string for query
02AD  50            0820          Ld      D,B             ;Put swath in DE
02AE  59            0821          Ld      E,C             ;
02AF  B7            0822          Or      A,A             ;A = Length of input string
02B0  CA3601        0823          Jp      Z,Cmderr        ;ERROR: No input string
02B3  47            0824          Ld      B,A             ;Put length in B
                    0825
02B4  C5            0826  QM.20:   Push    BC              ;Save length
02B5  D5            0827          Push    DE              ;Save swath
02B6  E5            0828          Push    HL              ;Save start address
02B7  1112BF        0829          Ld      DE,Input$       ;Point to binary buffer
02BA  1A            0830  QM.30:   Ld      A,(DE)          ;Get a byte
02BB  BE            0831          Cp      A,(HL)          ;Is it the same?
02BC  2004          0832          Jr      NZ,QM.40        ;No, failure here
02BE  13            0833          Inc     DE              ;Bump binary pointer
02BF  23            0834          Inc     HL              ;Bump memory pointer
02C0  10F8          0835          DJNZ    QM.30           ;Go until all expended
                    0836
                    0837          ;***********************************************
                    0838          ; String match located, display line
                    0839          ;***********************************************
                    0840
02C2  E1            0841  QM.40:   Pop     HL              ;Get memory pointer back
02C3  E5            0842          Push    HL              ;momentarily.
02C4  0610          0843          Ld      B,10h           ;Print 10 bytes
02C6  CCFE02        0844          Call    Z,DspLine       ;only if the strings matched
02C9  E1            0845          Pop     HL              ;Get memory pointer back again
02CA  D1            0846          Pop     DE              ;and swath
02CB  C1            0847          Pop     BC              ;and count
02CC  23            0848          Inc     HL              ;Bump memory pointer
02CD  1B            0849          Dec     DE              ;Drop swath
02CE  7A            0850          Ld      A,D             ;Did the swath
02CF  B3            0851          Or      A,E             ;go to 0?
02D0  20E2          0852          Jr      NZ,QM.20        ;No, keep looking
02D2  C9            0853          Ret                     ;Yes, all done here
                    0854
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21. 1984  15:28:53          Page 0023
*** THE CROMEMCO OCTART MONITOR ***
*** DISPLAY MEMORY COMMAND ***

                        0856  ;#######################################################################################
                        0857  Display:; Display memory command
                        0858          ; Forms of this command:
                        0859          ; DM start s swath <cr>
                        0860          ; DM start finish <cr>
                        0861          ; This command displays the contents of memory beginning
                        0862          ; at start for swath bytes (or from start through finish)
                        0863          ; If the start is missing, the last DM address is assumed
                        0864          ; If the swath is missing, 80h is assumed
                        0865          ; The letter M is optional
                        0866  ;#######################################################################################
                        0867
02D3  018000            0868  DM.10:  Ld      BC,80h          ;Default swath is 80h bytes
02D6  2A03BF            0869          Ld      HL,(DmPtr)      ;Get default data pointer
02D9  CD1C05            0870          Call    Arg2d           ;Get (new) arguments
02DC  1E10              0871  DM.20:  Ld      E,10h           ;Assume line length of 10h
02DE  AF                0872          Xor     A,A             ;Clear accumulator
02DF  B0                0873          Or      A,B             ;Check if near the end
02E0  2008              0874          Jr      NZ,DM.30        ;No, continue
02E2  79                0875          Ld      A,C             ;A = Number of bytes left
02E3  BB                0876          Cp      A,E             ;Are there less than 10h ?
02E4  3004              0877          Jr      NC,DM.30        ;No, still assume 10h
02E6  B7                0878          Or      A,A             ;Is swath 0?
02E7  2801              0879          Jr      Z,DM.30         ;Yes, assume 10h to dump all
02E9  59                0880          Ld      E,C             ;Get final amount
02EA  C5                0881  DM.30:  Push    BC              ;Save count to go
02EB  43                0882          Ld      B,E             ;Load line count
02EC  CDFE02            0883          Call    DspLine         ;Print the line
02EF  2203BF            0884          Ld      (DmPtr),HL      ;Save data address
02F2  C1                0885          Pop     BC              ;Get count back
02F3  79                0886          Ld      A,C             ;Adjust the count
02F4  93                0887          Sub     A,E             ;By subtracting
02F5  4F                0888          Ld      C,A             ;The number
02F6  3001              0889          Jr      NC,DM.40        ;Of bytes
02F8  05                0890          Dec     B               ;Just printed
02F9  78                0891  DM.40:  Ld      A,B             ;A = High byte of count
02FA  B1                0892          Or      A,C             ;Are we finished ?
02FB  20DF              0893          Jr      NZ,DM.20        ;No: Loop until BC = 0
02FD  C9                0894          Ret                     ;Yes, return
                        0895
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0024
*** THE CROMEMCO OCTART MONITOR ***
*** DISPLAY A LINE OF MEMORY UP TO 16 BYTES ***

                        0897  ;###############################################################################
                        0898  DspLine:; Display up to 16 bytes of memory on the console
                        0899          ; Entry: B contains the length
                        0900          ; HL points to the data
                        0901  ;###############################################################################
                        0902
02FE  CF10              0903          Jsys    .Prt2Hex        ;Print address
0300  C5                0904          Push    BC              ;Save count
0301  E5                0905          Push    HL              ;Save data pointer
0302  0E00              0906          Ld      C,0             ;0 Bytes printed
0304  3E03              0907  DL.10:  Ld      A,03h           ;Check if multiple
0306  A1                0908          And     A,C             ;of 4 bytes printed
0307  CCFD06            0909          Call    Z,Space         ;Yes, print a space
030A  7E                0910          Ld      A,(HL)          ;Get data byte
030B  CDF306            0911          Call    Hexbo           ;Print it
030E  23                0912          Inc     HL              ;Bump data pointer
030F  0C                0913          Inc     C               ;Bump byte count
0310  10F2              0914          DJNZ    DL.10           ;Go until all has been dumped
0312  063A              0915          Ld      B,58            ;Move out to column 58
0314  CDC406            0916          Call    MoveCsr         ;Go there
0317  E1                0917          Pop     HL              ;Get data pointer back
0318  C1                0918          Pop     BC              ;& Count, too
0319  7E                0919  DL.30:  Ld      A,(HL)          ;Get character
031A  23                0920          Inc     HL              ;Bump pointer
031B  E67F              0921          And     A,7Fh           ;Mask off bit 7 ("parity" bit)
031D  FE7F              0922          Cp      A,DEL           ;Is it <del>?
031F  2804              0923          Jr      Z,DL.40         ;Yes: Substitute '.'
0321  FE20              0924          Cp      A,' '           ;Is it printable ?
0323  3002              0925          Jr      NC,DL.50        ;Yes: Print it
0325  3E2E              0926  DL.40:  Ld      A,'.'           ;A = '.' instead of nonprintable character
0327  CF07              0927  DL.50:  Jsys    .ConOut         ;Print character
0329  10EE              0928          DJNZ    DL.30           ;& Go until this line done
032B  1818              0929          Jr      CRLF.1          ;Then goto next line & return
                        0930
```

```
CROMEMCO Z80 Macro Assembler version 03.08                    Feb 21. 1984  15:28:53                    Page 0025
*** THE CROMEMCO OCTART MONITOR ***
*** OTHER MISCELLANEOUS COMMANDS ***

                  ;##########################################################################
                  Batch.1:; Execute command string in memory at the specified address
                          ; Form of this command: @ Addr <cr>
                          ; This command starts up a batch job as
                          ; a sequence of commands to be executed.
                  ;##########################################################################

032D  CD0605              Call    Arg1q           ;Get the string address
0330  2B          Batch:  Dec     HL              ;
0331  220FBF              Ld      (BatchPtr),HL   ;Save it for batch processing
0334  3EFF        BM.On:  Ld      A,-1            ;Say that batch
0336  320DBF      BM.Set: Ld      (Batch.F),A     ;is active
0339  C9                  Ret                     ;END Batch

033A  AF          BM.Off: Xor     A,A             ;Clear accumulator
033B  18F9                Jr      BM.Set          ;Clear batch mode

                  ;##########################################################################
                  Examine:; Examine input port
                          ; Form of this command: E port <cr>
                          ; The contents of the input port are displayed on the console
                  ;##########################################################################

033D  CD0605              Call    Arg1q           ;Get argument where to look
0340  4D                  Ld      C,L             ;Load into c
0341  ED78                In      A,(C)           ;& Get the data
0343  CF0F                Jsys    .PrtHex         ;Print it
0345  C3D006      CRLF.1: Jp      CRLF            ;END Examine

                  ;##########################################################################
                  Output: ; Output data to a port
                          ; Form of this command: O data port <cr>
                          ; The data specified is sent to the port specified
                  ;##########################################################################

0348  CF0D                Jsys    .Arg            ;Get data byte
034A  E5                  Push    HL              ;Save it on the stack
034B  CD0605              Call    Arg1q           ;Get port #
034E  4D                  Ld      C,L             ;Load into c
034F  E1                  Pop     HL              ;Get data byte back
0350  ED69                Out     (C),L           ;Send it
0352  C9                  Ret                     ;END Output
```

```
                    0975
                    0976  ;#############################################################################
                    0977  Zap:    ; Zap memory with a constant
                    0978          ; Forms of this command:
                    0979          ; Z start s swath byte <cr>
                    0980          ; Z start finish byte <cr>
                    0981          ; This command zap the memory specified from
                    0982          ; start for swath bytes (or from start through
                    0983          ; finish) with the specified byte
                    0984  ;#############################################################################
                    0985
0353  CD2E05        0986          Call    Arg2s           ;Get addresses and string for fill
0356  B7            0987          Or      A,A             ;Was a character specified ?
0357  1A            0988          Ld      A,(DE)          ;A = Fill character
0358  5F            0989          Ld      E,A             ;Put in E
0359  2002          0990          Jr      NZ,Zap.10       ;Yes: skip next
035B  1E00          0991  Zap.05: Ld      E,0             ;Use NULL if none specified
035D  73            0992  Zap.10: Ld      (HL),E          ;Save the data byte in memory
035E  54            0993          Ld      D,H             ;Copy start address
035F  5D            0994          Ld      E,L             ;From HL to DE
0360  13            0995          Inc     DE              ;Bump destination to next byte
0361  0B            0996          Dec     BC              ;Adjust swath for first byte
0362  78            0997  Zap.20: Ld      A,B             ;Was "s1"
0363  B1            0998          Or      A,C             ;specified?
0364  C8            0999          Ret     Z               ;Yes, already done
0365  EDB0          1000          Ldir                    ;Zap!
0367  C9            1001          Ret                     ;All done, return
                    1002
                    1003  ;#############################################################################
                    1004  Goto:   ; Program control transfer; "go" command
                    1005          ; Form of this command: G addr <cr>
                    1006          ; This command jumps to the specified address
                    1007  ;#############################################################################
0368  CD0605        1008          Call    Arg1q           ;Get the address
036B  E9            1009          Jp      (HL)            ;END Goto
                    1010
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0027
*** THE CROMEMCO OCTART MONITOR ***
*** FILE OPERATIONS PROCESSING ***

                    1012  ;#############################################################################
                    1013  FileCmd:;File command processing
                    1014          ; Forms of these commands:
                    1015          ; 0     K <cr> (see FAbort command below)
                    1016          ; 1     Fd [filespecs] <cr>
                    1017          ; 2     Fr addr filespecs <cr>
                    1018          ; 3     Fw addr s swath filespecs <cr>
                    1019          ; 4     Fz filespecs <cr>
                    1020          ; 5     Fs [<host dependent>] <cr>
                    1021          ; 6     Fx [<host dependent>] <cr>
                    1022          ; These commands allow file operations through the host's operating
                    1023          ; system using a host-resident file processing program (OCTARTEX.COM).
                    1024          ; There are six commands allowed giving capability for reading, writing
                    1025          ; and deleting files, listing a directory, displaying status
                    1026          ; information about a disk, and a host-dependent command.
                    1027          ; The number listed to the left of each command above is
                    1028          ; the command code transferred to the host system.
                    1029  ;#############################################################################
                    1030
036C  13            1031          Inc     DE              ;Bump past command type
036D  47            1032          Ld      B,A             ;B = Command letter
036E  E5            1033          Push    HL              ;Save Argument #1 default
036F  218407        1034          Ld      HL,FileTbl-2    ;Point to file command look-up table
0372  23            1035  FC.10:  Inc     HL              ;Bump pointer
0373  23            1036          Inc     HL              ;HL --> Command letter from table
0374  7E            1037          Ld      A,(HL)          ;A = Command letter from table
0375  23            1038          Inc     HL              ;HL --> Command address
0376  B7            1039          Or      A,A             ;Is this the end of the command table ?
0377  CA3601        1040          Jp      Z,CmdErr        ;Yes: Print error message and Return
037A  B8            1041          Cp      A,B             ;Is this the command ?
037B  20F5          1042          Jr      NZ,FC.10        ;NO: Loop till valid command or end of table
037D  7E            1043          Ld      A,(HL)          ;A = Numeric argument count
037E  23            1044          Inc     HL              ;Bump table pointer
037F  46            1045          Ld      B,(HL)          ;B = Command byte
                    1046
                    1047          ;************************************************
                    1048          ; At this point, the command code is on the top
                    1049          ; of stack and the A register contains the number
                    1050          ; of numeric arguments.
                    1051          ;************************************************
                    1052
0380  E1            1053          Pop     HL              ;HL = Default for argument #1
0381  C5            1054          Push    BC              ;Put command byte on stack
0382  3D            1055          Dec     A               ;Check if 1 argument
0383  F5            1056          Push    AF              ;Save count for later
0384  45            1057          Ld      B,L             ;Clear B
0385  4D            1058          Ld      C,L             ;BC = Default for argument #2
0386  CC5805        1059          Call    Z,Arg           ;Yes, get 1 argument
0389  F1            1060          Pop     AF              ;Get count back
038A  3D            1061          Dec     A               ;Check if 2 arguments
038B  CC3905        1062          Call    Z,Arg2          ;Yes, get both arguments
038E  CD2304        1063          Call    FM.On           ;Set up for command to host
0391  F1            1064          Pop     AF              ;A = Command code
0392  CD3504        1065          Call    FileOut         ;Send command to host
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0028
*** THE CROMEMCO OCTART MONITOR ***
*** FILE OPERATIONS PROCESSING ***

0395  7C          1066           Ld      A,H             ;Get Arg1 high
0396  CD3504      1067           Call    FileOut         ;Send it to host
0399  7D          1068           Ld      A,L             ;Get Arg1 low
039A  CD3504      1069           Call    FileOut         ;Send it to host
039D  78          1070           Ld      A,B             ;Get Arg2 high
039E  CF03        1071           Jsys    .WrByte         ;Send it to host
03A0  79          1072           Ld      A,C             ;Get Arg2 low
03A1  CF03        1073           Jsys    .WrByte         ;Send it to host
                  1074
                  1075           ;************************************************
                  1076           ; Send the remainder of the command string
                  1077           ; to the host system for it to interpret.
                  1078           ;************************************************
                  1079
03A3  CD9B04      1080           Call    Scan            ;Skip past spaces & comma, if any
03A6  CDAC04      1081  FC.20:   Call    Scan.15         ;Get character & test for EOL,Rem & batch
03A9  13          1082           Inc     DE              ;Bump buffer pointer
03AA  CF03        1083           Jsys    .WrByte         ;No, send the character
03AC  20F8        1084           Jr      NZ,FC.20        ;Loop till accumulator = NUL (end of string)
```

```
                        1086
                        1087  ;#############################################################################
                        1088  FC.40:  ; This is home base for the OCTART while a file command is being
                        1089          ; processed. Commands are received from the host and are
                        1090          ; executed; then control returns here.  The commands are:
                        1091          ; 0 = End of job
                        1092          ; 1 = "Please" (pause for operator intervention)
                        1093          ; 2 = Print the following string terminated by ^7 or binary 0
                        1094          ; 3 = Read a block of data
                        1095          ; 4 = Write a block of data
                        1096  ;#############################################################################
                        1097
03AE  CF01              1098          Jsys    .RdByte         ;Now wait for a command back
03B0  3211BF            1099          Ld      (Abort.F),A     ;If non-zero then no console abort
03B3  B7                1100          Or      A,A             ;Zero ?, aAs in end-of-job?
03B4  285F              1101          Jr      Z,FM.Off        ;Yes, reset terminal & return
03B6  3D                1102          Dec     A               ;Perform a "please"?
03B7  2028              1103          Jr      NZ,FC.80        ;NO: check what else
03B9  CD1504            1104          Call    FM.Off          ;Set up regular console talk
03BC  CF02              1105          Jsys    .Ready          ;Check if character already there
03BE  C46906            1106          Call    NZ,GetChr       ;Yes, discard it
03C1  CF12              1107          Jsys    .GetChr         ;Now wait for the real thing
03C3  CD2304            1108          Call    FM.On           ;Set up for file talk again
03C6  FE03              1109          Cp      A,ETX           ;Is it a ^C ?
03C8  2002              1110          Jr      NZ,FC.60        ;No, echo it back to file system
03CA  3E1B              1111          Ld      A,ESC           ;Yes: Return an ESC
03CC  CF03              1112  FC.60:  Jsys    .WrByte         ;Send the character back
03CE  D61B              1113          Sub     A,ESC           ;Was it an <esc>?
03D0  20DC              1114          Jr      NZ,FC.40        ;No, wait for next command
03D2  3211BF            1115          Ld      (Abort.F),A     ;Yes, clear file abort flag (A = 0)
03D5  DB02              1116  FC.70:  In      A,Hport         ;Get the host status
03D7  E680              1117          And     HTBE            ;Has it gotten the <esc>?
03D9  28FA              1118          Jr      Z,FC.70         ;No, keep waiting
03DB  CD1504            1119          Call    FM.Off          ;Reset for normal talk
03DE  C3D006            1120          Jp      CRLF            ;Go to next line & return
                        1121
03E1  3D                1122  FC.80:  Dec     A               ;Is command to print a string?
03E2  2012              1123          Jr      NZ,FC.A0        ;No, check if I/O
03E4  CF01              1124  FC.90:  Jsys    .RdByte         ;Get a character
03E6  B7                1125          Or      A,A             ;End of string?
03E7  28C5              1126          Jr      Z,FC.40         ;Yes, go wait for next command
03E9  CD1504            1127          Call    FM.Off          ;Turn file mode off
03EC  CF07              1128          Jsys    .ConOut         ;Print the character
03EE  CD2304            1129          Call    FM.On           ;Restore file mode
03F1  F2E403            1130          Jp      P,FC.90         ;If still positive then keep going
03F4  18B8              1131          Jr      FC.40           ;That's all, get next command
                        1132
03F6  FE03              1133  FC.A0:  Cp      A,3             ;Is this a valid command ?
03F8  30B4              1134          Jr      NC,FC.40        ;No: ignore command
```

```
                    1136          ;***********************************************
                    1137          ; Get an address (arg1) and a swath (arg2) from
                    1138          ; the host system and return them in registers
                    1139          ; HL & BC (used with FR & FW commands)
                    1140          ;***********************************************
                    1141
03FA  F5            1142          Push    AF              ;Save command
03FB  CF01          1143          Jsys    .RdByte         ;Get arg1 high
03FD  67            1144          Ld      H,A             ;Save it
03FE  CF01          1145          Jsys    .RdByte         ;Get arg1 low
0400  6F            1146          Ld      L,A             ;Save it
0401  CF01          1147          Jsys    .RdByte         ;Get arg2 high
0403  47            1148          Ld      B,A             ;Save it
0404  CF01          1149          Jsys    .RdByte         ;Get arg2 low
0406  4F            1150          Ld      C,A             ;Save it
0407  F1            1151          Pop     AF              ;Restore command
0408  3D            1152          Dec     A               ;Is it a read command
0409  2805          1153          Jr      Z,RdFile        ;Yes, go process
040B  CD7E07        1154          Call    WrSeq.1         ;Do the write
040E  189E          1155          Jr      FC.40           ;Go get next command
                    1156
0410  CDAA06        1157  RdFile: Call    RdSeq.1         ;Do the read
0413  1899          1158          Jr      FC.40           ;Go get next command
                    1159
                    1160  ;#############################################################################
                    1161  FM.Off: ; Reset file console to original after
                    1162          ; file operations.
                    1163  ;#############################################################################
                    1164
0415  F5            1165          Push    AF              ;Save Acc
0416  3A0ABF        1166          Ld      A,(TempID)      ;Get original console #
0419  3207BF        1167          Ld      (ConID),A       ;Restore original console
041C  2F            1168          Cpl                     ;If 8 channels of OCTART
041D  E620          1169          And     A,QMode         ;then set QMode On and File flag off
041F  D300          1170          Out     Hport-2,A       ;Send status to host
0421  F1            1171          Pop     AF              ;Restore Acc
0422  C9            1172          Ret                     ;END FM.Off
                    1173
                    1174  ;#############################################################################
                    1175  FM.On:  ; Save current # console and set up console as host for
                    1176          ; file operations (set FMode to indicate file operation)
                    1177  ;#############################################################################
                    1178
0423  F5            1179          Push    AF              ;Save AF
0424  E5            1180          Push    HL              ;Save HL
0425  2107BF        1181          Ld      HL,ConID        ;Point to current console #
0428  7E            1182          Ld      A,(HL)          ;Save it for later at
0429  320ABF        1183          Ld      (TempID),A      ;The console temporary
042C  36FF          1184          Ld      (HL),-1         ;Say new console is host
042E  3E04          1185          Ld      A,FMode         ;Set FMode On and QMode off
0430  D300          1186          Out     Hport-2,A       ;Program flag
0432  E1            1187          Pop     HL              ;Restore HL
0433  F1            1188          Pop     AF              ;Restore AF
0434  C9            1189          Ret                     ;END FM.On
```

```
                        1191
                        1192  ;#############################################################################
                        1193  FileOut:; Send a byte to the host system for file commands
                        1194          ; If not responding, then give an error message
                        1195  ;#############################################################################
                        1196
0435  F5                1197          Push    AF              ;Save output byte
0436  C5                1198          Push    BC              ;Save BC (use BC for timeout counter)
0437  011027            1199          Ld      BC,10000        ;BC = Timeout constant
043A  DB02              1200  FO.10:  In      A,(HPort)       ;A = Host ready status
043C  E680              1201          And     HTBE            ;Is host ready for a byte?
043E  2018              1202          Jr      NZ,FO.20        ;Yes, send byte & return
0440  0B                1203          Dec     BC              ;Decrement timeout counter
0441  78                1204          Ld      A,B             ;A = High byte of count
0442  B1                1205          Or      A,C             ;Is counter = 0000 ?
0443  20F5              1206          Jr      NZ,FO.10        ;NO: Loop till counter = 0 or host ready
0445  CD1504            1207          Call    FM.Off          ;Reset file mode, Host system not ready
0448  CF09              1208          Jsys    .WrMsg          ;Print error message
044A  0754696D          1209          defm    BEL,'Timeout\r' ;Error message
                        1210
0453  3211BF            1211          Ld      (Abort.F),A     ;Reset file abort flag
0456  CF00              1212          Jsys    .Exit           ;Abort this job
                        1213
0458  C1                1214  FO.20:  Pop     BC              ;Restore BC
0459  F1                1215          Pop     AF              ;Restore output byte
045A  C36207            1216          Jp      WrByte          ;Send the byte & return
                        1217
                        1218  ;#############################################################################
                        1219  FAbort: ; File processor "kill" command
                        1220          ; Form of this command: K <cr>
                        1221          ; Stop the host system's file processor program
                        1222  ;#############################################################################
                        1223
045D  CD0F05            1224          Call    Eolchk          ;Check if end of line
0460  23                1225  FA.10:  Inc     HL              ;Delay a while for
0461  7C                1226          Ld      A,H             ;The host to
0462  B5                1227          Or      A,L             ;Recover before
0463  20FB              1228          Jr      NZ,FA.10        ;Proceeding
0465  3E04              1229          Ld      A,FMode         ;A = File mode flag
0467  D300              1230          Out     (Hport-2),A     ;Output to host
0469  DB02              1231          In      A,(Hport)       ;A = OCTART/Host status
046B  E680              1232          And     HTBE            ;Is host ready for input ?
046D  2805              1233          Jr      Z,FA.20         ;No: Skip next
046F  AF                1234          Xor     A,A             ;Clear accumulator
0470  D301              1235          Out     (Hport-1),A     ;Clear OCTART-Host status flags
0472  18EC              1236          Jr      FA.10           ;Keep poking until gone
                        1237
0474  D300              1238  FA.20:  Out     (Hport-2),A     ;Clear file operation flag
0476  C35201            1239          Jp      Init.20         ;Initialize new console & return
                        1240
```

```
                    1242  ;#############################################################################
                    1243  Jsys:   ; This routine dispatches user operation requests.  A "Jsys N"
                    1244          ; operation is executed by the user, where N is the function
                    1245          ; code.  Control is transferred here and the function code is
                    1246          ; converted to a routine address.  The pc is adjusted around
                    1247          ; the function code, then the user is sent to the routine they
                    1248          ; requested.  If an invalid function is requested, the job is
                    1249          ; aborted.  The registers are not disturbed.
                    1250  ;#############################################################################
                    1251
0479  E3            1252          Ex      (SP),HL         ;Get pc, save HL
047A  F5            1253          Push    AF              ;Save Acc
047B  E5            1254          Push    HL              ;Put address on stack for Crash routine
047C  7E            1255          Ld      A,(HL)          ;Get operation code
047D  3202BF        1256          Ld      (JsysOp),A      ;Save it for later
0480  FE13          1257          Cp      A,JsysMax       ;Are we out of range?
0482  D23800        1258          Jp      NC,Crash        ;Yes, assume we crashed
0485  E1            1259          Pop     HL              ;HL --> Jsys Opcode
0486  F1            1260          Pop     AF              ;Restore Acc
0487  23            1261          Inc     HL              ;HL --> Return address
0488  E3            1262          Ex      (SP),HL         ;Restore original HL & Return address
                    1263
                    1264          ;************************************************
                    1265          ; The function code has been saved and its range
                    1266          ; has been checked. All registers at this point
                    1267          ; are back to normal.
                    1268          ;************************************************
                    1269
0489  E5            1270          Push    HL              ;Save HL
048A  F5            1271          Push    AF              ;Save Acc
048B  3A02BF        1272          Ld      A,(JsysOp)      ;Get the operation code
048E  87            1273          Add     A,A             ;*2 For jump table
048F  21C707        1274          Ld      HL,JsysTbl      ;Point to jump table
0492  CDFA04        1275          Call    Addh            ;Point to actual entry
0495  F1            1276          Pop     AF              ;Restore Acc
0496  CDFF04        1277          Call    Loadhh          ;HL --> Routine to execute
0499  E3            1278          Ex      (SP),HL         ;Put routine address on stack & restore HL
049A  C9            1279          Ret                     ;Execute routine
                    1280
```

```
                    1282  ;#############################################################################
                    1283  Scan:   ; Skip past tabs & spaces & only 1 comma
                    1284          ; Entry: DE points to string
                    1285          ; Exit: DE points past tabs, spaces & comma
                    1286  ;#############################################################################
                    1287
049B  CDA304        1288          Call    Scan.10         ;Skip tabs and spaces
049E  FE2C          1289          Cp      A,','           ;Do we have a comma?
04A0  2025          1290          Jr      NZ,Scan.25      ;Set zero flag and return
04A2  13            1291  Scan.05:Inc     DE              ;Point past comma
                    1292
                    1293          ;***********************************************
                    1294          ; Skip past tabs & spaces
                    1295          ; Entry: DE points to string
                    1296          ; Exit:  DE points past tabs & spaces
                    1297          ;***********************************************
                    1298
04A3  1A            1299  Scan.10:Ld      A,(DE)          ;Get a character
04A4  FE20          1300          Cp      A,' '           ;Is it a space?
04A6  28FA          1301          Jr      Z,Scan.05       ;Yes, skip next
04A8  FE09          1302          Cp      Tab             ;Is it a <tab>?
04AA  28F6          1303          Jr      Z,Scan.05       ;Yes: skip next
04AC  1A            1304  Scan.15:Ld      A,(DE)          ;A = Character
04AD  ED530FBF      1305          Ld      (BatchPtr),DE   ;Save pointer
04B1  CDD205        1306          Call    UCase           ;Convert to upper case
04B4  FE25          1307          Cp      A,Rem$          ;Is it a remark ?
04B6  280D          1308          Jr      Z,Scan.20       ;Yes: Set zero flag and put marker in line
04B8  FE3B          1309          Cp      A,Cmd$          ;Is this a command delimeter
04BA  200B          1310          Jr      NZ,Scan.25      ;No: skip next
04BC  CD3403        1311          Call    BM.On           ;Turn batch mode on
04BF  13            1312          Inc     DE              ;DE --> Next byte in command line
04C0  ED530FBF      1313          Ld      (BatchPtr),DE   ;Save pointer
04C4  1B            1314          Dec     DE              ;DE --> End of this command line
04C5  AF            1315  Scan.20:Xor     A,A             ;Clear accumulator
04C6  12            1316          Ld      (DE),A          ;Put marker in line
04C7  B7            1317  Scan.25:Or      A,A             ;Set flags
04C8  C9            1318          Ret                     ;END Scan.10
```

```
                    1320
                    1321  ;#############################################################################
                    1322  InStr:  ; Get string of bytes from (DE) & leave final
                    1323          ; String at DE in Input$ with count in reg B
                    1324          ; Entry: DE points to string
                    1325          ; Exit: B contains count of bytes
                    1326          ; DE points to string in Input$
                    1327  ;#############################################################################
                    1328
04C9  E5            1329          Push    HL              ;Save HL
04CA  0600          1330          Ld      B,0             ;Initial count of 0
04CC  2112BF        1331          Ld      HL,Input$       ;Point to binary result buffer
04CF  CD9B04        1332  IS.10:  Call    Scan            ;Skip first comma and any spaces or tabs
04D2  2821          1333          Jr      Z,IS.40         ;Skip next if end of line
04D4  13            1334          Inc     DE              ;DE --> Next character
04D5  4F            1335          Ld      C,A             ;Save character in case delimiter
04D6  FE27          1336          Cp      A,''''          ; Is it single quote?
04D8  2816          1337          Jr      Z,IS.30         ;Yes, process string
04DA  FE22          1338          Cp      A,'"'           ;Is it double quote?
04DC  2812          1339          Jr      Z,IS.30         ;Yes, process string
04DE  1B            1340          Dec     DE              ;DE --> First character of argument
04DF  E5            1341          Push    HL              ;Save binary pointer
04E0  CF0D          1342          Jsys    .Arg            ;Get argument
04E2  7D            1343          Ld      A,L             ;Get result into a
04E3  E1            1344          Pop     HL              ;Restore binary pointer
04E4  380F          1345          Jr      C,IS.40         ;If no argument, then error
04E6  77            1346          Ld      (HL),A          ;Save argument
04E7  23            1347          Inc     HL              ;Bump binary pointer
04E8  04            1348          Inc     B               ;Bump count
04E9  4F            1349          Ld      C,A             ;Force C = A
04EA  B9            1350  IS.50:  Cp      A,C             ;Is it the delimiter?
04EB  28E2          1351          Jr      Z,IS.10         ;Yes, end of string
04ED  77            1352          Ld      (HL),A          ;Save the character
04EE  23            1353          Inc     HL              ;Bump binary pointer
04EF  04            1354          Inc     B               ;Bump count
04F0  1A            1355  IS.30:  Ld      A,(DE)          ;Get the character
04F1  13            1356          Inc     DE              ;Bump buffer pointer
04F2  B7            1357          Or      A,A             ;Is it EOL?
04F3  20F5          1358          Jr      NZ,IS.50        ;No: Skip next
04F5  1112BF        1359  IS.40:  Ld      DE,Input$       ;Point to beginning of buffer
04F8  E1            1360          Pop     HL              ;Restore HL
04F9  C9            1361          Ret                     ;& Return
                    1362
```

```
                1364  ;#############################################################################
                1365  Addh:   ; Add A to the contents of HL
                1366          ; HL = HL+A
                1367  ;#############################################################################
                1368
04FA  85        1369          Add     L               ;Add l
04FB  6F        1370          Ld      L,A             ;Resave it
04FC  D0        1371          Ret     NC              ;EXIT: No overflow
04FD  24        1372          Inc     H               ;Adjust h
04FE  C9        1373          Ret                     ;END Addh
                1374
                1375  ;#############################################################################
                1376  Loadhh: ; Load HL with that pointed to by HL
                1377          ; HL = (HL)
                1378  ;#############################################################################
                1379
04FF  F5        1380          Push    AF              ;Save Acc
0500  7E        1381          Ld      A,(HL)          ;Get low byte
0501  23        1382          Inc     HL              ;Bump pointer
0502  66        1383          Ld      H,(HL)          ;Get high byte
0503  6F        1384          Ld      L,A             ;Shuffle low byte
0504  F1        1385          Pop     AF              ;Restore AF
0505  C9        1386          Ret                     ;END Loadhh
                1387
```

```
                ;##########################################################################
                        ; Entry: DE points to argument string,
                        ; Leading spaces & tabs are ignored
                        ; Certain defaults may be allowed in BC
                        ; And HL, depending on routine called
                        ; Exit: DE points just past argument string,
                        ; Except in the case of arg3q, then
                        ; DE contains argument 3
                        ; BC contains swath, if required
                        ; (Argument 2 minus argument 1)
                        ; HL contains argument 1
                        ; A has the character breaking the string
                        ; For argx and argxd carry flag set
                        ; Indicates no argument given
                        ; Argxx get argument(s)
                        ; Xxx1x HL = arg
                        ; Xxx2x HL = arg1, BC = arg2-arg1
                        ; Xxx2x HL = arg1, BC = swath
                        ; Xxx3q HL = arg1, BC = arg2-arg1, DE = arg3
                        ; Xxx3q HL = arg1, BC = swath, DE = arg3
                        ; Xxxxq if no arg then error
                        ; Xxxxd if no arg then default
                        ; Xxxxq or xxxxd
                        ; If no eol then error
                ;##########################################################################

                        ; Get 1 argument, defaults not allowed

0506  CD0D05    Arg1q:  Call    Arg1d           ;Get number
0509  D0        Argcmc: Ret     NC              ;Number given, return
050A  C33601    Argcme: Jp      Cmderr          ;None given, error

                        ; Get 1 argument, defaults allowed

050D  CF0D      Arg1d:  Jsys    .Arg            ;Get (maybe) argument

                ;##########################################################################
                EolChk: ; Check if end of line, dispatch to error routine if not
                ;##########################################################################

050F  F5                Push    AF              ;Save Acc
0510  CDA304            Call    Scan.10         ;Check if end of command line
0513  20F5              Jr      NZ,Argcme       ;No, error
0515  F1                Pop     AF              ;Restore Acc
0516  C9                Ret                     ;END EolChk

                        ; Get 2 arguments, defaults not allowed

0517  CD1C05    Arg2q:  Call    Arg2d           ;Get 2 arguments
051A  18ED              Jr      Argcmc          ;Check if any given
```

```
                    1440
                    1441            ; Get 2 arguments, defaults allowed
                    1442
051C  CF0E          1443  Arg2d:    Jsys    .Arg2           ;Get 2 arguments
051E  18EF          1444            Jr      EolChk          ;& Go check eol
                    1445
                    1446            ; Get 3 arguments, 1st argument default allowed,
                    1447            ; Swath & 3rd argument defaults not allowed
                    1448
0520  CF0E          1449  Arg3q:    Jsys    .Arg2           ;Get first 2 arguments
0522  38E6          1450            Jr      C,Argcme        ;Error if none given
0524  E5            1451            Push    HL              ;Save 1st argument
0525  CF0D          1452            Jsys    .Arg            ;Get 3rd argument
0527  CD0F05        1453            Call    Eolchk          ;Check for eol
052A  EB            1454            Ex      DE,HL           ;Move 3rd argument into place
052B  E1            1455            Pop     HL              ;Restore 1st argument
052C  18DB          1456            Jr      Argcmc          ;Go check if all args given
                    1457
                    1458            ; Get 2 arguments, defaults not allowed, eol not checked for
                    1459
052E  CF0E          1460  Arg2s:    Jsys    .Arg2           ;Get first 2 arguments
0530  38D8          1461            Jr      C,ArgCme        ;Error if no arguments
0532  C5            1462            Push    BC              ;Save Swath
0533  CDC904        1463            Call    Instr           ;Convert 3rd argument to string
0536  78            1464            Ld      A,B             ;A = Count
0537  C1            1465            Pop     BC              ;Restore swath
0538  C9            1466            Ret                     ;END Arg2s
```

```
                    1468
                    1469            ; Get 2 arguments, defaults allowed, eol not checked for
                    1470
0539  CF0D          1471  Arg2:     Jsys    .Arg            ;Get 1st argument
                    1472
                    1473            ;***********************************************
                    1474            ; Get swath operator, default allowed
                    1475            ; Entry: DE points to string
                    1476            ; HL contains argument 1
                    1477            ; Exit: BC contains swath
                    1478            ; (Argument 2 minus argument 1)
                    1479            ; DE points just past string
                    1480            ; HL contains argument 1
                    1481            ;***********************************************
                    1482
053B  CD9B04        1483            Call    Scan            ;Skip to argument
053E  FE53          1484            Cp      A,'S'           ;Is it a true swath?
0540  E5            1485            Push    HL              ;(Save argument 1)
0541  280C          1486            Jr      Z,Args1         ;Yes: get swath argument
0543  CF0D          1487            Jsys    .Arg            ;Get argument 2
0545  380F          1488            Jr      C,Argsr         ;None, use default
0547  C1            1489            Pop     BC              ;Get a copy of
0548  C5            1490            Push    BC              ;Argument 1 into BC
0549  ED42          1491            SBC     HL,BC           ;Argument 1 from argument 2
054B  23            1492            Inc     HL              ;Adjust swath for the end points
054C  B7            1493            Or      A,A             ;Set break character flags
054D  1805          1494            Jr      Argsr1          ;Return
                    1495
054F  13            1496  Args1:    Inc     DE              ;Bump past "s"
0550  CF0D          1497            Jsys    .Arg            ;Get the swath
0552  38B6          1498            Jr      C,Argcme        ;ERROR: "S" with no argument
0554  44            1499  Argsr1:   Ld      B,H             ;Move swath
0555  4D            1500            Ld      C,L             ;From HL to BC
0556  E1            1501  Argsr:    Pop     HL              ;Restore argument 1
0557  C9            1502            Ret                     ;END ArgS
```

```
                    1504
                    1505          ;***********************************************
                    1506          ; Numeric argument radix conversion routine
                    1507          ; Used by argument processing routines
                    1508          ; Entry: DE points to string
                    1509          ; HL contains default
                    1510          ; Exit: DE points just past string
                    1511          ; HL contains value (or default if none given)
                    1512          ; A contains break character
                    1513          ;***********************************************
                    1514
0558  CD9B04        1515  Arg:     Call    Scan            ;Look for non-space
055B  CDA805        1516          Call    Hex             ;Check if valid hex/numeric character
055E  D8            1517          Ret     C               ;No, return with default
055F  D5            1518          Push    DE              ;Save the pointer
0560  E5            1519          Push    HL              ;Save any default
0561  CDB805        1520          Call    Hexin           ;Check for breaker
0564  E1            1521          Pop     HL              ;Restore default
0565  D1            1522          Pop     DE              ;& Pointer
0566  FE2E          1523          Cp      A,'.'           ;Is it decimal point?
0568  1A            1524          Ld      A,(DE)          ;Get first character
0569  280A          1525          Jr      Z,Argh1         ;Yes, it was decimal point
056B  CDB805        1526          Call    Hexin           ;Get new value
056E  FE48          1527          Cp      A,'H'           ;Was terminator an "H"?
0570  37            1528          SCF                     ;Set carry flag
0571  2813          1529          Jr      Z,Argh12        ;Yes, skip over the "H"
0573  B7            1530          Or      A,A             ;Reset carry flag
0574  C9            1531          Ret                     ;END Arg
                    1532
0575  CDB105        1533  Argh1:   Call    Numeric         ;Is first character numeric?
0578  D8            1534          Ret     C               ;No, return with default
0579  CDA304        1535          Call    Scan.10         ;Skip spaces
057C  210000        1536          Ld      HL,0000         ;Start with 0
057F  CDAC04        1537  Argh11:  Call    Scan.15         ;Check for EOL, Batch, remarks
0582  C8            1538          Ret     Z               ;END of line
0583  CDB105        1539          Call    Numeric         ;Is it numeric?
0586  13            1540  Argh12:  Inc     DE              ;Bump character pointer
0587  3003          1541          Jr      NC,Argh2        ;Yes: continue conversion
0589  C3AC04        1542          Jp      Scan.15         ;Test for EOL, Command, Rem
                    1543
058C  D630          1544  Argh2:   Sub     A,'0'           ;Convert to binary
058E  C5            1545          Push    BC              ;Save BC
058F  44            1546          Ld      B,H             ;Make copy of
0590  4D            1547          Ld      C,L             ;HL into BC
0591  29            1548          Add     HL,HL           ;*2
0592  29            1549          Add     HL,HL           ;*4
0593  09            1550          Add     HL,BC           ;*5
0594  29            1551          Add     HL,HL           ;*10
0595  C1            1552          Pop     BC              ;Restore BC
0596  CDFA04        1553          Call    Addh            ;Add in new digit
0599  18E4          1554          Jr      Argh11          ;And go for next
                    1555
```

```
                    1557  ;#############################################################################
                    1558  HexBin: ; Convert ASCII hex value in a to binary
                    1559          ; Entry: A contains character
                    1560          ; Exit:  A contains binary value
                    1561  ;#############################################################################
                    1562
059B  CDD205        1563          Call    UCase           ;Convert ASCII to upper case
059E  D630          1564          Sub     A,'0'           ;Convert to binary
05A0  FE0A          1565          Cp      A,10            ;Was it "a-f"?
05A2  D8            1566          Ret     C               ;No, return
05A3  D607          1567          Sub     A,7             ;Convert for 10-15
05A5  FE10          1568          Cp      A,10h           ;Check if valid
05A7  C9            1569          Ret                     ;END HexBin
                    1570
                    1571  ;#############################################################################
                    1572  Hex:    ; Verify if character in A is valid ASCII hex
                    1573          ; Entry: A contains character
                    1574          ; Exit: A contains character
                    1575          ; Carry flag reset if ok, else
                    1576          ; carry flag set if not ok
                    1577  ;#############################################################################
                    1578
05A8  C5            1579          Push    BC              ;Save BC
05A9  47            1580          Ld      B,A             ;Save character in B
05AA  CD9B05        1581          Call    HexBin          ;Convert to hex
05AD  78            1582          Ld      A,B             ;A = Original character
05AE  C1            1583          Pop     BC              ;Restore BC
05AF  3F            1584          CCF                     ;Adjust carry flag
05B0  C9            1585          Ret                     ;END Hex
                    1586
                    1587  ;#############################################################################
                    1588  Numeric:; Verify if character in A is valid ASCII numeric
                    1589          ; Entry: A contains character
                    1590          ; Exit: A contains character
                    1591          ; Carry flag reset if ok, else
                    1592          ; carry flag set if not ok
                    1593  ;#############################################################################
                    1594
05B1  FE30          1595          Cp      A,'0'           ;Is it <"0" ?
05B3  D8            1596          Ret     C               ;Yes, return
05B4  FE3A          1597          Cp      A,'9'+1         ;Is it >"9" ?
05B6  3F            1598          CCF                     ;Adjust carry flag
05B7  C9            1599          Ret                     ;END Numeric
```

```
                    1601
                    1602  ;#############################################################################
                    1603  HexIn:  ; Convert the ASCII hex string
                    1604          ; At DE to a binary value in HL
                    1605          ; Entry: DE points to string
                    1606          ; Exit:  HL contains value
                    1607          ; A contains break character
                    1608  ;#############################################################################
                    1609
05B8  CDA304        1610          Call    Scan.10         ;Skip spaces
05BB  210000        1611          Ld      HL,0000         ;Start with 0
05BE  CDAC04        1612  HI.10:  Call    Scan.15         ;Check for EOL, batch and remarks
05C1  C8            1613          Ret     Z               ;END of line
05C2  CDA805        1614          Call    Hex             ;Valid hex?
05C5  D8            1615          Ret     C               ;No, return
05C6  13            1616          Inc     DE              ;Bump ascii pointer
05C7  CD9B05        1617          Call    HexBin          ;Convert character to binary
05CA  29            1618          Add     HL,HL           ;Make room for new digit
05CB  29            1619          Add     HL,HL           ;By shifting 4 bits
05CC  29            1620          Add     HL,HL           ;Which is like
05CD  29            1621          Add     HL,HL           ;Multiplying by 10h
05CE  85            1622          Add     A,L             ;Combine new digit
05CF  6F            1623          Ld      L,A             ;& Resave it
05D0  18EC          1624          Jr      HI.10           ;Go for next digit
                    1625
                    1626  ;#############################################################################
                    1627  UCase:  ; Convert character in A to uppercase
                    1628  ;#############################################################################
                    1629
05D2  FE61          1630          Cp      A,'a'           ;Is it <"a"?
05D4  D8            1631          Ret     C               ;Yes: EXIT
05D5  FE7B          1632          Cp      'z'+1           ;Is it >"z"?
05D7  D0            1633          Ret     NC              ;Yes: EXIT
05D8  E65F          1634          And     A,5Fh           ;Convert to uppercase
05DA  C9            1635          Ret
                    1636
```

```
CROMEMCO Z80 Macro Assembler version 03.08                    Feb 21. 1984  15:28:53                    Page 0042
*** THE CROMEMCO OCTART MONITOR ***
** CONSOLE STATUS SUBROUTINES **

                  ;##########################################################################
                  Select: ; This routine selects the console in the accumulator as follows
                          ; FFh    = Host
                          ; 51h    = channel 0   59h   = channel 1
                          ; 11h    = channel 2   19h   = channel 3
                          ; 21h    = channel 4 , 29h   = channel 5
                          ; 31h    = channel 6 , 39h   = channel 7
                          ; Exit: A = -1 if character ready (Z reset)
                          ;       A =  0 if no character ready (Z set)
                  ;##########################################################################

05DB  3207BF              Ld      (ConID),A       ;Select Console in accumulator

                  ;##########################################################################
                  Ready:  ; Test if console has input character ready
                  ConRdy: ; Exit: A = 0 if none, Z flag is set
                          ; A contains -1 if ready, Z flag is reset
                  ;##########################################################################

05DE  C5                  Push    BC              ;Save BC
05DF  CF0B                Jsys    .ConGen         ;Get the hardware info
05E1  ED78                In      A,(C)           ;Get status
05E3  A0                  And     A,B             ;Check if anything there
05E4  C1                  Pop     BC              ;Restore BC
05E5  C8                  Ret     Z               ;Nothing, return
05E6  3EFF                Ld      A,True          ;Set Ready flag
05E8  C9                  Ret                     ;END Ready
```

```
                    1666
                    1667  ;#############################################################################
                    1668  ConGen: ; Return hardware information for current console
                    1669          ; Current console number is determined as follows:
                    1670          ; (ConID) =  -1, console is the host
                    1671          ;
                    1672          ; Exit: C = status port #  (Decrement for data port #)????
                    1673          ;       B = input mask
                    1674          ;       A = output mask
                    1675          ;
                    1676          ; The host command port is checked for an abort command each time
                    1677          ; this loop is entered.  If the abort command is detected (0FFh),
                    1678          ; a Restart 0 is executed and all tasks are aborted.
                    1679  ;#############################################################################
                    1680
                    1681          ;***********************************************
                    1682          ; Check if abort command received from HOST
                    1683          ;***********************************************
                    1684
05E9  DB02          1685          In      A,(HPort)       ;A = Status flags
05EB  E601          1686          And     A,1             ;Is there a command waiting
05ED  280A          1687          Jr      Z,CG.05         ;No: skip next
05EF  DB00          1688          In      A,(HPort-2) .   ;A = Host command
05F1  3C            1689          Inc     A               ;TASK ABORT COMMAND ?
05F2  2005          1690          Jr      NZ,CG.05        ;No: skip next
05F4  3E02          1691          Ld      A,2             ;A = Command clear bit
05F6  D300          1692          Out     (HPort-2),A     ;Clear command flag
05F8  C7            1693          Rst     0               ;Abort all tasks and return to monitor
                    1694
05F9  3A07BF        1695  CG.05:  Ld      A,(ConID)       ;Get the console #
05FC  87            1696          Add     A,A             ;Set console flags
05FD  010240        1697          Ld      BC,HRDA<<8|HPort;Load parameters
0600  3E80          1698          Ld      A,HTBE          ;for the host system
0602  D8            1699          Ret     C               ;Carry set = Host
0603  3A07BF        1700          Ld      A,(ConID)       ;A = OCTART channel number
0606  4F            1701          Ld      C,A             ;C --> Control port for SIO channel
0607  0601          1702          Ld      B,QRDA          ;B = RDA mask
0609  3E04          1703          Ld      A,QTBE          ;A = TBE mask
060B  C9            1704          Ret                     ;END ConGen
                    1705
```

```
                    1707  ;#############################################################################
                    1708  RdLine: ; Input a buffered line of characters
                    1709          ; Entry: DE points to line buffer
                    1710          ; A contains maximum length
                    1711          ; Exit: DE points to line buffer
                    1712          ; A contains actual entered length
                    1713          ; Input line is terminated with a NUL
                    1714  ;#############################################################################
                    1715
060C  C5            1716          Push    BC              ;Save BC
060D  E5            1717          Push    HL              ;Save HL
060E  4F            1718          Ld      C,A             ;Save maximum length
060F  0600          1719  RL.10:   Ld      B,0             ;Initial length of 0
0611  62            1720          Ld      H,D             ;Point HL to
0612  6B            1721          Ld      L,E             ;Line buffer
0613  CF12          1722  RL.20:   Jsys    .GetChr         ;Get a character
0615  28FC          1723          Jr      Z,RL.20         ;If <NUL>, then ignore it
0617  FE0D          1724          Cp      A,CR            ;Is it Carriage return ?
0619  283A          1725          Jr      Z,RL.60         ;Yes, end of line
061B  FE08          1726          Cp      A,BS            ;Is it BackSpace?
061D  2804          1727          Jr      Z,RL.30         ;Yes, delete a character
061F  FE7F          1728          Cp      A,DEL           ;Is it <del>?
0621  2010          1729          Jr      NZ,RL.40        ;No, check something else
0623  05            1730  RL.30:   Dec     B               ;Decrement counter
0624  FA0F06        1731          Jp      M,RL.10         ;Cannot backspace past beginning of line
0627  CDD406        1732          Call    BackSp          ;Blank out character
062A  2B            1733          Dec     HL              ;Backspace buffer pointer
062B  7E            1734          Ld      A,(HL)          ;A = Character under cursor
062C  FE20          1735          Cp      A,' '           ;Was it a control character?
062E  DCD406        1736          Call    C,BackSp        ;Yes, blank out the "^"
0631  18E0          1737          Jr      RL.20           ;Go get another character
                    1738
0633  F5            1739  RL.40:   Push    AF              ;Save character
0634  CDFF06        1740          Call    OutEcho         ;Output character, Echo controls as ^Chr
0637  F1            1741          Pop     AF              ;Restore it
0638  FE03          1742          Cp      A,ETX           ;Is it ^C?
063A  2804          1743          Jr      Z,RL.50         ;Yes, abort the job
063C  FE1A          1744          Cp      A,SUB           ;Is it ^Z?
063E  2007          1745          Jr      NZ,RL.55        ;No: Skip next
0640  AF            1746  RL.50:   Xor     A,A             ;Clear accumulator
0641  3201BF        1747          Ld      (Break+2),A     ;Clear debug flag & break point
0644  C33A01        1748          Jp      ReEntx          ;Abort any current task
                    1749
0647  FE15          1750  RL.55:   Cp      A,NAK           ;Is it ^U ?
0649  CCD006        1751          Call    Z,CRLF          ;Go to next line if so
064C  28C1          1752          Jr      Z,RL.10         ;Yes, scrap this line & get another
064E  77            1753          Ld      (HL),A          ;Save the character
064F  23            1754          Inc     HL              ;Bump pointer
0650  04            1755          Inc     B               ;& Count
0651  78            1756          Ld      A,B             ;Get maximum length
0652  91            1757          Sub     A,C             ;Are we there yet?
0653  20BE          1758          Jr      NZ,RL.20        ;No, go for another character
0655  AF            1759  RL.60:   Xor     A,A             ;Clear accumulator
0656  77            1760          Ld      (HL),A          ;Mark end of line
```

```
0657  23          1761          Inc     HL              ;with two
0658  77          1762          Ld      (HL),A          ;nulls.
0659  B0          1763          Or      A,B             ;A = Final length, Z flag set if no input
065A  E1          1764          Pop     HL              ;Get HL back
065B  C1          1765          Pop     BC              ;And BC also
065C  1872        1766          Jr      CRLF            ;Print CR,LF
                  1767
                  1768  ;#########################################################################
                  1769  ConIn:  ; Get a processed (echoed) character from current console
                  1770          ; Exit: a contains the character
                  1771  ;#########################################################################
                  1772
065E  CF12        1773          Jsys    .GetChr         ;Get a character
0660  FE7F        1774          Cp      A,DEL           ;Is it <del>?
0662  C8          1775          Ret     Z               ;Yes: END ConIn
0663  F5          1776          Push    AF              ;Save Acc
0664  CDFF06      1777          Call    OutEcho         ;Echo character
0667  F1          1778          Pop     AF              ;Restore Acc
0668  C9          1779          Ret                     ;
                  1780
                  1781  ;#########################################################################
                  1782  GetChr: ; Get a single character
                  1783          ; Exit: a contains the character
                  1784  ;#########################################################################
                  1785
0669  CF01        1786          Jsys    .RdByte         ;Get a byte
066B  E67F        1787  gcp:    And     7Fh             ;Mask off parity
066D  C9          1788          Ret                     ;& Return
                  1789
                  1790  ;#########################################################################
                  1791  RdByte: ; Read one absolute byte from current console
                  1792          ; Return input byte in register A
                  1793  ;#########################################################################
                  1794
066E  CF02        1795          Jsys    .Ready          ;Is console input ready ?
0670  28FC        1796          Jr      Z,RdByte        ;NO: Loop till ready
0672  C5          1797  grab:   Push    BC              ;Save BC
0673  CF0B        1798          Jsys    .ConGen         ;Get the hardware info
0675  0D          1799          dec     c               ;Host Consol data port
0676  3A07BF      1800          ld      a,(conid)       ;get consol id
0679  3C          1801          inc     a
067A  2803        1802          jr      z,grb20         ;if zero --> Host
067C  0C          1803          inc     c               ;
067D  0C          1804          inc     c
067E  0C          1805          inc     c               ;C --> #1 #2 Consol data port
067F  ED78        1806  grb20:  In      A,(C)           ;A = Input byte
0681  C1          1807          Pop     BC              ;Restore BC
0682  C9          1808          Ret                     ;END RdByte
```

```
CROMEMCO Z80 Macro Assembler version 03.08                    Feb 21, 1984  15:28:53                    Page 0046
*** THE CROMEMCO OCTART MONITOR ***
*** CONSOLE INPUT ROUTINES ***

                  ;#######################################################################
                  Gettim: ; Get a single character (Timed)
                          ; i.e. first, strip out all old characters from the SIO, next,
                          ; start the 240 ms countdown timer, next, wait for a new character
                          ; to arrive. If a new character doesn't arrive within 240 ms.,
                          ; return to caller with the Z flag set to indicate a timeout.
                          ;
                          ; Exit: A contains the character (Parity bit stripped)
                          ;       Z flag set if timeout occurred
                  ;#######################################################################

0683  C5                  push    bc
0684  CF02        gettz:  jsys    .ready          ; Strip out the SIO buffers
0686  2805                jr      z,getts         ;   /
0688  CD7206              call    grab            ;  /
068B  18F7                jr      gettz           ; /
068D  0618        getts:  ld      b,24            ;240 ms.
068F  CF02        gettx:  Jsys    .Ready          ;Is console input ready ?
0691  200E                jr      nz,getty        ;Yes: go get the character

                          ; Approx. 10 millisecond wait loop

0693  E5                  push    hl              ;
      (0000)          If test                     ;
                          ld      hl,1537         ; 1537 value is for RAM-based operation
                      Else                        ;
0694  210905              ld      hl,1289         ; 1289 value is for ROM-based operation
                      Endif                       ;
0697  2B          wl:     dec     hl              ; \
0698  7C                  ld      a,h             ;  \
0699  B5                  or      a,l             ;   \
069A  20FB                jr      nz,wl           ; Count 'em down
069C  E1                  pop     hl              ;   /
069D  10F0                djnz    gettx           ;  /
069F  C1                  pop     bc              ; /
06A0  C9                  ret

06A1  CD7206      getty:  call    grab            ; We have a character ready, so get it
06A4  C1                  pop     bc              ; Restore BC
06A5  18C4                jr      gcp             ; Go strip high bit and return

                  ;#######################################################################
                  RdSeq:  ; Read sequential binary data from the console into memory
                          ; Forms of this command:
                          ; R start s swath <cr>
                          ; R start finish <cr>
                          ; This command reads binary data from the current
                          ; console into the OCTART memory as specified
                  ;#######################################################################

06A7  CD1705              Call    Arg2q           ;Get start & swath
06AA  CF01        RdSeq.1:Jsys    .RdByte         ;Get a byte
06AC  77                  Ld      (HL),A          ;Save it in memory
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0047
*** THE CROMEMCO OCTART MONITOR ***
*** CONSOLE INPUT ROUTINES ***

06AD  EDA1        1864          CPI                        ;Bump pointer & check count
06AF  E0          1865          Ret     PO                 ;Done, return
06B0  18F8        1866          Jr      RdSeq.1            ;Keep reading
                  1867
```

```
                    1869  ;#############################################################################
                    1870  WrLine: ; Print string pointed to by HL
                    1871          ; String terminates with either a NULL or a byte with bit 7 high.
                    1872          ; Entry: HL points to string
                    1873  ;#############################################################################
                    1874
06B2  F5            1875          Push    AF              ;Save AF
06B3  7E            1876  WL.10:  Ld      A,(HL)          ;Get a character
06B4  23            1877          Inc     HL              ;Bump pointer
06B5  B7            1878          Or      A,A             ;Is it he end?
06B6  2805          1879          Jr      Z,WL.20         ;Yes, return
06B8  CF07          1880          Jsys    .ConOut         ;Print the character
06BA  F2B306        1881          Jp      P,WL.10         ;If bit 7 not high then go print more
06BD  F1            1882  WL.20:  Pop     AF              ;Restore AF
06BE  C9            1883          Ret                     ;END WrLine
                    1884
                    1885  ;#############################################################################
                    1886  WrMsg:  ; Print string immediately following call
                    1887          ; String terminates with either a NULL or the parity bit high
                    1888          ; Entry: string immediately follows the call
                    1889  ;#############################################################################
                    1890
06BF  E3            1891          Ex      (SP),HL         ;HL --> Message string
06C0  CF08          1892          Jsys    .WrLine         ;Print the string
06C2  E3            1893          Ex      (SP),HL         ;Put return address on stack
06C3  C9            1894          Ret                     ;END WrMsg
                    1895
                    1896  ;#############################################################################
                    1897  MoveCsr:; Move print head over to column specified in register B
                    1898          ; Entry: B contains column number
                    1899  ;#############################################################################
                    1900
06C4  F5            1901          Push    AF              ;Save AF
06C5  3A08BF        1902  MC.10:  Ld      A,(Column)      ;Get current position
06C8  B8            1903          Cp      A,B             ;Are we where we want to go?
06C9  30F2          1904          Jr      NC,WL.20        ;Yes: Pop Acc and return
06CB  CDFD06        1905          Call    Space           ;No, move over a column
06CE  18F5          1906          Jr      MC.10           ;Keep moving until done
                    1907
                    1908  ;#############################################################################
                    1909  CRLF:   ; Print a newline sequence <CR>,<LF>
                    1910  ;#############################################################################
                    1911
06D0  CF09          1912          Jsys    .WrMsg          ;Print the following
06D2  8D            1913          defm    CR              ;CRLF
06D3  C9            1914          Ret                     ;END CRLF
                    1915
                    1916  ;#############################################################################
                    1917  BackSp: ; Print a destructive backspace sequence   <BS>,<SPC>,<BS>
                    1918  ;#############################################################################
                    1919
06D4  CF09          1920          Jsys    .WrMsg          ;Use a short cut
06D6  082088        1921          defm    BS,' ',BS       ;Destructive BackSpace
06D9  C9            1922          Ret                     ;END BackSp
```

```
                    1924
                    1925  ;##############################################################################
                    1926  Prt2Hex:; Print 2 byte binary value in HL as hex
                    1927          ; Entry: HL contains value
                    1928          ; Register A gets destroyed
                    1929  ;##############################################################################
                    1930
06DA  7C            1931          Ld      A,H             ;Get high byte
06DB  CF0F          1932          Jsys    .PrtHex         ;Print it
06DD  7D            1933          Ld      A,L             ;Get low byte
                    1934
                    1935  ;##############################################################################
                    1936  PrtHex: ; Print binary value in a to console as hex
                    1937          ; Entry: a contains value
                    1938          ; Register A gets destroyed
                    1939  ;##############################################################################
                    1940
06DE  F5            1941          Push    AF              ;Save byte for later
06DF  0F            1942          Rrca                    ;Get
06E0  0F            1943          Rrca                    ;Left
06E1  0F            1944          Rrca                    ;Side
06E2  0F            1945          Rrca                    ;Nybble
06E3  CDE706        1946          Call    PH.10           ;Print it
06E6  F1            1947          Pop     AF              ;Get byte back
06E7  E60F          1948  PH.10:   And     A,0Fh           ;Mask off for right nibble
06E9  C630          1949          Add     A,'0'           ;Convert to ascii
06EB  FE3A          1950          Cp      A,'9'+1         ;Is it "a-f"?
06ED  3802          1951          Jr      C,PH.20         ;No, go print it
06EF  C607          1952          Add     A,7             ;Convert to "a-f"
06F1  181C          1953  PH.20:   Jr      ConOut          ;& Print it
                    1954
                    1955  ;##############################################################################
                    1956  HexBO:  ; Print a space, then print the binary
                    1957          ; value in A to console as hex
                    1958          ; Entry:   A contains value
                    1959          ; Register A gets destroyed
                    1960  ;##############################################################################
                    1961
06F3  F5            1962          Push    AF              ;Save hex value
06F4  CDFD06        1963          Call    Space           ;Print the space
06F7  F1            1964          Pop     AF              ;Get hex value back
06F8  18E4          1965          Jr      PrtHex          ;& Print it
```

```
                1967
                1968  ;#########################################################################
                1969  HexBS:  ; Print a space then binary then another space
                1970  ;#########################################################################
                1971
06FA  CDF306    1972          Call    HexBO           ;Print space and binary
                1973
                1974  ;#########################################################################
                1975  Space:  ; Print a space
                1976  ;#########################################################################
                1977
06FD  3E20      1978          Ld      A,' '           ;Load the space
                1979
                1980  ;#########################################################################
                1981  OutEcho:; Echo character in accumulator, Print '^' leader if control character
                1982          ; Entry: A contains the character
                1983          ; Exit: A gets destroyed
                1984  ;#########################################################################
                1985
06FF  FE20      1986          Cp      A,' '           ;Check if graphic character
0701  300C      1987          Jr      NC,ConOut       ;Yes, go print it
0703  FE0D      1988          Cp      A,CR            ;Is it a <cr> ?
0705  2808      1989          Jr      Z,ConOut        ;Yes, go print it
0707  F5        1990          Push    AF              ;Save character
0708  3E5E      1991          Ld      A,'^'           ;Accumulator = '^'
070A  CF07      1992          Jsys    .ConOut         ;Print it
070C  F1        1993          Pop     AF              ;Get character back
070D  C640      1994          Add     '@'             ;Convert to graphic
```

```
                        1996
                        1997  ;#############################################################################
                        1998  ConOut: ; Output a processed character via WrByte & check for characters typed.
                        1999          ; Also, adjust console position & check for batch errors
                        2000          ; Entry: A contains character
                        2001          ; All registers preserved unless output is aborted
                        2002  ;#############################################################################
                        2003
070F  F5                2004          Push    AF              ;Save character & flags
0710  E67F              2005          And     A,7Fh           ;Mask off parity
0712  E5                2006          Push    HL              ;Save HL
0713  2108BF            2007          Ld      HL,Column       ;Point to cursor position
0716  FE08              2008          Cp      A,BS            ;Is it <bs>?
0718  2003              2009          Jr      NZ,CO.30        ;No, check for <tab>
071A  35                2010          Dec     (HL)            ;Back up cursor pointer
071B  1817              2011          Jr      CO.60           ;Go print the <bs>
071D  FE09              2012  CO.30:  Cp      A,Tab           ;Is it <tab>?
071F  200A              2013          Jr      NZ,CO.50        ;No, check for graphic
0721  CDFD06            2014  CO.40:  Call    Space           ;Print a space
0724  3E07              2015          Ld      A,07h           ;Load mask for <tab> position
0726  A6                2016          And     A,(HL)          ;Are we there yet?
0727  20F8              2017          Jr      NZ,CO.40        ;No, keep printing
0729  1815              2018          Jr      CO.70           ;Yes, return
072B  FE7F              2019  CO.50:  Cp      A,DEL           ;Is it <del>?
072D  2805              2020          Jr      Z,CO.60         ;Yes, just print it
072F  FE20              2021          Cp      A,' '           ;Is it graphic?
0731  3801              2022          Jr      C,CO.60         ;No, just print it
0733  34                2023          Inc     (HL)            ;Yes, bump cursor pointer
0734  CF03              2024  CO.60:  Jsys    .WrByte         ;Print the character
0736  FE0D              2025          Cp      A,CR            ;Is it <cr>?
0738  2006              2026          Jr      NZ,CO.70        ;No, check if can be aborted
073A  3600              2027          Ld      (HL),0          ;Yes, say cursor position at 0
073C  3E0A              2028          Ld      A,LF            ;Load a <lf>
073E  CF07              2029          Jsys    .ConOut         ;Echo it also
0740  E1                2030  CO.70:  Pop     HL              ;Restore HL
0741  3A11BF            2031          Ld      A,(Abort.F)     ;Get abort flag
0744  B7                2032          Or      A,A             ;Are we allowed to abort?
0745  2019              2033          Jr      NZ,CO.80        ;No, restore character & return
0747  CF02              2034          Jsys    .Ready          ;Is anything waiting for us?
0749  2815              2035          Jr      Z,CO.80         ;No, return
074B  CF12              2036          Jsys    .GetChr         ;Get what it was
074D  FE13              2037          Cp      A,DC3           ;Is it pause?
074F  CC6906            2038          Call    Z,GetChr        ;Yes, wait for something
0752  FE03              2039          Cp      A,ETX           ;Control-C?
0754  2802              2040          Jr      Z,CO.75         ;Yes, echo it & go back to command
0756  FE1A              2041          Cp      A,SUB           ;Control-C?
0758  CA3306            2042  CO.75:  Jp      Z,RL.40         ;Yes, echo it & go back to command
075B  FE1B              2043          Cp      A,ESC           ;Is it <esc> to abort?
075D  CA3A01            2044          Jp      Z,ReEntx        ;Yes, go back to command level
0760  F1                2045  CO.80:  Pop     AF              ;Restore character
0761  C9                2046          Ret                     ;& Return
```

```
                2048
                2049  ;#############################################################################
                2050  WrByte: ; Write byte in accumulator to current console
                2051          ; Entry: A contains the byte
                2052          ; All registers preserved
                2053  ;#############################################################################
                2054
0762  C5         2055          Push    BC              ;Save BC
0763  F5         2056          Push    AF              ;Save character in accumulator
0764  CF0B       2057          Jsys    .ConGen         ;Get the hardware info
0766  47         2058          Ld      B,A             ;B = Output ready mask
0767  ED78       2059  WB.10:  In      A,(C)           ;A = Console status
0769  A0         2060          And     A,B             ;Is console ready for output ?
076A  28FB       2061          Jr      Z,WB.10         ;NO: Loop till ready
076C  0D         2062          Dec     C               ;C --> Host Console data port
076D  3A07BF     2063          ld      a,(conid)       ;get consol id
0770  3C         2064          inc     a
0771  2803       2065          jr      z,wb.20
0773  0C         2066          inc     c
0774  0C         2067          inc     c
0775  0C         2068          inc     c               ;C --> #1 #2 consol data port
0776  F1         2069  wb.20:  Pop     AF              ;A = Output character
0777  ED79       2070          Out     (C),A           ;Send the character
0779  C1         2071          Pop     BC              ;Restore BC
077A  C9         2072          Ret                     ;END WrByte
                2073
                2074  ;#############################################################################
                2075  WrSeq:  ; Write binary data from memory to the console
                2076          ; Forms of this command:
                2077          ; W start s swath <cr>
                2078          ; W start finish <cr>
                2079          ; This command writes binary data from the OCTART
                2080          ; memory specified to the current console
                2081  ;#############################################################################
                2082
077B  CD1705     2083          Call    Arg2q           ;Get start & swath
077E  7E         2084  WrSeq.1:Ld      A,(HL)          ;Get a byte from memory
077F  CF03       2085          Jsys    .WrByte         ;Send it
0781  EDA1       2086          CPI                     ;Bump pointer & check count
0783  E0         2087          Ret     PO              ;Done, return
0784  18F8       2088          Jr      WrSeq.1         ;Keep writing
                2089
```

```
                    2091  ;#########################################################################
                    2092  FileTbl:; File command parameter look-up table
                    2093  ;#########################################################################
                    2094
0786  44            2095          defb    'D'             ;Directory command
0787  0001          2096          defb    00h,01h         ;No numeric args, 1 = Dir
                    2097
0789  52            2098          defb    'R'             ;Read command
078A  0102          2099          defb    01h,02h         ;1 numeric arg, 2 = Read
                    2100
078C  53            2101          defb    'S'             ;Status command
078D  0005          2102          defb    00h,05h         ;No numeric args, 5 = Status
                    2103
078F  57            2104          defb    'W'             ;Write command
0790  0203          2105          defb    02h,03h         ;2 numeric args, 3 = Write
                    2106
0792  58            2107          defb    'X'             ;Special command
0793  0006          2108          defb    00h,06h         ;No numeric args, 6 = Special
                    2109
0795  5A            2110          defb    'Z'             ;Delete command
0796  0004          2111          defb    00h,04h         ;No numeric args, 4 = Delete
                    2112
0798  00            2113          defb    0               ;END of FileTbl
                    2114
```

```
                    2116  ;#############################################################################
                    2117  CmdTbl: ; Command routine address table
                    2118  ;#############################################################################
                    2119
0799  40            2120          defb    '@'             ;@ - Submit for batch processing
079A  2D03          2121          defw    Batch.1         ;    Batch routine address
                    2122
079C  44            2123          defb    'D'             ;D - Display memory
079D  D302          2124          defw    Display         ;    Display routine address
                    2125
079F  45            2126          defb    'E'             ;E - Examine input port
07A0  3D03          2127          defw    Examine         ;    Examine routine address
                    2128
07A2  46            2129          defb    'F'             ;F - File operations processing
07A3  6C03          2130          defw    FileCmd         ;    File operations routine address
                    2131
07A5  47            2132          defb    'G'             ;G - Goto memory location
07A6  6803          2133          defw    Goto            ;    Goto routine address
                    2134
07A8  49            2135          defb    'I'             ;I - Initialize console
07A9  4F01          2136          defw    Init            ;    Initialize routine address
                    2137
07AB  4B            2138          defb    'K'             ;K - Kill file processor
07AC  5D04          2139          defw    FAbort          ;    Kill routine address
                    2140
07AE  4D            2141          defb    'M'             ;M - Move memory
07AF  4202          2142          defw    Move            ;    Move routine address
                    2143
07B1  4F            2144          defb    'O'             ;O - Output to port
07B2  4803          2145          defw    Output          ;    Output routine address
                    2146
07B4  51            2147          defb    'Q'             ;Q - Query memory
07B5  AA02          2148          defw    Query           ;    Query routine address
                    2149
07B7  52            2150          defb    'R'             ;R - Read binary data
07B8  A706          2151          defw    RdSeq           ;    Read routine address
                    2152
07BA  53            2153          defb    'S'             ;S - Set memory
07BB  6702          2154          defw    Setmem          ;    Set memory routine address
                    2155
07BD  56            2156          defb    'V'             ;V - Verify memory
07BE  4A02          2157          defw    Verify          ;    Verify routine address
                    2158
07C0  57            2159          defb    'W'             ;W - Write binary data
07C1  7B07          2160          defw    WrSeq           ;    Write routine address
                    2161
07C3  5A            2162          defb    'Z'             ;Z - Zap memory
07C4  5303          2163          defw    Zap             ;    Zap routine address
                    2164
07C6  00            2165          defb    00              ;END of Command Table
                    2166
```

```
                    2168  ;#############################################################################
                    2169  JsysTbl:; Jsys entry look-up table
                    2170          ; This table is used for user entry processing;
                    2171          ; Each of the addresses listed below
                    2172          ; corresponds to a user function.
                    2173  ;#############################################################################
                    2174
07C7  F600          2175          defw    Exit            ;00:Re-enter the monitor
07C9  6E06          2176          defw    RdByte          ;01:Input a pure byte
07CB  DE05          2177          defw    Ready           ;02:Check for pure byte ready
07CD  6207          2178          defw    WrByte          ;03:Output a pure byte
07CF  0C06          2179          defw    RdLine          ;04:Input a buffered line
07D1  5E06          2180          defw    ConIn           ;05:Input a processed character
07D3  DE05          2181          defw    ConRdy          ;06:Check for character ready
07D5  0F07          2182          defw    ConOut          ;07:Output a processed character
07D7  B206          2183          defw    WrLine          ;08:Output a string
07D9  BF06          2184          defw    WrMsg           ;09:Output immediate string
07DB  D006          2185          defw    Crlf            ;0A:Output a <cr><lf> sequence
07DD  E905          2186          defw    ConGen          ;0B:Get console hardware info
07DF  3003          2187          defw    Batch           ;0C:Set up a batch job
07E1  5805          2188          defw    Arg             ;0D:Convert a single argument
07E3  3905          2189          defw    Arg2            ;0E:Convert two arguments
07E5  DE06          2190          defw    PrtHex          ;0F:Display number in reg. A on console in hex
07E7  DA06          2191          defw    Prt2Hex         ;10:Display number in HL on console in hex
                    2192
                    2193  ;These are new functions
                    2194
07E9  DB05          2195          defw    Select          ;11:Select Console channel in accumulator
07EB  6906          2196          defw    GetChr          ;12:Input processed character no echo
                    2197  ;;;;    defw    OCTARTInit      ;13:Initialize all devices connected to OCTART
      (0013)        2198  JsysMax:defv    [$-JsysTbl]/2   ;Total # of functions
                    2199
                    2200
      (FFFF)        2201     if not test
07ED                2202          Origin  0800h
      (0013)        2203+ Swath:  defv    [Origin + 0800h - $]
      (FFFF)        2204+   If Swath > 0
07ED  (0013)        2205+         Rept    Swath
                    2206+         Defb    0FFh            ;Fill in empty spaces
                    2207+         Mend
07ED  FF            2208+         Defb    0FFh            ;Fill in empty spaces
07EE  FF            2209+         Defb    0FFh            ;Fill in empty spaces
07EF  FF            2210+         Defb    0FFh            ;Fill in empty spaces
07F0  FF            2211+         Defb    0FFh            ;Fill in empty spaces
07F1  FF            2212+         Defb    0FFh            ;Fill in empty spaces
07F2  FF            2213+         Defb    0FFh            ;Fill in empty spaces
07F3  FF            2214+         Defb    0FFh            ;Fill in empty spaces
07F4  FF            2215+         Defb    0FFh            ;Fill in empty spaces
07F5  FF            2216+         Defb    0FFh            ;Fill in empty spaces
07F6  FF            2217+         Defb    0FFh            ;Fill in empty spaces
07F7  FF            2218+         Defb    0FFh            ;Fill in empty spaces
07F8  FF            2219+         Defb    0FFh            ;Fill in empty spaces
07F9  FF            2220+         Defb    0FFh            ;Fill in empty spaces
07FA  FF            2221+         Defb    0FFh            ;Fill in empty spaces
```

```
CROMEMCO Z80 Macro Assembler version 03.08                    Feb 21. 1984  15:28:53                    Page 0056
*** THE CROMEMCO OCTART MONITOR ***
*** JSYS ENTRY LOOKUP TABLE ***

07FB  FF          2222+          Defb    OFFh            ;Fill in empty spaces
07FC  FF          2223+          Defb    OFFh            ;Fill in empty spaces
07FD  FF          2224+          Defb    OFFh            ;Fill in empty spaces
07FE  FF          2225+          Defb    OFFh            ;Fill in empty spaces
07FF  FF          2226+          Defb    OFFh            ;Fill in empty spaces
                  2227+    EndIf
      (0000)      2228+    If Swath < 0
                  2229+           MNote Program Origin Address Overlap @ 0800h
                  2230+    EndIf
                  2231      endif
                  2232
```

```
      (BEFF)      2234          Org     TopRam-100h     ;Variables at the top page of RAM
                  2235  ;#########################################################################
                  2236  ;
                  2237  ;                       RAM-BASED VARIABLE TABLE
                  2238  ;
                  2239  ;#########################################################################
                  2240
                  2241  Vartbl:
BEFF  (0003)      2242  Break:  defs    3               ;Debug break transfer (jp nn)
BF02  (0001)      2243  JsysOp: defs    1               ;Current Jsys opcode
BF03  (0002)      2244  DmPtr:  defs    2               ;Display memory command address
BF05  (0002)      2245  SmPtr:  defs    2               ;Substitute memory  command address
BF07  (0001)      2246  ConID:  defs    1               ;Current console #      -1 = host
BF08  (0001)      2247  Column: defs    1               ;Console column location
BF09  (0001)      2248  ChrBfr: defs    1               ;****** NOT Used In This Version ********
BF0A  (0001)      2249  TempID: defs    1               ;Temporary storage for console #
BF0B  (0002)      2250  Return: defs    2               ;Return execution address
BF0D  (0000)      2251  BatchDat:defs   0               ;Reference for batch data block
BF0D  (0001)      2252  Batch.F:defs    1               ;Indicates batch job in progress
BF0E  (0001)      2253  BatchErr:defs   1               ;Indicates error occurred
BF0F  (0002)      2254  BatchPtr:defs   2               ;Batch command line pointer
BF11  (0001)      2255  Abort.F:defs    1               ;Abort inhibit flag (File operations)
      (0048)      2256  Length: defv    72              ;Console input line length
BF12  (0049)      2257  Input$: defs    Length+1        ;Console input line buffer
                  2258
      (BFDF)      2259          Org     TopRam-20h
BFDF  (0000)      2260  Stack:  defs    0               ;Monitor stack pointer
BFDF  (0002)      2261  UserJsys:defs   2               ;Address of user subroutine for Rst.08 (Jsys)
BFE1  (0002)      2262  User1:  defs    2               ;Address of user subroutine for Rst.10
BFE3  (0002)      2263  User2:  defs    2               ;Address of user subroutine for Rst.18
BFE5  (0002)      2264  User3:  defs    2               ;Address of user subroutine for Rst.20
BFE7  (0002)      2265  User4:  defs    2               ;Address of user subroutine for Rst.28
BFE9  (0002)      2266  UserNMI:defs    2               ;Address of user subroutine for NMI
BFEB  (0001)      2267  Errbit: defs    1               ;error byte
BFEC  (0001)      2268  QBase:  defs    1               ;Base port address of active OCTART
BFED  (0000)      2269  UserRam:defs    0               ;User available free space
                  2270
                  2271  *Include        OCTJSYS.Z80
```

```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21, 1984  15:28:53          Page 0058
*** THE CROMEMCO OCTART MONITOR ***
*** OCTJSYS.Z80 -- OCT JSYS DEFINITION FILE ***

                2273  ;#############################################################################
                2274  ;#                                                                           #
                2275  ;#                      OCTART Jsys Function Definitions                     #
                2276  ;#                                                                           #
                2277  ;#############################################################################
(0000)          2278  .Exit:          Equ     00      ;Abort job & return to command level
(0001)          2279  .RdByte:        Equ     01      ;Input an 8 bit byte from current console
(0002)          2280  .Ready:         Equ     02      ;Check if current console has character ready
(0003)          2281  .WrByte:        Equ     03      ;Output an 8 bit byte to current console
(0004)          2282  .Rdline:        Equ     04      ;Read buffered line into (DE), Acc=max length
(0005)          2283  .ConIn:         Equ     05      ;Get processed character from current console
(0006)          2284  .ConRdy:        Equ     06      ;Check if processed character ready
(0007)          2285  .ConOut:        Equ     07      ;Output a processed character
(0008)          2286  .WrLine:        Equ     08      ;Output the string (HL)
(0009)          2287  .WrMsg:         Equ     09      ;Output the message (SP)+1
(000A)          2288  .CRLF:          Equ     10      ;Output a newline sequence
(000B)          2289  .ConGen:        Equ     11      ;Return console hardware information
(000C)          2290  .Batch:         Equ     12      ;Execute batch string (HL)
(000D)          2291  .Arg:           Equ     13      ;Get 1 arg from str @ DE into HL
(000E)          2292  .Arg2:          Equ     14      ;Get 2 args from str @ DE into HL, BC
(000F)          2293  .PrtHex:        Equ     15      ;Print # in Acc on console in hex
                2294  ;
                2295  ; The following four functions are newly-added with monitor version 03.00
                2296  ;
(0010)          2297  .Prt2Hex:       Equ     16      ;Print # in HL on console in hex
(0011)          2298  .Select:        Equ     17      ;Select console in Acc
(0012)          2299  .GetChr:        Equ     18      ;Get processed character without echo
                      (***** end of include *****)
                2300
```

```
                2302
(0007)          2303  Bel:    Equ     7               ;ASCII bell
(0008)          2304  Bs:     Equ     8               ;ASCII backspace
(000D)          2305  Cr:     Equ     13              ;ASCII carriage return
(0013)          2306  DC3:    Equ     13h             ;ASCII DC3
(007F)          2307  Del:    Equ     7Fh             ;ASCII delete
(001B)          2308  Esc:    Equ     1Bh             ;ASCII escape
(0003)          2309  Etx:    Equ     3               ;ASCII end-of-text
(000A)          2310  Lf:     Equ     0Ah             ;ASCII linefeed
(0015)          2311  Nak:    Equ     15h             ;ASCII negative ack
(001A)          2312  Sub:    Equ     1Ah             ;ASCII substitute
(0009)          2313  Tab:    Equ     9               ;ASCII tab
(0025)          2314  Rem$:   Equ     '%'             ;Command line remark specifier
(003B)          2315  Cmd$:   Equ     ';'             ;Command separator/delimeter
(0004)          2316  FMode:  Equ     00000100b       ;Tells host file operations are in progress
(0020)          2317  QMode:  Equ     00100000b       ;Tells host Octart is being used
(0008)          2318  Ramerr: Equ     00001000b       ;Tells host of a RAM test error
(0010)          2319  Romerr: Equ     00010000b       ;Tells host of a ROM checksum error
                2320
                2321          ;I/O port equates
                2322  .
(0002)          2323  Hport:  Equ     02h             ;Host console status port
(0000)          2324  base    Equ     0h
                2325  ;Write register for chip #1 channel A
(0050)          2326  MR1A    Equ     50h             ;channel mode register 1
(0050)          2327  MR2A    Equ     50h             ;channel mode register 2
(0051)          2328  CSRA    Equ     51h             ;clock select register A
(0052)          2329  CRA     Equ     52h             ;command register A
(0053)          2330  THRA    Equ     53h             ;TX holding register
(0054)          2331  ACR     Equ     54h             ;Aux. control register
(0055)          2332  IMR     Equ     55h             ;interrupt mask register
(0056)          2333  CTUR    Equ     56h             ;C/T upper register
(0057)          2334  CTLR    Equ     57h             ;C/T lower register
                2335  ;Status port for 8 channels
(0051)          2336  STAT_1A Equ     51H             ;chip #1 channel A status port
(0059)          2337  STAT_1B Equ     59H             ;chip #1 channel B status port
(0011)          2338  STAT_2A Equ     11H             ;chip #2 channel A status port
(0019)          2339  STAT_2B Equ     19H             ;chip #2 channel B status port
(0021)          2340  STAT_3A Equ     21H             ;chip #3 channel A status port
(0029)          2341  STAT_3B Equ     29H             ;chip #3 channel B status port
(0031)          2342  STAT_4A Equ     31H             ;chip #4 channel A status port
(0039)          2343  STAT_4B Equ     39H             ;chip #4 channel B status port
                2344
                2345          ;I/O mask equates
(0040)          2346  HRDA:   Equ     01000000b       ;Host receive data available
(0080)          2347  HTBE:   Equ     10000000b       ;Host transmitter buffer empty
(0001)          2348  QRDA:   Equ     00000001b       ;Quadart receive data available
(0004)          2349  QTBE:   Equ     00000100b       ;Quadart transmitter buffer empty
                2350
                2351
BFED  (0000)    2352          End     OCTMon

Errors           0
Range Count      1
```

CROMEMCO Z80 Macro Assembler version 03.08 Feb 21, 1984 15:28:53 Page 0061
*** THE CROMEMCO OCTART MONITOR ***

Symbol	Value	Defn	References
.	BFED	2322	
.Arg	000D	2291	0967 1342 1423 1452 1471 1487 1497
.Arg2	000E	2292	1443 1449 1460
.Batch	000C	2290	
.ConGen	000B	2289	1658 1798 2057
.ConIn	0005	2283	
.ConOut	0007	2285	0927 1128 1880 1992 2029
.ConRdy	0006	2284	
.CRLF	000A	2288	0755
.Exit	0000	2278	1212
.GetChr	0012	2299	1107 1722 1773 2036
.Prt2Hex	0010	2297	0279 0747 0753 0773 0903
.PrtHex	000F	2293	0958 1932
.RdByte	0001	2279	1098 1124 1143 1145 1147 1149 1786 1862
.RdLine	0004	2282	0472 0778
.Ready	0002	2280	1105 1795 1823 1828 2034
.Select	0011	2298	0558 0591
.WrByte	0003	2281	1071 1073 1083 1112 2024 2085
.WrLine	0008	2286	1892
.WrMsg	0009	2287	0276 0443 0464 0502 0707 0712 0795 1208 1912 1920
Abort.F	BF11	2255	1099 1115 1211 2031
ACR	0054	2331	0645
Addh	04FA	1365	1275 1553
Arg	0558	1515	1059 2188
Arg1d	050D	1423	0771 1417
Arg1q	0506	1417	0939 0955 0969 1008
Arg2	0539	1471	1062 2189
Arg2d	051C	1443	0870 1437
Arg2q	0517	1437	1861 2083
Arg2s	052E	1460	0819 0986
Arg3q	0520	1449	0729 0743
Argcmc	0509	1418	1438 1456
Argcme	050A	1419	1431 1450 1461 1498
Argh1	0575	1533	1525
Argh11	057F	1537	1554
Argh12	0586	1540	1529
Argh2	058C	1544	1541
Args1	054F	1496	1486
Argsr	0556	1501	1488
Argsr1	0554	1499	1494
Arnd66	0084	0367	
BackSp	06D4	1917	1732 1736
base	0000	2324	
Batch	0330	0940	2187
Batch.1	032D	0933	2121
Batch.F	BF0D	2252	0461 0943
BatchDat	BF0D	2251	
BatchErr	BF0E	2253	
BatchPtr	BF0F	2254	0459 0941 1305 1313
BEL	0007	2303	0503 0796 1209
BM.Off	033A	0946	0513 0772
BM.On	0334	0942	1311
BM.Set	0336	0943	0947
Break	BEFF	2242	0250 0379 0381 0517 1747

CROMEMCO Z80 Macro Assembler version 03.08 Feb 21, 1984 15:28:53 Page 0062
*** THE CROMEMCO OCTART MONITOR ***

Symbol	Value	Defn	References
BS	0008	2304	1726 1921 1921 2008
CG.05	05F9	1695	1687 1690
ChrBfr	BF09	2248	
ckerror	021A	0703	0442
Cmd$	003B	2315	1309
CmdErr	0136	0499	0482 0823 1040 1419
CmdTbl	0799	2117	0476
CO.30	071D	2012	2009
CO.40	0721	2014	2017
CO.50	072B	2019	2013
CO.60	0734	2024	2011 2020 2022
CO.70	0740	2030	2018 2026
CO.75	0758	2042	2040
CO.80	0760	2045	2033 2035
Column	BF08	2247	0514 1902 2007
ConGen	05E9	1668	2186
ConID	BF07	2246	0102 1167 1181 1649 1695 1700 1800 2063
ConIn	065E	1769	2180
ConOut	070F	1998	1953 1987 1989 2182
ConRdy	05DE	1653	2181
CR	000D	2305	0277 0447 0615 0708 0713 1724 1913 1988 2025
CRA	0052	2329	0639 0657
Crash	0038	0269	0380 1258
CRLF	06D0	1909	0516 0604 0959 1120 1751 1766 2185
CRLF.1	0345	0959	0929
CRTest	01AF	0609	0560 0593
crtto	01B7	0615	0613
CSRA	0051	2328	0636
CTLR	0057	2334	0654
CTUR	0056	2333	0651
DC3	0013	2306	2037
Default	4000	0069	0298 0375 0494
DEL	007F	2307	0922 1728 1774 2019
Display	02D3	0857	2124
DL.10	0304	0907	0914
DL.30	0319	0919	0928
DL.40	0325	0926	0923
DL.50	0327	0927	0925
DM.10	02D3	0868	
DM.20	02DC	0871	0893
DM.30	02EA	0881	0874 0877 0879
DM.40	02F9	0891	0889
DmPtr	BF03	2244	0377 0869 0884
DspLine	02FE	0898	0844 0883
Eolchk	050F	1426	0540 1224 1444 1453
err10	022E	0709	0706
errbit	BFEB	2267	0313 0351 0357 0552 0704 0709
ESC	001B	2308	1111 1113 2043
ETX	0003	2309	1109 1742 2039
Examine	033D	0950	2127
Exit	00F6	0452	2175
FA.10	0460	1225	1228 1236
FA.20	0474	1238	1233
FAbort	045D	1219	2139

```
Symbol      Value  Defn  References

False       0000   0032  0033 0040
FC.10       0372   1035  1042
FC.20       03A6   1081  1084
FC.40       03AE   1088  1114 1126 1131 1134 1155 1158
FC.60       03CC   1112  1110
FC.70       03D5   1116  1118
FC.80       03E1   1122  1103
FC.90       03E4   1124  1130
FC.A0       03F6   1133  1123
FileCmd     036C   1013  2130
FileOut     0435   1193  1065 1067 1069
FileTbl     0786   2092  1034
FM.Off      0415   1161  1101 1104 1119 1127 1207
FM.On       0423   1175  1063 1108 1129
FMode       0004   2316  1185 1229
FO.10       043A   1200  1206
FO.20       0458   1214  1202
gcp         066B   1787  1850
GetChr      0669   1782  1106 2038 2196
Gettim      0683   1812  0612 0614
getts       068D   1827  1824
gettx       068F   1828  1844
getty       06A1   1848  1829
gettz       0684   1823  1826
Goto        0368   1004  2133
grab        0672   1797  1825 1848
grb20       067F   1806  1802
Hex         05A8   1572  1516 1614
HexBin      059B   1558  1581 1617
Hexbo       06F3   1956  0911 1972
HexBS       06FA   1969  0749 0751 0775
Hexin       05B8   1603  1520 1526
HI.10       05BE   1612  1624
HPORT       0002   2323  0550 0556 0596 0603 1116 1170 1186 1200 1230 1231 1235 1238 1685 1688 1692 1697
HRDA        0040   2346  1697
HTBE        0080   2347  1117 1201 1232 1698
ilist       0210   0686  0629
IMR         0055   2332  0648
inisc       01BA   0628  0371
Init        014F   0530  2136
Init.20     0152   0546  0253 0441 0587 1239
Init.25     0153   0547  0549
init.27     0162   0555  0553
Init.30     016F   0566  0559
Init.50     0199   0590  0568 0570 0573 0575 0578 0580 0583 0585
init.55     01AA   0601  0599
Init.70     01AC   0604  0561
Input$      BF12   2257  0470 0776 0829 1331 1359
Instr       04C9   1322  0787 1463
IS.10       04CF   1332  1351
IS.30       04F0   1355  1337 1339
IS.40       04F5   1359  1333 1345
IS.50       04EA   1350  1358
Jsys        0479   1243  0364
```

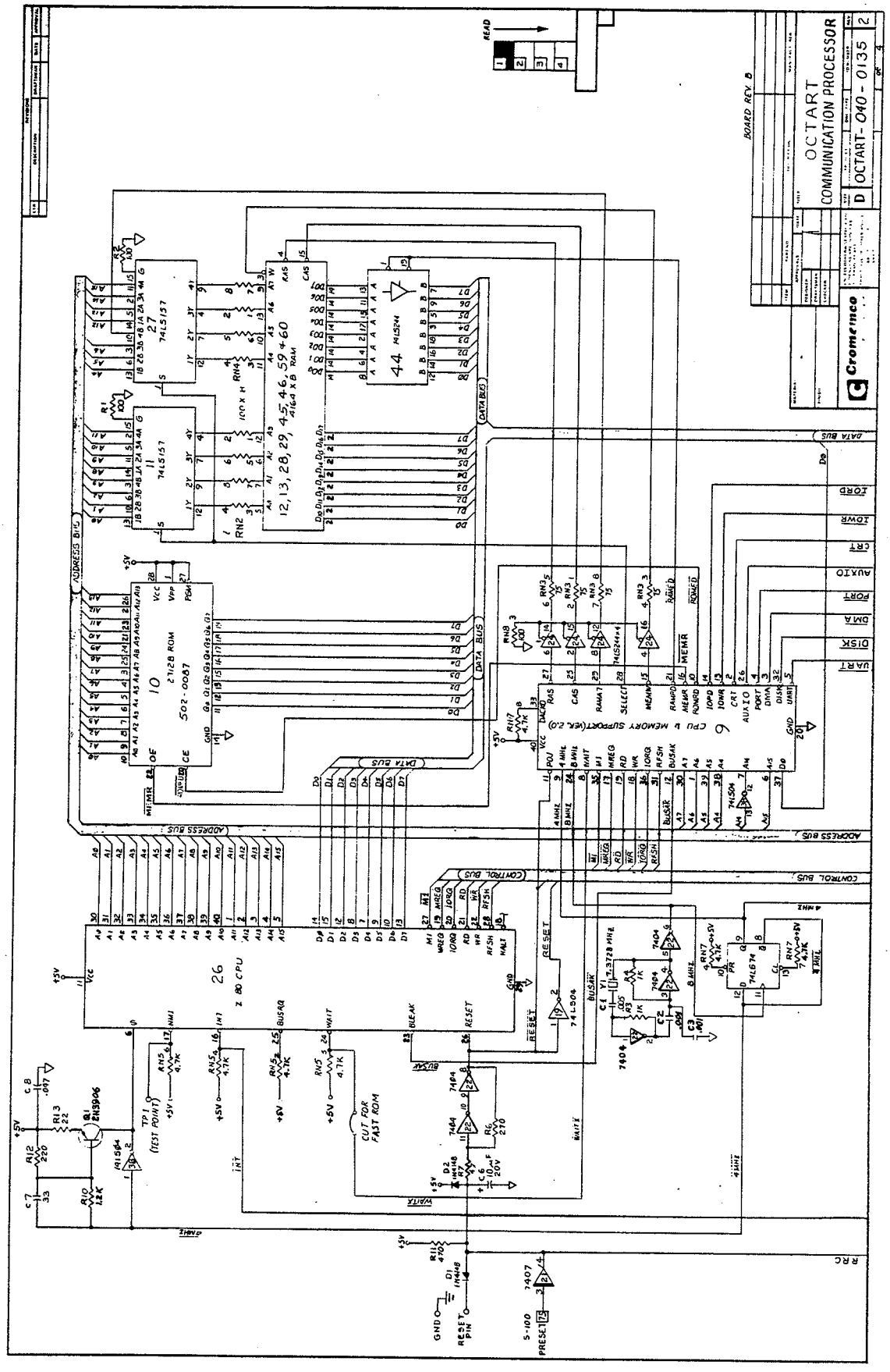
CROMEMCO Z80 Macro Assembler version 03.08 Feb 21, 1984 15:28:53 Page 0064
*** THE CROMEMCO OCTART MONITOR ***

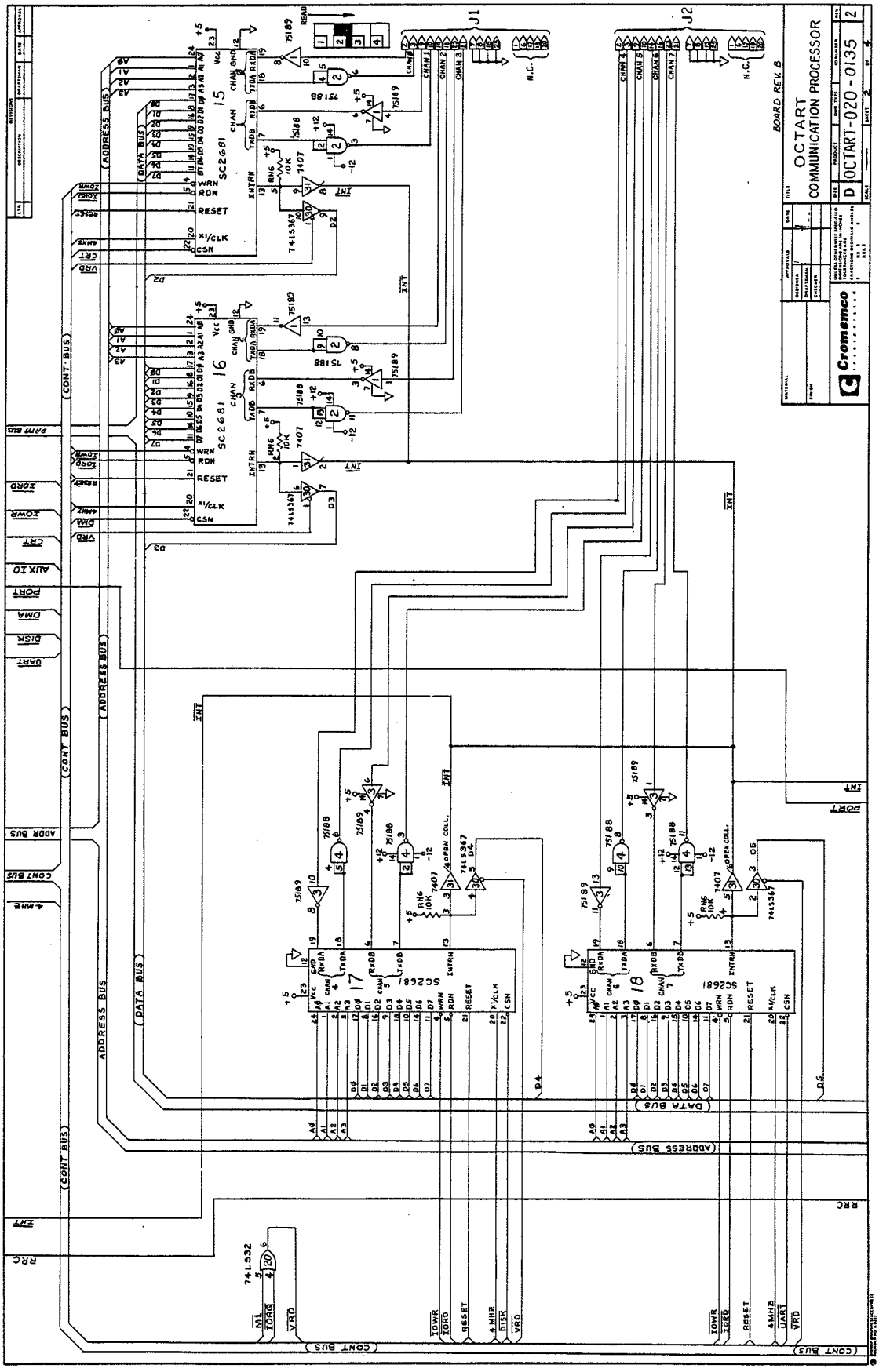
Symbol	Value	Defn	References
JsysMax	0013	2198	1257
JsysOp	BF02	2243	1256 1272
JsysTbl	07C7	2169	1274 2198
Length	0048	2256	0471 0777 2257
lf	000A	2310	0708 0713 2028
Loadhh	04FF	1376	0485 1277
MC.10	06C5	1902	1906
MonErr1	0076	0356	0307
MonErr2	006C	0351	0404
Move	0242	0721	2142
MoveCsr	06C4	1897	0916
MR1A	0050	2326	0630
MR2A	0050	2327	0633
NAK	0015	2311	1750
Numeric	05B1	1588	1533 1539
OCTMon	0000	0095	2352
Origin	0000	0068	0093 0107 0137 0160 0185 0209 0233 0257 0318 0392 2203
out20	0201	0674	0663 0669
OutEcho	06FF	1981	1740 1777
outport	01F0	0661	0631 0634 0637 0640 0643 0646 0649 0652 0655 0658
Output	0348	0962	2145
PH.10	06E7	1948	1946
PH.20	06F1	1953	1951
Prt2Hex	06DA	1926	2191
PrtHex	06DE	1936	1965 2190
QBase	BFEC	2268	
QM.20	02B4	0826	0852
QM.30	02BA	0830	0835
QM.40	02C2	0841	0832
QMode	0020	2317	0598 1169
QRDA	0001	2348	1702
QTBE	0004	2349	1703
Query	02AA	0809	2148
Ramerr	0008	2318	0356
RdByte	066E	1791	1796 2176
RdFile	0410	1157	1153
RdLine	060C	1708	2179
RdSeq	06A7	1853	2151
RdSeq.1	06AA	1862	1157 1866
RE.10	010E	0473	0463
RE.20	0117	0477	0484
RE.30	012D	0491	0489
Ready	05DE	1652	2177
ReEntry	00F6	0451	0491 0523
ReEntx	013A	0506	0474 1748 2044
Rem$	0025	2314	1307
Return	BF0B	2250	0524
RL.10	060F	1719	1731 1752
RL.20	0613	1722	1723 1737 1758
RL.30	0623	1730	1727
RL.40	0633	1739	1729 2042
RL.50	0640	1746	1743
RL.55	0647	1750	1745
RL.60	0655	1759	1725

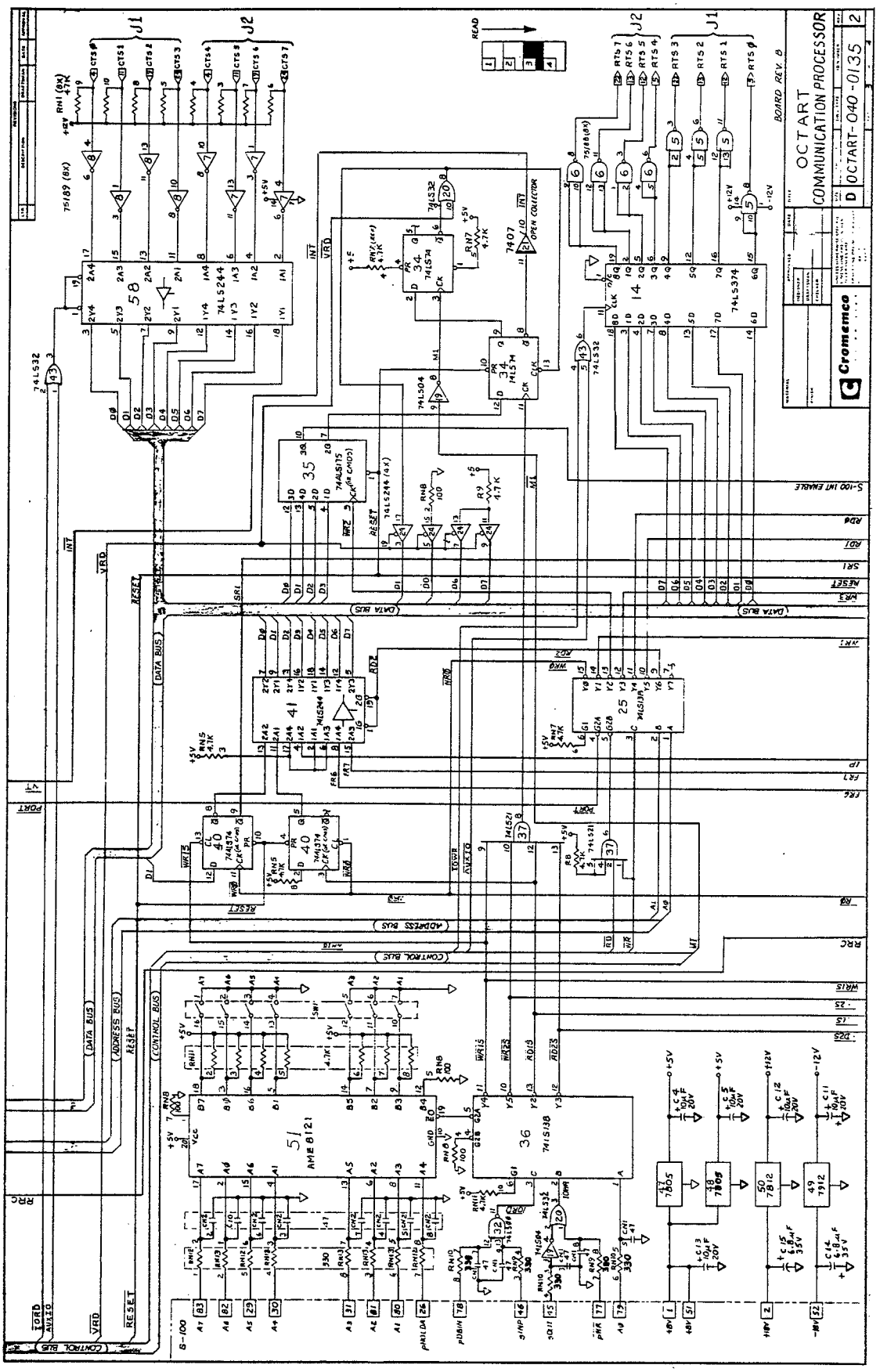
CROMEMCO Z80 Macro Assembler version 03.08 Feb 21, 1984 15:28:53 Page 0065
*** THE CROMEMCO OCTART MONITOR ***

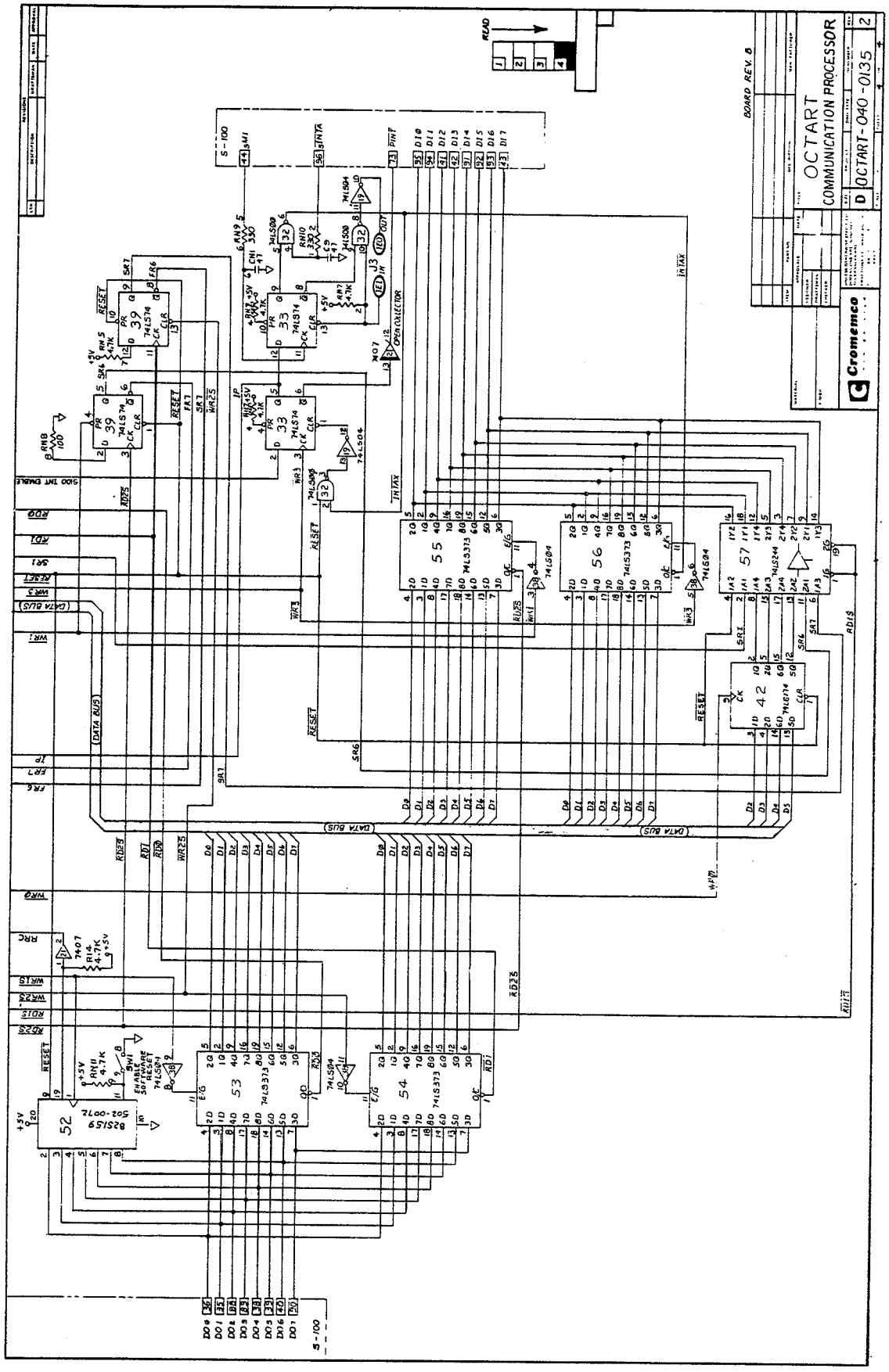
Symbol	Value	Defn	References
Romerr	0010	2319	
Rst.08	0008	0118	
Rst.10	0010	0149	
Rst.18	0018	0173	
Rst.20	0020	0198	
Rst.28	0028	0222	
Rst.30	0030	0246	
Scan	049B	1283	1080 1332 1483 1515
Scan.05	04A2	1291	1301 1303
Scan.10	04A3	1299	0473 0487 0779 1288 1430 1535 1610
Scan.15	04AC	1304	1081 1537 1542 1612
Scan.20	04C5	1315	1308
Scan.25	04C7	1317	1290 1310
Select	05DB	1639	2195
SetMem	0267	0763	2154
SM.20	026D	0772	0252 0797 0805
SM.40	0294	0792	0780
SM.50	0297	0795	0788
SM.60	029E	0799	0791
SM.70	02A5	0804	0785 0793
SmPtr	BF05	2245	0376 0770 0804
Space	06FD	1975	0909 1905 1963 2014
Stack	BFDF	2260	0275 0363 0373 0458
StartUp	0048	0283	0099
STAT_1A	0051	2336	0567
STAT_1B	0059	2337	0569
STAT_2A	0011	2338	0572
STAT_2B	0019	2339	0574
STAT_3A	0021	2340	0577
STAT_3B	0029	2341	0579
STAT_4A	0031	2342	0582
STAT_4B	0039	2343	0584
SU.20	004C	0299	0310
su.25	007B	0363	0314
SU.30	00BA	0395	0397
SU.40	00C5	0415	0435
SU.50	00D2	0432	0418
su.60	00D9	0441	0354 0405 0426
SUB	001A	2312	1744 2041
Tab	0009	2313	1302 2012
TempID	BF0A	2249	1166 1183
THRA	0053	2330	0642
TopRam	BFFF	0067	2234 2259
True	FFFF	0031	0041 1663
UCase	05D2	1627	1306 1563
User1	BFE1	2262	0154 0382
User2	BFE3	2263	0178 0383
User3	BFE5	2264	0203 0384
User4	BFE7	2265	0227 0385
UserJsys	BFDF	2261	0129 0365
UserNMI	BFE9	2266	0341 0386
UserRam	BFED	2269	0103
VarTbl	BEFF	2241	0372 0373
Verify	024A	0734	2157

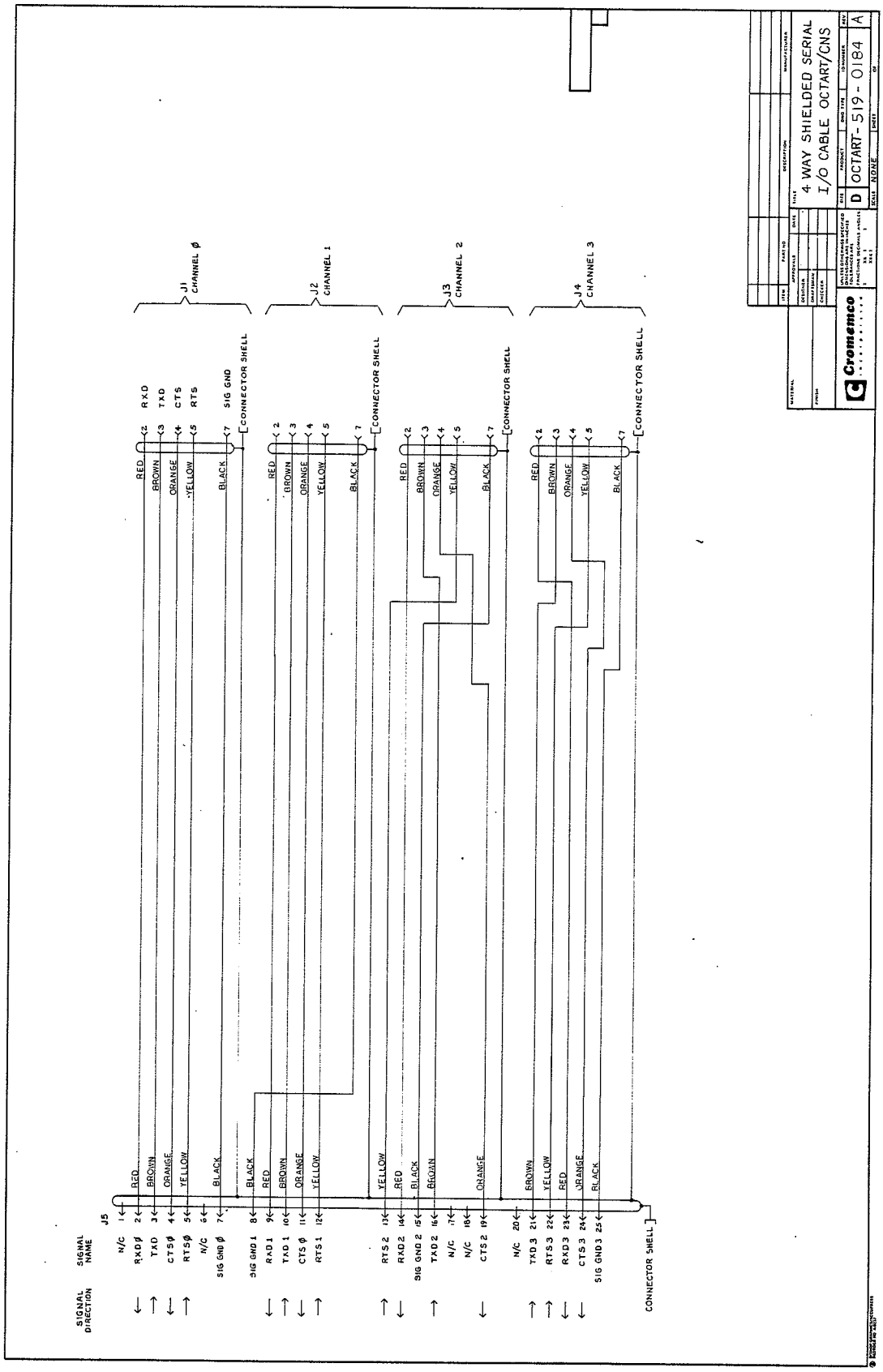
```
CROMEMCO Z80 Macro Assembler version 03.08          Feb 21. 1984  15:28:53          Page 0066
*** THE CROMEMCO OCTART MONITOR ***
Symbol      Value  Defn  References

VM.10       024D   0744  0759
VM.20       0261   0756  0746
WB.10       0767   2059  2061
wb.20       0776   2069  2065
wl          0697   1839  1842
WL.10       06B3   1876  1881
WL.20       06BD   1882  1879 1904
WrByte      0762   2050  1216 2178
WrLine      06B2   1870  2183
WrMsg       06BF   1886  2184
WrSeq       077B   2075  2160
WrSeq.1     077E   2084  1154 2088
Zap         0353   0977  2163
Zap.05      035B   0991  0374
Zap.10      035D   0992  0990
Zap.20      0362   0997  0730
```


OCTART
COMMUNICATION PROCESSOR
OCTART-040-0135
BOARD REV. B
Cromemco
Z 80 CPU
CPU & MEMORY SUPPORT (REV. 2.0)
2716 ROM
502-0087
16Kx8 RAM
12,13,28,29,45,46,59,60
74LS157
74LS244
74LS04
74LS74
74LS04
74LS00
74LS04
74LS08


OCTART
COMMUNICATION PROCESSOR
OCTART-020-0135
BOARD REV. B
Cromemco
J1
J2
SC2681
74LS367
75188
75189
74LS32
74LS367
ADDRESS BUS
DATA BUS
CONT BUS
ADDR BUS
CONT BUS
RESET
INT
RRC
IORD
IOWR
VRD
4MHZ
UART
DISK
DMA
PORT
AUX IO
CRT
DATA BUS
ADDRESS BUS
CONT BUS


OCTART
COMMUNICATION PROCESSOR
OCTART-040-0135
Cromemco
BOARD REV. B
74LS244
74LS374
74LS32
74LS04
74LS175
74LS244
74LS374
74LS21
74LS138
AME 8121
74LS32
74LS74
74LS74
74LS244
74LS244
74LS374
74LS374
74LS21
74LS138
7407
OPEN COLLECTOR
DATA BUS
ADDRESS BUS
CONTROL BUS
S-100 INT ENABLE
S-100


OCTART
COMMUNICATION PROCESSOR
OCTART-040-0135
Cromemco
BOARD REV. B
READ
S-100
INTAX
DATA BUS
RESET
OPEN COLLECTOR
ENABLE SOFTWARE RESET
74LS373
74LS74
74LS04
74LS244
74LS174
82S159
502-0072


4 WAY SHIELDED SERIAL I/O CABLE OCTART/CNS
OCTART-519-0184
Cromemco
J1 CHANNEL Ø
J2 CHANNEL 1
J3 CHANNEL 2
J4 CHANNEL 3
J5
SIGNAL DIRECTION
SIGNAL NAME
CONNECTOR SHELL
RXD Ø
TXD
CTS Ø
RTS Ø
SIG GND Ø
RXD 1
TXD 1
CTS 1
RTS 1
SIG GND 1
RXD 2
TXD 2
CTS 2
RTS 2
SIG GND 2
RXD 3
TXD 3
CTS 3
RTS 3
SIG GND 3
N/C
RED
BROWN
ORANGE
YELLOW
BLACK

*, Notation, 2

Access time, ROM, 4
ASCII character code, 53

Base address, Octart, 4
Bbase, defined, 4

Cables, Octart, 6
Command from host available, bit, 16, 25
Command from host empty, bit, 16, 21
Commands from host, register, 16, 19
Configure memory, register, 29
Connector J1, J2, Octart, 8
Connector J1, Octart, 13
Connector J3, Octart, 9
Connectors, Octart, 6

Data from host available, bit, 17, 24
Data from host empty, bit, 17, 20
Data from host, register, 17, 22
Data to host available, bit, 17, 20
Data to host empty, bit, 17, 24
Data to host, register, 17, 23

EBCDIC character code, 55
Enable host interrupts, bit, 15, 26
Enable Octart interrupts, bit, 15, 26

Host interrupt pending, bit, 24
Host interrupts, 15
Host/Octart, communications ports, 5, 16

In 1xh, port, 31
In 50h, port, 19
In 51h, port, 22
In 52h, port, 24
In bbase+00h, port, 20
In bbase+01h, port, 23
Installation, Octart board, 3
Internal Octart interrupts, 15
Interrupt modes, Z80a, 27
Interrupt priority wiring, S-100, 9
Interrupt vector to host, register, 15, 27
Interrupt vector, fah, fch, 26
Interrupts, host, 15

Interrupts, internal Octart, 15

Jumper, ROM speed, 4

Memory configuration, Octart, 3, 14, 29

Octart control, register, 26
Octart flags, register, 17, 24
Octart reset, bit, 21
Out 1xh, port, 32
Out 4xh, port, 29
Out 50h, port, 20
Out 51h, port, 23
Out 52h, port, 26
Out 53h, port, 27
Out bbase+00h, port, 19
Out bbase+01h, port, 22

Parallel input, register, 31
Parallel output, register, 32
Pin-outs, Octart, 7
Positive logic, definition, 2
Priority in, out, Octart, 9
Program store, Octart, 11
Programming information, Octart, 11

RAM, Octart memory, 11
RAM/ROM*, bit, 29
Reference 1, 2, 2
Register summary, Octart, 12
Reset state, Octart registers, 14
Reset, definition, 2
Reset, Octart, 9, 13
Reset, S-100 bus, 13
Reset, software, Octart, 13
Reset, Z80A, 14
ROM, Octart memory, 3
RS-232C levels, Octart, 8

Set, definition, 2
Setup, Octart, 3
Six-bit transcode, 51
Status bits, 20
Status from host empty, bits, 25
Status to host empty, bit, 17
Status to host, register, 17, 20
Switch SW-1, section 8, 13

LIMITED WARRANTY

Cromemco, Inc. ("Cromemco") warrants this equipment against defects in material and workmanship to the original purchaser for ninety (90) days from the date of purchase, subject to the following terms and conditions.

What Is Covered By This Warranty:

During the ninety (90) day warranty period Cromemco will, at its option, repair or replace this Cromemco product or repair or replace with new or used parts any parts or components, manufactured by Cromemco, which prove to be defective, provided the product is returned to your Dealer as set forth below.

How To Obtain Warranty Service:

You should immediately notify IN WRITING your Dealer or Cromemco of problems encountered during the warranty period in order to obtain warranty service; first obtain a return authorization number by contacting the Dealer from whom you purchased the product. Then attach to the product:

1. Your name, address, and telephone number,
2. the return authorization number
3. a description of the problem, and
4. proof of the date of retail purchase

Ship or otherwise return the product, transportation and insurance costs prepaid, to your Dealer. If you are unable to receive warranty repair from the Dealer from whom you purchased the product, you should contact Cromemco Customer Support at: Cromemco, Inc., 280 Bernardo Ave., Mountain View, Ca. 94043.

What Is Not Covered By This Warranty:

Cromemco does not warrant any products, components, or parts not manufactured by Cromemco.

This warranty does not apply if the product has been damaged by accident, abuse, misuse, modification, or misapplication; by damage during shipment; or by improper service. This product is not warranted to operate satisfactorily with peripherals or products not manufactured by Cromemco. Transportation and insurance charges incurred in transporting the product to and from your Dealer or Cromemco are not covered by this warranty.

Exclusion of Liability, Damages, and Other Warranties:

THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED. ANY IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE EXCLUDED WHERE SUCH EXCLUSION IS ALLOWED AND OTHERWISE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF PURCHASE OF THIS PRODUCT. IF THIS PRODUCT IS NOT IN GOOD WORKING ORDER AS WARRANTED ABOVE, YOUR SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. CROMEMCO SHALL NOT BE LIABLE FOR INCIDENTAL AND/OR CONSEQUENTIAL DAMAGES FOR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING DAMAGE TO PROPERTY AND, TO THE EXTENT PERMITTED BY LAW, DAMAGES FOR PERSONAL INJURY, EVEN IF CROMEMCO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE AGENTS, DEALERS, AND EMPLOYEES OF CROMEMCO ARE NOT AUTHORIZED TO MAKE MODIFICATIONS TO THIS WARRANTY, OR ADDITIONAL WARRANTIES BINDING ON CROMEMCO ABOUT OR FOR PRODUCTS SOLD OR LICENSED BY CROMEMCO. ACCORDINGLY, ADDITIONAL STATEMENTS WHETHER ORAL OR WRITTEN EXCEPT SIGNED WRITTEN STATEMENTS FROM AN OFFICER OF CROMEMCO DO NOT CONSTITUTE WARRANTIES AND SHOULD NOT BE RELIED UPON. SOFTWARE, TECHNICAL INFORMATION, AND FIRMWARE ARE LICENSED ONLY BY A SEPARATE AGREEMENT ON AN "AS IS" BASIS.

Limitation on Statute of Limitation and Transferability:

THIS WARRANTY AND THE STATUTE OF LIMITATIONS SHALL RUN CONCURRENTLY WITH ANY ACCEPTANCE PERIOD. THIS WARRANTY IS NOT TRANSFERABLE. NO SUIT, LITIGATION, OR ACTION SHALL BE BROUGHT BASED ON THE ALLEGED BREACH OF THIS WARRANTY OR IMPLIED WARRANTIES OR FOR OTHER CAUSE MORE THAN ONE YEAR AFTER THE DATE OF PURCHASE IN THOSE JURISDICTIONS ALLOWING SUCH A LIMITATION; OTHERWISE NO SUCH ACTION SHALL BE BROUGHT MORE THAN ONE YEAR AFTER THE EXPIRATION OF THIS WARRANTY.

Other Important Provisions:

Some states do not allow the exclusion or limitation of incidental or consequential damages or limitations on how long an implied warranty lasts, so the above limitation or exclusion may not apply to you. This warranty shall not be applicable to the extent that any provision of this warranty is prohibited by any federal, state, or municipal law which cannot be preempted. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Rev. D 6/14/84