TF30194 MICROWAVE ATE INTERFACE

Operator, programming and service manual

For Software revision level 1.2

Written by Chris Marshall, September 11th, 1987

TF-30194 MICROWAVE ATE INTERFACE

Operator and service manual
Table of Contents

TF-30194 MICROWAVE ATE INTERFACE

Operation And Programming
System Description

The TF-30194 is a GPIB/RS-232 controlled remote microwave switching interface. It was designed specifically with a second-generation ATE system in mind, but it's generic command structure, and versatile interfacing capability may open up other uses. It is designed to provide switching from as many as 7 input generators to two RF outputs. It also provides auxiliary switching that can be used for spectrum analyzers. The generator inputs can go from DC to 18 GHz, depending which paths you use. The interface can also switch variable attenuation into the VHF path of the primary generator, and 20 dB of fixed attenuation into the microwave path of the primary generator. It has a built-in pulse modulator good to 1 GHz, and a combiner good from DC to 18 GHz. It has a built-in noise source good from 10 MHz to 18 GHz. The remote programming is simple and straightforward.

TF-30194 MICROWAVE ATE INTERFACE

Operation And Programming
Hardware Description

The hardware for TF-30194 has been designed in a manner very similar to the software. It is a modular package (This may be due to the fact that the hardware was designed by the same person who designed the software). If you look to the hardware references in the rear of this manual, you will find block diagrams and schematics. You will also find pinouts for the front and rear panel connectors, as well as layouts of the RF inputs and outputs.

A couple of things were done in the prototype that are unusual. First of all, the three microwave instrument inputs are in an inset on the left side of the box (facing the front from in front of it). This is because the system that it has been designed for is meant to have those three instruments as roll-up instruments. The side connections allow access to the box without having to get into the back of a system rack. Also, you will find that the bezels are customized for the system it was designed for.

The fixture basically consists of three sections: The power supplies, the microprocessor/control section, and the RF switching/processing section.

It has three power supplies: A +28 Volt supply for the RF relays and the front panel LED indicators, a +5 Volt supply for the digital control and a +-12 Volt supply for the RS-232 communications of the digital controller.

The microprocessor/control section consists of an STD card cage, and four digital boards: A Ziatech ZT7805 GPIB computer and three PRO-LOG 7501 DC control cards.

The digital control is distributed through the A3 wiring distribution board. Three 40-conductor ribbon cables are run from the 7501's to the board, along with the +28V supply and the +5V supply. From this board, the control is separated into 37 lines going to the RF relay section, six lines going to the rear panel external control port and four lines going to the front panel LED indicators.

The 37 control/power lines going to the RF Relay section are passed through the bulkhead separator through a DBM-37 filtered connector that has been fashioned into a filtered male/female adapter.

The RF Relay section is completely isolated from the microprocessor/power section with a bulkhead. As mentioned, all of the lines going through the bulkhead are filtered by the 37-pin connector. This insures maximum noise immunity. All of the internal RF relays, mixers, combiners and attenuators are mounted on a single plate. The noise diode is mounted on a smaller plate raised above the main relay plate in order to insure minimum path distance and ease of removal.

All of the internal RF connections are made with semi-rigid tubing. All RF components are shielded. The noise diode's DC switching is handled by a small relay plate immediately behind it.

The box has two main outputs: RF Out 1 and RF Out 2. They are equal in switching capabilities, and either one can be connected to the various paths.

The box also has a couple of auxiliary spectrum analyzer outputs: SP1, on the rear panel, and SP2, on the side inset. These can be connected to the RF outputs, and to three auxiliary RF inputs. There is also provision to route one of them straight through the box to be used elsewhere in a system.

There are 7 main RF inputs, and 5 auxiliary RF inputs. Two of the main inputs, and all of the auxiliary inputs are good from DC to 18 GHz. Two more of the main inputs are rated for VHF frequencies. The other three can be used up through VHF frequencies, but two of them were designed for LF/HF generators.

The VHF path of the primary generator path (GN1) can have a variable attenuation added to it. The attenuator is rated to 1.3 GHz.

The microwave path of the primary generator path can have a 20 dB pad added to it.

The VHF path of the secondary generator path can be pulse modulated through a built-in pulse modulator rated up to 1 GHz. The LF generator of the secondary path is automatically switched into the modulator as the modulating source. The modulator provides over 80 dB of on/off modulation, switched as fast as 20 nS.

Both generator paths can be run through a combiner good from DC to 18 GHz.

Both RF outputs can be connected to a noise source good from 10 MHz to 18 GHz.

The two RF outputs can be simultaneously connected to a combination of paths. There are too many possibilities to go over here. Some of the paths are mutually exclusive, but many are not.

For a more complete view of the switching capabilities of this unit, it is suggested that you consult the General Signal Routing Diagram in the rear of this manual, and read the programmer's guide in the following section.

TF-30194 MICROWAVE ATE INTERFACE

Operation And Programming
Programmer's Guide

The interface has two operating modes: Command, where paths are set, and system parameters are changed or set, and "Pass-through", where it acts as an GPIB/RS-232 translator. The pass through mode is terminated when either interface sends an ASCII 127 (rubout).

The interface can be controlled at any time by either of it's communication interfaces. This allows tremendous flexibility when installed in a system. It even has the ability to simultaneously be accessed by both interfaces, and can output to either one, or both simultaneously.

The serial port's parameters can be changed by software commands, including the character considered to be a serial message termination by the box. (Default is a line-feed, or ASCII 10)

The GPIB's address is set by the switches on the ZT-7805 board, but commands are provided to change it by software. It's messages are terminated by line-feeds in command mode, and/or EOI in both modes.

There are four possible SRQ's on the GPIB buss: Input errors in command mode, request to output in command mode, pass-through mode selected, and request to output in pass-through mode. The default mask sets SRQ's for input errors only, but the operator may select any mask he/she/it wishes.

The path control is done through 3 PRO-LOG™ 7501 cards. They are medium-power active low drivers. They are separated into six 8-line ports. (40-45)

Ports 40-43 are direct connections to the relays, and to 5 external control lines accessible on the back panel of the interface through a DBM-15 connector. Port 44 is dedicated solely to control of the programmable attenuator in the VHF path of the GN1 path. Port 45 is used to control 4 indicator lamps on the front panel of the interface that indicate communications, errors and SRQ's.

The port control is accomplished through the use of a six-byte bit map that is updated by the path commands, and then sent to the ports. Two commands can return the status of this bit map to the operator; "RPS" (Return path setup), and "RBM" (Return bit map in binary form).

The commands are matched in lookup tables, and use a tabular system to set/clear bits in the bit map. System commands (Ones that do not affect the paths) are handled on an interpretive basis. Commands to set the attenuator are also interpreted as opposed to being tabularized.

Adding new commands, especially path commands, or altering existing ones is quite simple, involving only three tables for path commands, and one table for system commands. System commands must have the interpretive routines written.

Numerical input is allowed from 0 to 255. Anything other than this is considered an error. Negative signs are ignored, as are leading 0's. The number is read from the first numerical character encountered in the command (0-9), to the first non-numerical character encountered after that. Numerical output is from 0 to 255, with no signs, and no leading 0's.

In the command mode, lowercase letters are considered the same as uppercase letters, and spaces are ignored. System commands do not have to have equal signs (=) preceding the numbers, but it usually helps for clarity. Path commands are literally interpreted, and must follow the formats outlined in this manual. All commands except for "CLR" and "CLC" can be concatenated by commas or semicolons. They will be executed in the order received. If an error is encountered in a concatenated string of commands, the commands up to that error are executed, but the execution is aborted for the command in error and any commands after it.

The maximum length of input from either interface is 128 characters, including termination characters. If the length is exceeded, the first characters received start getting written over by new characters. This applies to the pass-through mode too, where data is sent in "packets" delimited by the termination conditions specified for each interface.

If an output to the GPIB bus from the interface is not entered by the computer before another one is requested, the new information is appended to the old reply. Commands that generate replies terminate command streams, as do errors received in command streams.

Command	Description
BOT	Designate both interfaces as output ports.
CLC	Clear communication ports: Resets and clears 488/232 interfaces, and "cleans up" the buffers.
CLP	Set paths to default state. (Path clear)
CLR	System Clear: Clears all paths to default state, clears buffers and communication ports, and sets default system conditions. (Same as power-up initialization)
IEO	Designate the IEEE-488 interface as the output communication port.
PTM	Set "Pass-through" mode.
RAD	Return the IEEE-488 address.
RBM	Return the bit map: Tells the box to directly, without translation, output the contents of the bit map on the designated output port(s).
RPS	Return the path setup: Tells the box to return the translated path strings to the designated output ports(s).
RSB	Return the status byte: Tells the box to read the status byte, and to return it on the communication port(s) designated as output port(s).
RSO	Designate the RS-232 interface as the output communication port.

Command	Description
ADR = (XX)	Set the IEEE-488 interface to the address given by the decimal number XX (0-31).
ATT = (XXX)	Set the attenuator to XXX dB. (0-127)
BDR = (XX)	Set the baud rate according to the decimal number XX. (0-17)
SER = (XXX)	Set the serial mask according to the decimal number XXX.
STM = (XXX)	Designate the serial termination character as that whose ASCII (or higher) number is given by XXX. (0-255)
STS = (XXX)	Set the SRQ mask according to the decimal number XXX.

Operation And Programming
Path Command Definitions

Command	Description
GN1=RO1	Connect the primary generator path to RF Output 1.
GN2=RO1	Connect the secondary generator path to RF Output 1.
RI1=RO1	Connect RF Input 1 to RF Output 1.
NOS=RO1	Connect the noise diode to RF Output 1.
SPA=RO1	Connect the spectrum analyzer path to RF Output 1.
GN1=RO2	Connect the primary generator path to RF Output 2.
GN2=RO2	Connect the secondary generator path to RF Output 2.
RI2=RO2	Connect RF Input 2 to RF Output 2.
NOS=RO2	Connect the noise diode to RF Output 2.
SPA=RO2	Connect the spectrum analyzer path to RF Output 2.
SPA=RI3	Connect the spectrum analyzer path to RF Input 3.
SPA=RI4	Connect the spectrum analyzer path to RF Input 4.
SPA=RI5	Connect the spectrum analyzer to RF Input 5.
TR1	Terminate RF Output 1 in 50 Ohms.
TR2	Terminate RF Output 2 in 50 Ohms.
EF1	Connect the primary generator path to the primary microwave generator input.
VF1	Connect the primary generator path to the primary VHF generator input.
LF1	Connect the primary generator path to the primary low frequency generator input.

Command	Description
EF2	Connect the secondary generator path to the secondary microwave generator input.
VF2	Connect the secondary generator path to the secondary VHF generator input.
LF2	Connect the secondary generator path to the secondary low frequency generator.
AIN	Connect the primary generator path to the auxiliary input. (VCO)
SP1	Connect the spectrum analyzer path to the VHF spectrum analyzer.
SP2	Connect the spectrum analyzer path to the microwave spectrum analyzer.
CBN	Connect the selected primary and secondary generators to the combiner, and provide the output on the primary generator path.
CBF	Switch out the combiner, restoring the previous primary and/or secondary generator path connections.
SIF	Connect the VHF spectrum analyzer to the auxiliary rear panel input. (IF IN)
MAN	Switch a 20 dB pad into the primary microwave generator's path.
MAF	Switch out the 20 dB pad, restoring straight-through connection.
AOT	Connect the output of the secondary generator path to the auxiliary rear panel output. (AVR LO)
AOF	Switch out the auxiliary output, restoring previous secondary generator path.
NON	Turn on the the noise diode.
NOF	Turn off the noise diode.

Command	Description
PMN	Switch the special pulse modulator into the secondary generator path.
PMF	Switch out the special pulse modulator.
LX1=1	Select external control line 1. (Active low)
LX1=0	Deselect external control line 1. (Floating)
LX2=1	Select external control line 2.
LX2=0	Deselect external control line 2.
LX3=1	Select external control line 3.
LX3=0	Deselect external control line 3.
LX4=1	Select external control line 4.
LX4=0	Deselect external control line 4.
LX5=1	Select external control line 5.
LX5=0	Deselect external control line 5.

Operation And Programming
Setting the serial parameters

The serial port can be set up to user specifications through the use of 3 system commands: "SER", "BDR" and "STM". "SER" sets up parity, stop bits, and data bits, while "BDR" sets the baud rate. "STM" sets the character that the system should consider a termination character.

"STM": This can be any byte from 0 to 255. The default is 10 (Line Feed).

"BDR": This can be any number from 0 to 17. The set is as follows:

0 ..... 50 Baud
1 75
2 110
3 ..... 134.5
4 150
5 300
6 600
7 1,200
8 1,800
9 2,000
10 2,400
11 3,600
12 4,800
13 7,200
14 9,600
15 19,200
16 38,400
17 56,000

"SER": This command is followed by a decimal number determined as shown below:

Bits 0 & 1 ..... Word length: 00 is 5 bits, 01 is 6 bits, 10 is 7 bits, 11 is 8 bits.

Bit 2 ..... Number of stop bits: 0 is 1 bit, 1 is 1.5 or 2 stop bits.

Bit 3 ..... Parity enable: 0 is no parity (ignore bits 4 & 5), 1 is parity as determined by bits 4 & 5.

Bit 4 & 5 ..... Parity bits: 00 is odd, 01 is even, 10 is always 1, 11 is always 0.

Bits 6 & 7 ..... Not used

The default serial setup is: 9600 Baud, 8 data bits, 1 stop bit and no parity. This is set whenever the instrument is powered up, and when it receives a "CLR" or "CLC" command.

Operation And Programming
The GPIB Status Byte

The status byte will contain flags indicating various conditions within the operating system of the box. It will be maskable by the SRQ mask byte, indicating which of the conditions will be grounds for a 488 SRQ.

The format of both the status and mask bytes will be:

X A X X P R S E (LSB)

Where: A = The SRQ flag; Indicates that service was requested.
P = The unit is now in pass-through mode.
R = There is a command reply ready.
S = There is pass-through data available from the serial port.
E = There has been an error.

The default gives an SRQ for errors only (The mask is a decimal 1).

Operation And Programming
Replies From the Interface in Command Mode

The interface has four commands in the Command mode that generate replies: "RSB", "RBM", "RAD" and "RPS". The reply is generated on the designated output buss(es). The output buss is designated using the "IEO", "RSO" and "BOT" commands as explained in the command definitions. The default is IEEE-488 only.

When an output is generated on the GPIB buss, the GPIB status byte has it's # 2 bit set, indicating that a system command reply is available. If the mask set by the "STS" command allows it, an SRQ is generated.

"RSB" generates a single number which is a decimal translation of the GPIB status byte. The message is terminated by EOI and a carriage return/line feed.

"RBM" sends the contents of the bit map directly across the buss in binary format (six bytes). The message is terminated by EOI set with the sixth byte.

"RAD" returns a single number corresponding to the present IEEE-488 address. (0-31)

"RPS" sends five lines, each delimited by a carriage return/line feed, but EOI is not set until the last line feed has been sent. Below is an example of the default return:

LINES: 20,24,27, $*^{C}_{R}$
$^{L}_{F}$ATT = 0^{C}_{R}
$^{L}_{F}$EXT: $*^{C}_{R}$
$^{L}_{F}$STM = 10^{C}_{R}
$^{L}_{F}$ADR = 2^{C}_{R}
$^{L}_{F}$ <-{EOI is set with this byte}

The first line contains the numbers of all the relay control lines that have been selected, or are active. The second line returns the value that the attenuator is set to. The third line returns which of the external control lines, if any, are active. The fourth line returns the serial termination character's number. The fifth line gives the present GPIB address.

The GPIB interface is almost completely "Timeout proof". The only possible way that it should time out is if the box receives a "CLR" or "CLC" from the RS-232 interface while it is simultaneously communicating with the GPIB buss. As these reset the buss, you will probably get a timeout. Other than that, you should always be able to enter data on the GPIB buss. If no output is available, a line feed with EOI set is sent.

One note: The box expects intelligible data. If it is not in the pass-through mode, all data sent must be a proper command as specified previously. Anything else is considered an error, and is treated as such (control characters, spaces and the serial termination character sent in the command mode are ignored and are not considered errors). In the pass-through mode, everything except the pass-through terminator is sent straight through.

Operation And Programming
The "Pass-through" Mode

The Pass-through mode is a mode that the interface can be put into wherein it acts as a GPIB/RS-232 translator. All input from one buss is sent unaltered to the opposite buss and output. It is initiated by sending the "PTM" command, and exited when either interface sends an ASCII 127 (rubout).

As mentioned before, the data is received and sent in "packets" not to exceed 128 bytes in length, including termination characters. The GPIB buss must terminate it's packets with EOI set with the last byte, and the serial interface uses the designated serial termination character, as set by the "STM" command. The pass-through terminator is not considered a terminator. If it is received, the pass-through mode is exited immediately, and the buffer is cleared. If the packet is over 128 bytes long, the first bytes received will be overwritten by subsequent bytes received.

In Pass-through mode, outputs are generated on the buss opposite that which the input was received on. If the output is on the GPIB buss, and the "STS" command allows it, an SRQ is generated. The # 2 bit of the GPIB status byte is set to indicate that a pass-through reply is available. All ASCII characters from 0 to 126 are passed through as is.

The serial interface is set up according to the commands "SER", "BDR" and "STM". Otherwise the default settings are in effect.

TF-30194 MICROWAVE ATE INTERFACE

Assembly Language Software
Technical references

In the following section, you will find some brief descriptions of the operating system of the TF-30194 Microwave Interface. It has been written in 8085 assembly language software to be implemented on the Huntsville Macro Assembler for Intel 8085. An assembler listing of revision 1.2 can be found in the rear of this manual. It is extremely readable, and should provide you with a detailed view of it's operation. The explanations in this section are mainly to clarify some of the more obscure or complicated issues in the system.

The processor is a Ziatech ZT 7805 GPIB computer.

The assembly language operating system for the TF-30194 Microwave ATE interface box is an interrupt-driven I/O-intensive control system.

The system consists basically of 6 parts:

1).....The initialization routines.
2).....The GPIB (IEEE-488) I/O routine.
3).....The RS-232 I/O routine.
4).....The pass-thru handlers.
5).....The system command handlers.
6).....The path control command handlers.

Assembly Language Software
Using the Buffers

There are five 128-byte buffers. Two are for IEEE-488 I/O, two are for RS-232 I/O, and one is a "character buffer" that holds strings that are being processed by central processing routines. The buffers are all global. These buffers should be considered data storage only; They should not be used for operating space.

Each buffer has a control block assigned to it. Each control block consists of three bytes: a fill pointer, an empty pointer and a byte count. The pointers and the byte count are initialized to 0. The fill pointer is incremented each time a byte is entered into the buffer, and the empty pointer is incremented each time a byte is read from the buffer. The byte count is incremented when a byte is entered, and decremented when a byte is removed. It is not a pointer, but a numeric count. It should not go negative, nor should it exceed 128. The pointers are indexes from the base address of the buffer.

The buffers are "circular". i.e. the pointers will go back to the beginning of the buffer when the end of the buffer is encountered.

In the command mode, the input strings are "cleaned up" by a processing routine, and then compared, byte by byte, to a table of stored comparison strings. If the string matches, a command number is returned, and the appropriate command is executed. If no match occurs, an error is flagged, and execution is halted.

The string matches are stored in a packed string format as shown:

First byte..... This will be the length indicator (1-255)
Second byte..... The first character's ASCII number.
Following bytes (If any)..... The rest of the string, in ASCII format.

Example: For the string "GN1=RO1".

Byte	Hex Value	Character
1	$07	Length
2	$47	"G"
3	$4E	"N"
4	$31	"1"
5	$3D	"="
6	$52	"R"
7	$4F	"O"
8	$31	"1"

TF-30194 MICROWAVE ATE INTERFACE

Assembly Language Software
Bit Map Definition

The bit map provides a software "mirror" of the six control ports provided by the 7501 DC driver cards. Byte 0 corresponds to port 40, byte 1 corresponds to port 41, etc...

The bit map will be set up as follows: The bit number is the number that is used to determine it's location in the map, and the number below it is the line number controlled by that bit.

	MSB							LSB
Bit #:	7	6	5	4	3	2	1	0
Byte 0	8	7	6	5	4	3	2	1
	Relay Control lines							
Bit #:	15	14	13	12	11	10	9	8
Byte 1	16	15	14	13	12	11	10	9
	Relay Control lines							
Bit #:	23	22	21	20	19	18	17	16
Byte 2	24	23	22	21	20	19	18	17
	Relay Control lines							
Bit #:	31	30	29	28	27	26	25	24
Byte 3	39	38	37	36	35	27	26	25
	External Control lines					Relay Control lines		
Bit #:	39	38	37	36	35	34	33	32
Byte 4	N/U	34	33	32	31	30	29	28
	Attenuator Control							
Bit #:	47	46	45	44	43	42	41	40
Byte 5	N/U	N/U	N/U	N/U	43	42	41	40
	Front Panel Lamp Control lines							

When a path is selected, the program brings in and decodes a set of control codes. These codes determine which bit(s) will be set or cleared. Each code stands for one bit of the bit map. Each code consists of one byte that is decoded as follows:

A C B_5 B_4 B_3 B_2 B_1 B_0 LSB

A indicates whether the bit should be set, or cleared. If it is 1, then the bit is set. If the bit is set, it indicates that the line should be an active LOW, except for the attenuator byte, which is complemented.

B is the number of the bit in the bit map: B_0-B_2 is the number of the bit, (from 0-7), and B_3-B_4 is the number of the byte. (from 0-5)

C is set if this is the last byte in the setup stream. It is cleared otherwise.

Assembly Language Software
Path Code Definitions

The following chart contains the bit map setting fields.

The Hex$ numbers are the actual numbers, in hexadecimal, that should be placed in the field to achieve the desired results on the corresponding line.

The Line numbers correspond to the actual hardware lines in the test fixture, and the bit numbers indicate the bit in the bit map that corresponds to that line.

The T/F line indicates whether or not the bit should be set on that line.

The last byte of each field has it's #6 bit set, to indicate that it is the last byte. See the section on the bit map for further information on the byte format.

TF-30194 MICROWAVE ATE INTERFACE

Assembly Language Software
Path Code Tables

Path description	Hex$	Lines	T/F	Bit #
GN1 = RO1	07	8	0	7
	08	9	0	8
	0F	16	0	15
	10	17	0	16
	11	18	0	17
	12	19	0	18
	D3	20	1	19
GN2 = RO1	87	8	1	7
	0F	16	0	15
	10	17	0	16
	11	18	0	17
	12	19	0	18
	53	20	0	19
RI1 = RO1	07	8	0	7
	0F	16	0	15
	10	17	0	16
	11	18	0	17
	92	19	1	18
	53	20	0	19
NOS = RO1	07	8	0	7
	0D	14	0	13
	8F	16	1	15
	10	17	0	16
	11	18	0	17
	12	19	10	17
	11	18	0	17
	12	19	0	18
	53	20	0	19
SPA = RO1	07	8	0	7
	0F	16	0	15
	10	17	0	16
	91	18	1	17
	12	19	0	18
	13	20	0	19
	15	22	0	21
	18	25	0	24
	19	26	0	25
	5A	27	0	26

Assembly Language Software
Path Code Tables (continued)

Path description	Hex$	Lines	T/F	Bit #
GN1 = RO2	88	9	1	8
	0D	14	0	13
	14	21	0	20
	15	22	0	21
	16	23	0	22
	57	24	0	23
GN2 = RO2	07	8	0	7
	08	9	0	8
	0A	11	0	10
	0D	14	0	13
	14	21	0	20
	15	22	0	21
	16	23	0	22
	D7	24	1	23
RI2 = GN2	08	9	0	8
	0D	14	0	13
	14	21	0	20
	15	22	0	21
	96	23	1	22
	57	24	0	23
NOS = RO2	08	9	0	8
	8D	14	1	13
	14	21	0	20
	15	22	0	21
	16	23	0	22
	57	24	0	23
SPA = RO2	08	9	0	8
	0D	14	0	13
	11	18	0	17
	14	21	0	20
	95	22	1	21
	16	23	0	22
	17	24	0	23
	18	25	0	24
	19	26	0	25
	5A	27	0	26
SPA = RI3	11	18	0	17
	15	22	0	21
	18	25	0	24
	99	26	1	25
	5A	27	0	26

Assembly Language Software
Path Code Tables (continued)

Path description	Hex$	Lines	T/F	Bit #
SPA = RI4	11	18	0	17
	15	22	0	21
	98	25	1	24
	19	26	0	25
	5A	27	0	26
SPA = RI5	11	18	0	17
	15	22	0	21
	18	25	0	24
	19	26	0	25
	DA	27	1	26
TR1	07	8	0	7
	0F	16	0	15
	90	17	1	16
	11	18	0	17
	12	19	0	18
	53	20	0	19
TR2	08	9	0	8
	0D	14	0	13
	94	21	1	20
	15	22	0	21
	16	23	0	22
	57	24	0	23
EF1	C3	4	1	3
VF1	01	2	0	1
	43	4	0	3
LF1	81	2	1	1
	02	3	0	2
	43	4	0	3
EF2	85	6	1	5
	46	7	0	6
VF2	04	5	0	4
	05	6	0	5
	46	7	0	6
LF2	84	5	1	4
	05	6	0	5
	46	7	0	6

Assembly Language Software
Path Code Tables (continued)

Path description	Hex$	Lines	T/F	Bit #
AIN	81	2	1	1
	82	3	1	2
	43	4	0	3
SP1	8B	12	1	11
	4C	13	0	12
SP2	CC	13	1	12
CBN	CA	11	1	10
CBF	4A	11	0	10
SIF	4B	12	0	11
MAN	C0	1	1	0
MAF	40	1	0	0
AOT	86	7	1	6
	4A	11	0	10
AOF	46	7	0	6
NON	CE	15	1	14
NOF	4E	15	0	14
PMN	C9	10	1	9
PMF	49	10	0	9

Assembly Language Software
Path Code Tables (continued)

Path description	Hex$	Lines	T/F	Bit #
LX1 = 1	DB	35	1	27
LX1 = 0	5B	35	0	27
LX2 = 1	DC	36	1	28
LX2 = 0	5C	36	0	28
LX3 = 1	DD	37	1	29
LX3 = 0	5D	37	0	29
LX4 = 1	DE	38	1	30
LX4 = 0	5E	38	0	30
LX5 = 1	DF	39	1	31
LX5 = 0	5F	39	0	31

TF-30194 MICROWAVE ATE INTERFACE

Assembly Language Listing of
Software Revision Level 1.2

```
Huntsville Macro Assembler  8085 cross assembler for PC-DOS 2.0  v1.82L  Page 1
                                                        Module:  tf30194d

                ;*****************************************************************************
                ;*    TF-30194 MICROWAVE ATE INTERFACE ASSEMBLY LANGUAGE OPERATING SYSTEM     *
                ;*****************************************************************************

                ;                       Version 1.2 as of September 10, 1987
                ;                            Written by Chris Marshall

                ;   This version has several changes from 1.11: The pass-thru terminator is
                ;   now a "passive" terminator. It will not terminate a command stream, and
                ;   will clear the buffer of all data at termination time. Also, straight
                ;   line-feeds that are input in command mode will no longer cause errors.
                ;   I have "souped up" the RS-232 receive routine in order to increase the
                ;   receive speed. This involved altering the way that the cleaning routine
                ;   works and I also had the communication lamp stay on until the entire
                ;   message was finished, as opposed to flashing on and off for each character.

                ;   The last version was 1.11.

                ;   The following system has been written in Intel 8085 assembly code to be
                ;   implemented on the HMA8085 Macro assembler.


                ; Interrupt Vectors:

0000            Start     EQU       00H       ; Reset/CLR vector
002C            Intr5     EQU       2CH       ; 5.5 (RS-232) interrupt vector
0034            Intr6     EQU       34H       ; 6.5 (IEEE-488) interrupt vector

0030            Bport     EQU       30H       ; Base address for communication ports

                ; TMS-9914A Equates (IEEE-488)

                ; Register locations:

0030            Int0      EQU       Bport+00H ; Interrupt register 0 location
0031            Int1      EQU       Bport+01H ; Interrupt register 1 location
0032            Addsta    EQU       Bport+02H ; Address status register location
0033            Auxcmd    EQU       Bport+03H ; Auxilliary command register location
0034            Addswt    EQU       Bport+04H ; Address switch register location
0034            Addreg    EQU       Bport+04H ; Address register location
0035            Spoll     EQU       Bport+05H ; Serial poll register location
0037            Datin     EQU       Bport+07H ; Data in register location
0037            Datout    EQU       Bport+07H ; Data out register location

                ; Bit masks:

0004            Spas      EQU       04H       ; Mask for Serial Poll Active bit
0080            Swrst     EQU       80H       ; Mask for software reset
0000            Clrst     EQU       00H       ; Mask for Clearing software registers
0008            EOIm      EQU       08H       ; Mask for EOI active bit
0010            Bo        EQU       10H       ; Mask for Byte Out ready bit
0020            Bi        EQU       20H       ; Mask for Byte In received bit
0002            Tads      EQU       02H       ; Mask for Talker addressed state
0004            Lads      EQU       04H       ; Mask for Listener addressed state
0008            Tpas      EQU       08H       ; Mask for Talker primary addressed state
```

```
0010            Lpas    EQU     10H     ; Mask for Listener primary addressed state
0020            Mla     EQU     20H     ; Mask for My Listen Address bit active
0040            Mta     EQU     40H     ; Mask for My Talk Address bit active
0001            Erstat  EQU     01H     ; Mask for Command error bit of status byte
0002            Ptstat  EQU     02H     ; Mask for Pass-throughH bit of status byte
0004            Srstat  EQU     04H     ; Mask for Serial data ready bit of status byte
0098            Rsv2    EQU     98H     ; Mask for asserting SRQ

                ; WD-8520 Equates (RS-232):

                ; Register locations:

0038            Recbuf  EQU     Bport+08H ; Receive buffer register location
0038            Dllsb   EQU     Bport+08H ; LSB of divisor latch location
0039            Dlmsb   EQU     Bport+09H ; MSB of divisor latch location
0038            Xmtbuf  EQU     Bport+08H ; Transmit buffer register location
0039            Int2    EQU     Bport+09H ; Serial interrupt register location
003A            Intrid  EQU     Bport+0AH ; Interrupt identification register location
003B            Linctl  EQU     Bport+0BH ; Line control register location
003C            Modctl  EQU     Bport+0CH ; Modem control register location
003D            Linsta  EQU     Bport+0DH ; Line status register location
003E            Modsta  EQU     Bport+0EH ; Modem status register location

                ; Bit Masks:

0001            Recda   EQU     01H     ; Mask for Received data availiable bit
0002            Tbei    EQU     02H     ; Mask for Transmitter holding register empty bit
0080            Dlab    EQU     80H     ; Mask for divisor latch enable
0001            Dtr     EQU     01H     ; Mask for Data Terminal Ready active bit
0002            Rts     EQU     02H     ; Mask for Request To Send active bit
0010            Cts     EQU     10H     ; Mask for Clear To Send line active bit
0020            Dsr     EQU     20H     ; Mask for Data Set Ready line active bit
0001            Drbit   EQU     01H     ; Mask for Data Ready
0020            Thre    EQU     20H     ; Mask for Transmitter holding register empty bit
0040            Tsre    EQU     40H     ; Mask for Transmitter shift register empty bit
0004            Rcv     EQU     04H     ; Mask for received data
0002            Txm     EQU     02H     ; Mask for transmit data

                ; I/O Port addresses:

0040            Port40  EQU     40H     ; Relay control lines 1-8
0041            Port41  EQU     41H     ; Relay control lines 9-16
0042            Port42  EQU     42H     ; Relay control lines 17-24
0043            Port43  EQU     43H     ; Relay control lines 25-27, and lines 35-39
0044            Port44  EQU     44H     ; Attenuator control lines 28-34
0045            Port45  EQU     45H     ; Front panel lamp control lines 40-43

                ; Default bit map conditions:

0000            Defbt0  EQU     00H     ; No lines active
0000            Defbt1  EQU     00H     ; No lines active
0088            Defbt2  EQU     88H     ; Lines active: #24, #20
0004            Defbt3  EQU     04H     ; Lines active: #27
0000            Defbt4  EQU     00H     ; No lines active
0000            Defbt5  EQU     00H     ; No lines active
```

```
                ; Front panel lamp masks:

0001            Hpibon  EQU     01H     ; IEEE-488 Front panel lamp on/off
0002            Serlon  EQU     02H     ; RS-232 Front panel lamp on/off
0004            Erron   EQU     04H     ; ERROR Front panel lamp on/off
0008            Srqon   EQU     08H     ; SRQ Front panel lamp on/off

                ; SRQ masks:

0001            Srmask  EQU     01H     ; Default SRQ mask (error only)
0001            Errr    EQU     01H     ; Input error
0002            Serl    EQU     02H     ; Serial "Pass through" data ready
0004            Replie  EQU     04H     ; Reply ready
0008            Pstr    EQU     08H     ; Pass through mode indicator

                ; System setups:

0004            Sysbyt  EQU     04H     ; System status byte default value
000B            Intwrd  EQU     0BH     ; Mask for 8 data bit, 1 stop bit and odd parity
0040            Pastru  EQU     40H     ; "Pass-through" mode mask
0004            Ieomsk  EQU     04H     ; IEEE-488 selected as an output port
0008            Rsomsk  EQU     08H     ; RS-232 selected as an output port
007F            Stoppt  EQU     7FH     ; Pass-thru termination character ("Rubout")
0020            Act232  EQU     20H     ; Indicates RS-232 input buffer is active
0010            Nocmnd  EQU     10H     ; Indicates that a command has not been matched
0001            Scrtch  EQU     01H     ; "Scratch" flag
0080            Ptproc  EQU     80h     ; Final pass-through process flag (Temporary)
0002            Lstone  EQU     02H     ; Used to indicate last byte sent to GPIB

0014            Bd9600  EQU     0014H   ; Mask for 9600 Baud divisor word

                ; RAM allocation:

3C00            Ramstr  EQU     3C00H   ; Bottom of RAM
3FFF            Stack   EQU     3FFFH   ; Top of RAM

                ; This is a dynamic allocation routine

3C00                    ORG     Ramstr  ; Start from the bottom, and work up

3C00            Iibfp   DS      01H     ; IEEE-488 input buffer fill pointer
3C01            Iibep   DS      01H     ; IEEE-488 input buffer MT pointer
3C02            Iibbc   DS      01H     ; IEEE-448 input buffer byte count
3C03            Iibase  DS      80H     ; IEEE-488 input buffer

3C83            Iobfp   DS      01H     ; IEEE-488 output buffer fill pointer
3C84            Iobep   DS      01H     ; IEEE-488 output buffer MT pointer
3C85            Iobbc   DS      01H     ; IEEE-488 output buffer byte count
3C86            Iobase  DS      80H     ; IEEE-488 output buffer

3D06            Ribfp   DS      01H     ; RS-232 input buffer fill pointer
3D07            Ribep   DS      01H     ; RS-232 input buffer MT pointer
3D08            Ribbc   DS      01H     ; RS-232 input buffer byte count
3D09            Ribase  DS      80H     ; RS-232 input buffer
```

```
3D89            Robfp     DS        01H       ; RS-232 output buffer fill pointer
3D8A            Robep     DS        01H       ; RS-232 output buffer MT pointer
3D8B            Robbc     DS        01H       ; RS-232 output buffer byte count
3D8C            Robase    DS        80H       ; RS-232 output buffer

3E0C            Chbfp     DS        01H       ; Character buffer fill pointer
3E0D            Chbep     DS        01H       ; Character buffer MT pointer
3E0E            Chbbc     DS        01H       ; Character buffer byte count
3E0F            Chbase    DS        80H       ; Character buffer

3E8F            Byte0     DS        01H       ; Byte 0 of the bit map
3E90            Byte1     DS        01H       ; Byte 1 of the bit map
3E91            Byte2     DS        01H       ; Byte 2 of the bit map
3E92            Byte3     DS        01H       ; Byte 3 of the bit map
3E93            Byte4     DS        01H       ; Byte 4 of the bit map
3E94            Byte5     DS        01H       ; Byte 5 of the bit map

3E95            Sertrm    DS        01H       ; Serial termination character
3E96            Wordse    DS        01H       ; Serial status byte
3E97            Status    DS        01H       ; GPIB status byte
3E98            Srqmsk    DS        01H       ; Current SRQ mask
3E99            Systat    DS        01H       ; System status byte
3E9A            Adgpib    DS        01H       ; GPIB address

                ;*****************************************************************************
                ;*                      INITIALIZATION/RESTART ROUTINE                       *
                ;*****************************************************************************

0000                      ORG       Start               ; Start at the beginning

0000 F3         Begin     DI                            ; Disable interrupts during initialize

0001 AF         Clrin     XRA       A                   ; "CLR" command enters here

0002 3D         Delays    DCR       A                   ; 1 mS delay for stabilization
0003 C20200               JNZ       Delays

0006 31FF3F     Stkint    LXI       SP,Stack            ; Initialize the stack pointer

0009 CD7902               CALL      Intram              ; Clear the RAM
000C CDA702               CALL      Clchbf              ; Clear the character buffer
000F CDF202               CALL      Dfault              ; Set up default parameters and values
0012 CD1603               CALL      Defsys              ; Clear the system

0015 C33700               JMP       Contrl              ; Go to interrupt wait

                ;*****************************************************************************
                ;*                    COMMUNICATION INTERRUPT VECTORS                        *
                ;*****************************************************************************

002C                      ORG       Intr5               ; RS-232 Interrupt vector
002C C34701     Int5_5    JMP       Serial              ; Jump to a more spacious work area

0034                      ORG       Intr6               ; IEEE-488 Interrupt vector
```

```
Huntsville Macro Assembler  8085 cross assembler for PC-DOS 2.0  v1.82L  Page 5
                                                          Module:  tf30194d

0034 C34100     Int6_5  JMP     Gpib            ; Jump to a more spacious work area

                ;*****************************************************************************
                ;*                      MAIN INTERRUPT SETUP AND WAIT                        *
                ;*****************************************************************************

0037 3E08       Contrl  MVI     A,08H           ; Set up 5.5, 6.5 as interrupts
0039 30                 SIM                     ; Set the interrupt mask
003A 31FF3F             LXI     SP,Stack        ; Initialize the stack

003D FB         Wait    EI                      ; Enable the interrupts
003E C33D00             JMP     Wait            ; Cirle around aimlessly

                ;*****************************************************************************
                ;*                    IEEE-488 INPUT/OUTPUT ROUTINE                          *
                ;*****************************************************************************

0041 F5         Gpib    PUSH    PSW             ; Save all registers
0042 C5                 PUSH    B
0043 D5                 PUSH    D
0044 E5                 PUSH    H

0045 3A943E             LDA     Byte5           ; Get the comm. lamp byte of the map
0048 E601               ANI     Hpibon          ; See if the light is already on
004A C25700             JNZ     Go_0            ; Skip the rest if it's on

004D 21943E             LXI     H,Byte5         ; Get the address of the bit map byte 6
0050 3E01               MVI     A,Hpibon        ; Get the mask to turn HPIB light on
0052 B6                 ORA     M               ; OR it with the contents of the byte
0053 77                 MOV     M,A             ; Re-store the result
0054 CD4C03             CALL    Setprt          ; Set the ports

0057 DB30       Go_0    IN      Int0            ; Get contents of GPIB interrupt reg
0059 47                 MOV     B,A             ; Stick it into the B register
005A E604               ANI     Spas            ; Is it a serial poll?
005C CA7700             JZ      Tlklst          ; If not, check for talk/listen

005F 3A973E     Serpll  LDA     Status          ; Get the status byte
0062 E6FE               ANI     0FEH            ; AND out the error bit
0064 32973E             STA     Status          ; Put it back
0067 D335               OUT     Spoll           ; Send it to the serial poll register

0069 3A943E             LDA     Byte5           ; Get the display byte
006C 47                 MOV     B,A             ; Shove it into B
006D 3E0C               MVI     A,Srqon+Erron   ; Get the SRQ and error masks
006F 2F                 CMA                     ; Complement the result
0070 A0                 ANA     B               ; AND in the bit map byte
0071 32943E             STA     Byte5           ; Store it in it's place

0074 C34101             JMP     Fin             ; Go home

0077 DB31       Tlklst  IN      Addsta          ; Get contents of address status reg
0079 E601               ANI     Lads            ; Check for listener addressed state
007B C28700             JNZ     Istnto          ; If it's an input, go handle it
```

```
007E 78         Talk    MOV     A,B             ; Swap registers again
007F E610               ANI     Bo              ; Check for byte out available intr
0081 CAF101             JZ      Fin             ; Go home if not available

0084 3A853C             LDA     Iobbc           ; Get the IEEE-488 output byte count
0087 FE01               CPI     01H             ; See if it's the last character
0089 CA9700             JZ      Eoiset          ; Set EOI if the last byte
008C DAAB00             JC      Linefd          ; Check for Xtra byte out and line feed

008F CDD704     Outie   CALL    Tkieo           ; Get the output byte
0092 D337               OUT     Datout          ; Send the data

0094 C3F101             JMP     Fin             ; Git

0097 3A993E     Eoiset  LDA     Systat          ; Get the system status byte
009A F602               ORI     Lstone          ; OR in the bit indicating last byte
009C 32993E             STA     Systat          ; Put the system status byte back
009F 3E08               MVI     A,Eoim          ; Get the EOI set mask
00A1 D333               OUT     Auxcmd          ; Send it to the auxilliary command reg

00A3 CDD704     Finish  CALL    Tkieo           ; Get the byte to be output
00A6 D337               OUT     Datout          ; Send it with EOI
00A8 C3F101             JMP     Fin             ; Bye!

00AB 3A993E     Linefd  LDA     Systat          ; Get the system status byte
00AE E602               ANI     Lstone          ; Was a final byte sent?
00B0 C2BB00             JNZ     Clr_it          ; Ignore byte out and clear flag if so

00B3 3E0A               MVI     A,0AH           ; Put a line feed into the output buff.
00B5 CD9F04             CALL    Adieo           ; Place into buffer
00B8 C39700             JMP     Eoiset          ; Set EOI with this one

00BB 3A993E     Clr_it  LDA     Systat          ; Get the system status byte
00BE 47                 MOV     B,A             ; Store it in B
00BF 3E02               MVI     A,Lstone        ; Get the last byte mask
00C1 2F                 CMA                     ; Complement it
00C2 A0                 ANA     B               ; AND the flag out of the status byte
00C3 32993E             STA     Systat          ; Re-store the status byte
00C6 3A973E             LDA     Status          ; Get the GPIB status byte
00C9 47                 MOV     B,A             ; Stick it in B
00CA 3E06               MVI     A,Replie+Serl   ; Get the two reply masks
00CC 2F                 CMA                     ; Complement them
00CD A0                 ANA     B               ; AND in the old byte
00CE 32973E             STA     Status          ; Restore the new byte
00D1 D335               OUT     Spoll           ; New serial poll condition
00D3 CD9F02             CALL    Clrobf          ; Clear the output buffer
00D6 C3F101             JMP     Fin             ; Go away, ignoring the byte out

00D9 78         Ieinto  MOV     A,B             ; Get the interrupt status byte
00DA E620               ANI     Bi              ; Is there a byte in?
00DC CAF101             JZ      Fin             ; Get out if not

00DF DB37               IN      Datin           ; Get the input byte
00E1 4F                 MOV     C,A             ; Make a copy of it in C
00E2 CD8504             CALL    Adiei           ; Put it into the buffer
```

```
00E5 3A993E             LDA     Systat          ; Get the system status byte
00E8 E640               ANI     Pastru          ; Check for pass-thru mode
00EA CA2901             JZ      Cmpr            ; Normal process if not

00ED 3E7F               MVI     A,Stoppt        ; Bring in the pass thru terminator
00EF B9                 CMP     C               ; Check it against received byte
00F0 C21501             JNZ     Kp              ; If it is not, keep checkin'

00F3 3A993E             LDA     Systat          ; Get the system status byte
00F6 47                 MOV     B,A             ; Save it in B
00F7 3E40               MVI     A,Pastru        ; Get the pass-thru mask
00F9 2F                 CMA                     ; Complement it
00FA A0                 ANA     B               ; AND the bit out of the byte
00FB 32993E             STA     Systat          ; Re-store the new status byte
00FE 3A973E             LDA     Status          ; Get the GPIB status byte
0101 47                 MOV     B,A             ; Shove it up your B
0102 3E08               MVI     A,Pstr          ; Get the pass thru mask
0104 2F                 CMA                     ; Complement it
0105 A0                 ANA     B               ; AND it out of the byte
0106 32973E             STA     Status          ; Stick it back, Jack
0109 CDD20A             CALL    Srq_do          ; See if an SRQ is appropriate
010C CD0A02             CALL    Fploff          ; Kill the front panel lamp
010F CD8702             CALL    Cliibf          ; Clear the 488 buffer
0112 C3FA01             JMP     Scat            ; Git

0115 78         Kp      MOV     A,B             ; Get the interrupt status byte
0116 E608               ANI     Eoim            ; Was EOI asserted with this byte?
0118 C21E01             JNZ     Ptpg            ; If not, wait for next byte
011B
011B C3FA01             JMP     Scat            ; Keep goin'

011E 3A993E     Ptpg    LDA     Systat          ; Get the byte again
0121 F680               ORI     Ptproc          ; Set send pass-through flag
0123 32993E             STA     Systat          ; Re-store the byte
0126 C33B01             JMP     Pr1             ; Go to deliver it

0129 CD6605     Cmpr    CALL    Cln488          ; Process the byte just received
012C 3E0A               MVI     A,0AH           ; Put a line feed into the accumulator
012E B9                 CMP     C               ; Check it against the character
012F CA3B01             JZ      Pr1             ; Process if it is a line feed

0132 78                 MOV     A,B             ; Get the interrupt status byte
0133 E608               ANI     Eoim            ; Was EOI asserted with this byte?
0135 C23B01             JNZ     Pr1             ; If so, process the input

0138 C3FA01     Keepup  JMP     Scat            ; Keep going

013B CD0A02     Pr1     CALL    Fploff          ; Kill front panel lamp
013E CDE105             CALL    Pr488           ; Process the input
0141 CD8702             CALL    Cliibf          ; Clear the 488 input buffer

0144 C3FA01             JMP     Scat            ; Leave town

        ;*****************************************************************************
```

```
                ;*                    RS-232 INPUT/OUTPUT ROUTINE                    *
                ;*********************************************************************

0147 F5         Serial  PUSH    PSW             ; Save all registers
0148 C5                 PUSH    B
0149 D5                 PUSH    D
014A E5                 PUSH    H

014B 3A943E             LDA     Byte5           ; Get the comm. lamp byte of the map
014E E602               ANI     Serlon          ; See if the light is already on
0150 C25D01             JNZ     Go_1            ; Skip the rest if it's on

0153 21943E             LXI     H,Byte5         ; Get the address of the bit map byte 6
0156 3E02               MVI     A,Serlon        ; Get the mask to turn SERIAL light on
0158 B6                 ORA     M               ; OR it with the contents of the byte
0159 77                 MOV     M,A             ; Re-store the result
015A CD4C03             CALL    Setprt          ; Set the ports

015D DB3A       Go_1    IN      Intrid          ; Get contents of serial interrupt reg
015F E606               ANI     Rcv+Txm         ; Is it transmit or receive?
0161 CAF101             JZ      Fin             ; Blow town if it ain't

0164 E604               ANI     Rcv             ; Check to see if it was receive
0166 C28901             JNZ     Rcvser          ; Go take care of it if it was

0169 3A8B3D             LDA     Robbc           ; Get the RS-232 output byte count
016C FE00               CPI     00H             ; See if there be no mo'
016E CA7901             JZ      Laster          ; Spit out a terminator if so

0171 CD6704             CALL    Tkrso           ; Get the next buffer byte
0174 D338               OUT     Xmtbuf          ; Transmit it
0176 C3FA01             JMP     Scat            ; Return to interrupt wait

0179 3E01       Laster  MVI     A,Recda         ; Get the bit mask to disable Xmit intr
017B D339               OUT     Int2            ; Send it to the interrupt port
017D 21953E             LXI     H,Sertrm        ; Get the address of the term char
0180 7E                 MOV     A,M             ; Get the serial termination character
0181 D338               OUT     Xmtbuf          ; Output it
0183 CD9F02             CALL    Clrobf          ; Clear the output buffer
0186 C3F101             JMP     Fin             ; Git

0189 DB3D       Rcvser  IN      Linsta          ; Is there an input byte ready?
018B E601               ANI     Drbit           ; Data ready mask
018D CAF101             JZ      Fin             ; Return if not

0190 DB38               IN      Recbuf          ; Input the character

0192 4F                 MOV     C,A             ; Save a copy in C
0193 CD2F04             CALL    Adrsi           ; Store the input in the buffer

0196 3A993E             LDA     Systat          ; Get the system status byte
0199 E640               ANI     Pastru          ; Check for system pass-thru
019B CADE01             JZ      Normal          ; Normal process if not

019E 3A953E             LDA     Sertrm          ; Get the current serial termination
```

```
01A1 B9                 CMP     C               ; Is it a serial termination character?
01A2 CACD01             JZ      Serpth          ; Process if it is

01A5 3E7F               MVI     A,Stoppt        ; Get the pass-thru terminator
01A7 B9                 CMP     C               ; Compare it to the received byte
01A8 C2FA01             JNZ     Scat            ; Git if it ain't

01AB 3A993E             LDA     Systat          ; Get the system status byte
01AE 47                 MOV     B,A             ; Save it in B
01AF 3E40               MVI     A,Pastru        ; Get the pass-thru mask
01B1 2F                 CMA                     ; Complement it
01B2 A0                 ANA     B               ; AND the bit out of the byte
01B3 32993E             STA     Systat          ; Re-store the status byte
01B6 3A973E             LDA     Status          ; Get the GPIB status byte
01B9 47                 MOV     B,A             ; Shove it up your B
01BA 3E08               MVI     A,Pstr          ; Get the pass thru mask
01BC 2F                 CMA                     ; Complement it
01BD A0                 ANA     B               ; AND it out of the byte
01BE 32973E             STA     Status          ; Stick it back, Jack
01C1 CDD20A             CALL    Srq_do          ; See if an SRQ is appropriate
01C4 CD0A02             CALL    Fploff          ; Kill front panel lamp
01C7 CD9702             CALL    Clribf          ; Clear the buffer
01CA C3FA01             JMP     Scat            ; Bye-bye

01CD CD4705     Serpth  CALL    Tkrsia          ; Scrap the termination character
01D0 3A993E             LDA     Systat          ; Get the system status byte
01D3 F680               ORI     Ptproc          ; Set the pass-thru process flag
01D5 32993E             STA     Systat          ; Put the byte back
01D8 C3E501             JMP     Proc2           ; Process the input

01DB CD7605     Normal  CALL    Cln232          ; Process the byte
01DE 3A953E             LDA     Sertrm          ; Get the current serial termination
01E1 B9                 CMP     C               ; Is it a serial termination character?
01E2 C2F101             JNZ     Fin             ; Just keep going if not

01E5 CD0A02     Proc2   CALL    Fploff          ; Turn off front panel lamp
01E8 CDCC05             CALL    Pr232           ; Process the command
01EB CD9702     Gh      CALL    Clribf          ; Clear the 232 input buffer

01EE C3FA01             JMP     Scat            ; Beat it

                ;*****************************************************************************
                ;*                          "WRAP-UP" ROUTINES                               *
                ;*****************************************************************************

01F1 CD0A02     Fin     CALL    Fploff          ; Kill F.P. lamps
01F4 CD1502             CALL    Bufchk          ; Check the buffers
01F7 C30502             JMP     Hgt             ; Git while the goin's good

01FA 3A993E     Scat    LDA     Systat          ; Get the system status byte
01FD 47                 MOV     B,A             ; Save it in B
01FE 3E00               MVI     A,Ptproc        ; Get the temporary pass-thru mask
0200 2F                 CMA                     ; Complement it
0201 A0                 ANA     B               ; AND it out of the byte
0202 32993E             STA     Systat          ; Put the byte back
```

```
0205 E1          Hgt       POP       H                    ; Retrieve the registers
0206 D1                    POP       D
0207 C1                    POP       B
0208 F1                    POP       PSW

0209 C9                    RET                            ; Bye!

020A 21943E      Fploff    LXI       H,Byte5              ; Get the address of byte 6 of bit map
020D 3E03                  MVI       A,Serlon+Hpibon      ; Get masks for lights on/off
020F 2F                    CMA                            ; Complement it
0210 A6                    ANA       M                    ; AND it with what's already there
0211 77                    MOV       M,A                  ; Put it back where it came from
0212 CD4C03                CALL      Setprt               ; Set the output ports

0215 C9                    RET

0216 F5          Bufchk    PUSH      PSW                  ; Save A and CCR

0217 3A003C                LDA       Iibfp                ; Get 488 input buffer fill pointer
021A CD7302                CALL      Checkr               ; Checkitout
021D 32003C                STA       Iibfp                ; Puterback

0220 3A013C                LDA       Iibep                ; Get 488 input buffer MT pointer
0223 CD7302                CALL      Checkr               ; Checkitout
0226 32013C                STA       Iibep                ; Puterback

0229 3A833C                LDA       Iobfp                ; Get 488 output buffer fill pointer
022C CD7302                CALL      Checkr               ; Checkitout
022F 32833C                STA       Iobfp                ; Puterback

0232 3A843C                LDA       Iobep                ; Get 488 output buffer MT pointer
0235 CD7302                CALL      Checkr               ; Checkitout
0238 32843C                STA       Iobep                ; Puterback

023B 3A063D                LDA       Ribfp                ; Get 232 input buffer fill pointer
023E CD7302                CALL      Checkr               ; Checkitout
0241 32063D                STA       Ribfp                ; Puterback

0244 3A073D                LDA       Ribep                ; Get 232 input buffer MT pointer
0247 CD7302                CALL      Checkr               ; Checkitout
024A 32073D                STA       Ribep                ; Puterback

024D 3A893D                LDA       Robfp                ; Get 232 output buffer fill pointer
0250 CD7302                CALL      Checkr               ; Checkitout
0253 32893D                STA       Robfp                ; Puterback

0256 3A8A3D                LDA       Robep                ; Get 232 output buffer MT pointer
0259 CD7302                CALL      Checkr               ; Checkitout
025C 328A3D                STA       Robep                ; Puterback

025F 3A0C3E                LDA       Chbfp                ; Get character buffer fill pointer
0262 CD7302                CALL      Checkr               ; Checkitout
0265 320C3E                STA       Chbfp                ; Puterback
```

```
Huntsville Macro Assembler  8085 cross assembler for PC-DOS 2.0  v1.82L  Page 11
                                                        Module:  tf30194d

0268 3A0D3E             LDA     Chbep           ; Get character buffer MT pointer
026B CD7302             CALL    Checkr          ; Checkitout
026E 320D3E             STA     Chbep           ; Puterback

0271 F1                 POP     PSW             ; Get the stuff back

0272 C9                 RET

0273 FE80       Checkr  CPI     80H             ; Make sure it's not too high
0275 D8                 RC                      ; Return if O.K.
0276 3E00               MVI     A,00H           ; Reset the pointer if too high
0278 C9                 RET

                ;*****************************************************************************
                ;*                          RAM CLEAR ROUTINE                                *
                ;*****************************************************************************

0279 21FC3F     Intram  LXI     H,Stack-3       ; Start clearing below return address

027C 3EFF               MVI     A,0FFH          ; Clear stack area
027E 47                 MOV     B,A             ; Hold the count

027F 3600       Rambot  MVI     M,00H           ; Place a 0 into RAM location
0281 2B                 DCX     H               ; Decrement the H register by 1
0282 05                 DCR     B               ; Decrement the count variable
0283 C27F02             JNZ     Rambot          ; Keep clearing until 0 is reached
0286 C9                 RET                     ; Return to caller when finished

                ;*****************************************************************************
                ;*                        BUFFER CLEAR ROUTINES                              *
                ;*****************************************************************************

0287 F5         Cliibf  PUSH    PSW             ; Save registers
0288 E5                 PUSH    H

0289 21003C             LXI     H,Iibfp         ; Get base address for 488 input buff
028C C3AC02             JMP     Clearr          ; Go to the clearing routine

028F F5         Cliobf  PUSH    PSW             ; Save registers
0290 E5                 PUSH    H

0291 21833C             LXI     H,Iobfp         ; Get base address for 488 output buff
0294 C3AC02             JMP     Clearr          ; Go to the clearing routine

0297 F5         Clribf  PUSH    PSW             ; Save registers
0298 E5                 PUSH    H

029B 21073D             LXI     H,Ribfp         ; Get base address for 232 input buff
029C C3AC02             JMP     Clearr          ; Go to the clearing routine


02A1 F5         Clrobf  PUSH    PSW             ; Save registers
02A2 E5                 PUSH    H

02A5 21873D             LXI     H,Robfp         ; Get base address for 232 output buff
```

```
02A4 C3AC02             JMP     Clearr          ; Go to the clearing routine

02A7 F5         Clchbf  PUSH    PSW             ; Save registers
02A8 E5                 PUSH    H

02A9 210C3E             LXI     H,Chbfp         ; Get base address for character buffer

02AC 3E83       Clearr  MVI     A,83H           ; Clear 131 memory locations

02AE 3600       Countr  MVI     M,00H           ; Put a "0" into the memory location
02B0 3D                 DCR     A               ; Decrement the "A" register
02B1 CAB802             JZ      Fns             ; Scat if all done

02B4 23                 INX     H               ; Increment the byte address by 1
02B5 C3AE02             JMP     Countr          ; Return to the counter

02B8 E1         Fns     POP     H               ; Clean up the mess
02B9 F1                 POP     PSW

02BA C9                 RET                     ; G'home Gnome

                ;*****************************************************************************
                ;*                   INITIALIZE THE 488 CHIP AND BUFFERS                     *
                ;*****************************************************************************

02BB F5         Int488  PUSH    PSW             ; Save registers
02BC E5                 PUSH    H

02BD 3E80               MVI     A,Swrst         ; Initialize the 488 chip
02BF D333               OUT     Auxcmd          ; Output to auxilliary command register

02C1 DB34               IN      Addswt          ; Get the address from the switches
02C3 E61F               ANI     1FH             ; AND out the address
02C5 D334               OUT     Addreg          ; Send it to the address register
02C7 329A3E             STA     Adgpib          ; Store it in RAM

02CA 3E34               MVI     A,Bi+Bo+Spas    ; Allow byte in, byte out, and SPOLL
02CC D330               OUT     Int0            ; Send it to interrupt register 0

02CE AF                 XRA     A               ; Place a 0 into:
02CF D335               OUT     Spoll           ;                   the serial poll reg
02D1 D331               OUT     Int1            ;                   interrupt reg 1
02D3 32973E             STA     Status          ;                   the status byte

02D6 3E00               MVI     A,Clrst         ; Clear software reset
02D8 D333               OUT     Auxcmd          ; Send to auxilliary command register

02DA CD8702             CALL    Cliibf          ; Clear 488 input buffer
02DD CD8F02             CALL    Cliobf          ; Clear 488 output buffer

02E0 E1                 POP     H               ; Retrieve the registers
02E1 F1                 POP     PSW

02E2 C9                 RET                     ; Return to the calling context
```

```
                ;*****************************************************************************
                ;*                    INITIALIZE THE 232 CHIP AND BUFFERS                    *
                ;*****************************************************************************

02E3 F5         Int232  PUSH    PSW                     ; Save the registers
02E4 E5                 PUSH    H

02E5 3E01               MVI     A,Recda                 ; Allow data received interrupt
02E7 D339               OUT     Int2                    ; Send to the RS-232 interrupt register

02E9 CD9702             CALL    Clribf                  ; Clear 232 input buffer
02EC CD9F02             CALL    Clrobf                  ; Clear 232 output buffer

02EF E1                 POP     H                       ; Retrieve the registers
02F0 F1                 POP     PSW

02F1 C9                 RET                             ; Return to caller

                ;*****************************************************************************
                ;*                          SET DEFAULT BIT MAP                              *
                ;*****************************************************************************

02F2 F5         Dfault  PUSH    PSW                     ; Stick the registers onto the stack

02F3 3E00               MVI     A,Defbt0                ; Load the first default bit map byte
02F5 32BF3E             STA     Byte0                   ; into the first bit map byte
02F8 3E00               MVI     A,Defbt1                ; Load the second default bit map byte
02FA 32903E             STA     Byte1                   ; into the second bit map byte
02FD 3E88               MVI     A,Defbt2                ; Load the third default bit map byte
02FF 32913E             STA     Byte2                   ; into the third bit map byte
0302 3E04               MVI     A,Defbt3                ; Load the fourth default bit map byte
0304 32923E             STA     Byte3                   ; into the fourth bit map byte
0307 3E00               MVI     A,Defbt4                ; Load the fifth default bit map byte
0309 32933E             STA     Byte4                   ; Into the fifth bit map byte
030C 3E00               MVI     A,Defbt5                ; Load the sixth default bit map byte
030E 32943E             STA     Byte5                   ; into the sixth bit map byte

0311 CD4C03             CALL    Setprt                  ; Send the bit map to the ports

0314 F1                 POP     PSW                     ; Get your s--- together

0315 C9                 RET                             ; Go on home

                ;*****************************************************************************
                ;*                      VARIOUS SYSTEM DEFAULT SETUPS                        *
                ;*****************************************************************************

0316 F5         Defsys  PUSH    PSW                     ; Save the registers
0317 E5                 PUSH    H

                ; RS 232 Default:

0318 CD0302             CALL    Int232                  ; Clear the 232 interface
031B 3E9B               MVI     A,Intwrd+Dlab           ; Set up the default parity conditions
031D D3F3               OUT     Linctl                  ; and enable the divisor
```

```
031F 211400             LXI     H,Bd9600        ; Load the code for 9600 Baud
0322 7D                 MOV     A,L             ; Store the lower byte
0323 D338               OUT     Dllsb           ; in the lower divisor byte register
0325 7C                 MOV     A,H             ; Store the upper byte
0326 D339               OUT     Dlmsb           ; in the upper divisor byte register

0328 DB3B               IN      Linctl          ; Load the contents of the line ctl reg
032A E67F               ANI     7FH             ; AND it's last bit out (disable divis)
032C D33B               OUT     Linctl          ; Send it back C.O.D.

032E 3E03               MVI     A,Dtr+Rts       ; Set the DTR and RTS lines up
0330 D33C               OUT     Modctl          ; Enable them

0332 3E0B               MVI     A,Intwrd        ; Set the serial data structure byte
0334 32963E             STA     Wordse          ; Store it in it's RAM location

0337 3E0A               MVI     A,0AH           ; Put a line feed into
0339 32953E             STA     Sertrm          ; The serial termination byte

                ; Other system defaults:

033C 3E01               MVI     A,Srmask        ; Get the default SRQ mask
033E 32983E             STA     Srqmsk          ; Store it in it's place

0341 3E04               MVI     A,Sysbyt        ; Load the default system status byte
0343 32993E             STA     Systat          ; into it's RAM location

0346 CDBB02             CALL    Int488          ; Clear the 488 interface

0349 E1                 POP     H               ; Clean up de mess
034A F1                 POP     PSW

034B C9                 RET                     ; It's been real!

                ;*****************************************************************************
                ;*                        SEND THE BIT MAP TO THE PORTS                      *
                ;*****************************************************************************

034C F5         Setprt  PUSH    PSW             ; Save the registers
034D E5                 PUSH    H

034E 218F3E             LXI     H,Byte0         ; Get the address for byte 0
0351 7E                 MOV     A,M             ; Load the first bit map byte into A
0352 D340               OUT     Port40          ; Send it to the first output port
0354 21903E             LXI     H,Byte1         ; Get the address for byte 1
0357 7E                 MOV     A,M             ; Load the second bit map byte into A
0358 D341               OUT     Port41          ; Send it to the second output port
035A 21913E             LXI     H,Byte2         ; Get the address for byte 2
035D 7E                 MOV     A,M             ; Load the third bit map byte into A
035E D342               OUT     Port42          ; Send it to the third output port
0360 21923E             LXI     H,Byte3         ; Get the address for byte 3
0363 7E                 MOV     A,M             ; Load the fourth bit map byte into A
0364 D343               OUT     Port43          ; Send it to the fourth output port
0366 21933E             LXI     H,Byte4         ; Get the address for byte 4
```

```
0369 7E                 MOV     A,M             ; Load the fifth bit map byte into A
036A 2F                 CMA                     ; The attenuator must be complemented
036B D344               OUT     Port44          ; Send it to the fifth output port
036D 21943E             LXI     H,Byte5         ; Get the address for byte 5
0370 7E                 MOV     A,M             ; Load the sixth bit map byte into A
0371 D345               OUT     Port45          ; Send it to the sixth output port

0373 E1                 POP     H               ; Gimmie dat back!
0374 F1                 POP     PSW

0375 C9                 RET                     ; G'wan, git outta here!

                ;*****************************************************************************
                ;*                          POINTER RETURN ROUTINES                          *
                ;*****************************************************************************

                ; These routines load the H/L register pair with the absolute address of the
                ; byte in the specified buffer as pointed to by the specified pointer.

                ; RS-232 input buffer fill pointer

0376 F5         Rsifp   PUSH    PSW             ; Save the accumulator
0377 21093D             LXI     H,Ribase        ; Get the address of the buffer base
037A 3A063D             LDA     Ribfp           ; Get the fill pointer
037D 85                 ADD     L               ; Add the index to L
037E 6F                 MOV     L,A             ; Put the updated index back
037F D21304             JNC     Leeve           ; Scat if no carry
0382 24                 INR     H               ; Increment the MSB
0383 C31304             JMP     Leeve           ; Make tracks

                ; RS-232 input buffer MT pointer

0386 F5         Rsiep   PUSH    PSW             ; Save the accumulator
0387 21093D             LXI     H,Ribase        ; Get the address of the buffer base
038A 3A073D             LDA     Ribep           ; Get the fill pointer
038D 85                 ADD     L               ; Add the index to L
038E 6F                 MOV     L,A             ; Put the updated index back
038F D21304             JNC     Leeve           ; Scat if no carry
0392 24                 INR     H               ; Increment the MSB
0393 C31304             JMP     Leeve           ; Make tracks

                ; RS-232 output buffer fill pointer

0396 F5         Rsofp   PUSH    PSW             ; Save the accumulator
0397 210C3D             LXI     H,Robase        ; Get the address of the buffer base
039A 3A893D             LDA     Robfp           ; Get the fill pointer
039D 85                 ADD     L               ; Add the index to L
039E 6F                 MOV     L,A             ; Put the updated index back
039F D21304             JNC     Leeve           ; Scat if no carry
03A2 24                 INR     H               ; Increment the MSB
03A3 C31304             JMP     Leeve           ; Make tracks

                ; RS-232 output buffer MT pointer

03A6 F5         Rsoep   PUSH    PSW             ; Save the accumulator
```

```
03A7 218C3D             LXI     H,Robase        ; Get the address of the buffer base
03AA 3A8A3D             LDA     Robep           ; Get the fill pointer
03AD 85                 ADD     L               ; Add the index to L
03AE 6F                 MOV     L,A             ; Put the updated index back
03AF D21304             JNC     Leeve           ; Scat if no carry
03B2 24                 INR     H               ; Increment the MSB
03B3 C31304             JMP     Leeve           ; Make tracks

                ; IEEE-488 input buffer fill pointer

03B6 F5         Ieifp   PUSH    PSW             ; Save the accumulator
03B7 21033C             LXI     H,Iibase        ; Get the address of the buffer base
03BA 3A003C             LDA     Iibfp           ; Get the fill pointer
03BD 85                 ADD     L               ; Add the index to L
03BE 6F                 MOV     L,A             ; Put the updated index back
03BF D21304             JNC     Leeve           ; Scat if no carry
03C2 24                 INR     H               ; Increment the MSB
03C3 C31304             JMP     Leeve           ; Make tracks

                ; IEEE-488 input buffer MT pointer

03C6 F5         Ieiep   PUSH    PSW             ; Save the accumulator
03C7 21033C             LXI     H,Iibase        ; Get the address of the buffer base
03CA 3A013C             LDA     Iibep           ; Get the fill pointer
03CD 85                 ADD     L               ; Add the index to L
03CE 6F                 MOV     L,A             ; Put the updated index back
03CF D21304             JNC     Leeve           ; Scat if no carry
03D2 24                 INR     H               ; Increment the MSB
03D3 C31304             JMP     Leeve           ; Make tracks

                ; IEEE-488 output buffer fill pointer

03D6 F5         Ieofp   PUSH    PSW             ; Save the accumulator
03D7 21863C             LXI     H,Iobase        ; Get the address of the buffer base
03DA 3A833C             LDA     Iobfp           ; Get the fill pointer
03DD 85                 ADD     L               ; Add the index to L
03DE 6F                 MOV     L,A             ; Put the updated index back
03DF D21304             JNC     Leeve           ; Scat if no carry
03E2 24                 INR     H               ; Increment the MSB
03E3 C31304             JMP     Leeve           ; Make tracks

                ; IEEE-488 output buffer MT pointer

03E6 F5         Ieoep   PUSH    PSW             ; Save the accumulator
03E7 21863C             LXI     H,Iobase        ; Get the address of the buffer base
03EA 3A843C             LDA     Iobep           ; Get the empty pointer
03ED 85                 ADD     L               ; Add the index to L
03EE 6F                 MOV     L,A             ; Put the updated index back
03EF D21304             JNC     Leeve           ; Scat if no carry
03F2 24                 INR     H               ; Increment the MSB
03F3 C31304             JMP     Leeve           ; Make tracks

                ; Character buffer fill pointer

03F6 F5         Chfp    PUSH    PSW             ; Save the accumulator
```

```
03F7 210F3E             LXI     H,Chbase        ; Get the address of the buffer base
03FA 3A0C3E             LDA     Chbfp           ; Get the fill pointer
03FD 85                 ADD     L               ; Add the index to L
03FE 6F                 MOV     L,A             ; Put the updated index back
03FF D21304             JNC     Leeve           ; Scat if no carry
0402 24                 INR     H               ; Increment the MSB
0403 C31304             JMP     Leeve           ; Make tracks

                ; Character buffer MT pointer

0406 F5         Chep    PUSH    PSW             ; Save the accumulator
0407 210F3E             LXI     H,Chbase        ; Get the address of the buffer base
040A 3A0D3E             LDA     Chbep           ; Get the MT pointer
040D 85                 ADD     L               ; Add the index to L
040E 6F                 MOV     L,A             ; Put the updated index back
040F D21304             JNC     Leeve           ; Scat if no carry
0412 24                 INR     H               ; Increment the MSB

0413 F1         Leeve   POP     PSW             ; Get accumulator back
0414 C9                 RET                     ; Take off

                ;*****************************************************************************
                ;*          ROUTINES TO TAKE FROM, AND TO ADD TO, THE BUFFERS                *
                ;*****************************************************************************

                ; These routines will take from, or add to, one byte to/from the accumulator.
                ; The pointer(s) to the specified buffer are updated accordingly.

                ; Add a byte to the RS-232 output buffer

0415 F5         Adrso   PUSH    PSW             ; Save registers
0416 E5                 PUSH    H

0417 CD9603             CALL    Rsofp           ; Address the buffer's fill byte
041A 77                 MOV     M,A             ; Put the byte into the buffer
041B 3A893D             LDA     Robfp           ; Get the fill pointer
041E 3C                 INR     A               ; Increment it
041F 32893D             STA     Robfp           ; Replace it
0422 3A8B3D             LDA     Robbc           ; Get the byte count
0425 3C                 INR     A               ; Increment it
0426 328B3D             STA     Robbc           ; Put it back
0429 CD1602             CALL    Bufchk          ; Check the buffers

042C E1                 POP     H               ; Get the registers back
042D F1                 POP     PSW

042E C9                 RET

                ; Add a byte to the RS-232 input buffer

042F F5         Adrsi   PUSH    PSW             ; Save registers
0430 E5                 PUSH    H

0431 CD9203             CALL    Rsifp           ; Address the buffer's fill byte
0434 77                 MOV     M,A             ; Put the byte into the buffer
```

```
0435 3A063D             LDA     Ribfp           ; Get the fill pointer
0438 3C                 INR     A               ; Increment it
0439 32063D             STA     Ribfp           ; Replace it
043C 3A083D             LDA     Ribbc           ; Get the RS-232 byte count
043F 3C                 INR     A               ; Increment it
0440 32083D             STA     Ribbc           ; Put it back
0443 CD1602             CALL    Bufchk          ; Check the buffers

0446 E1                 POP     H               ; Get the registers back
0447 F1                 POP     PSW

0448 C9                 RET

                ; Take a byte from the RS-232 input buffer

0449 C5         Tkrsi   PUSH    B               ; Save registers
044A E5                 PUSH    H

044B 3A083D             LDA     Ribbc           ; Get the RS-232 byte count
044E FE00               CPI     00H             ; Is it 0?
0450 CA6105             JZ      Cr              ; Git if MT
0453 3D                 DCR     A               ; Decrement it
0454 32083D             STA     Ribbc           ; Put it back
0457 CD8603             CALL    Rsiep           ; Address the buffer's MT byte
045A 7E                 MOV     A,M             ; Put the byte into the accumulator
045B 21073D             LXI     H,Ribep         ; Get the address of the MT pointer
045E 46                 MOV     B,M             ; Get the MT pointer
045F 04                 INR     B               ; Increment it
0460 70                 MOV     M,B             ; Replace it
0461 CD1602             CALL    Bufchk          ; Check the buffers

0464 E1                 POP     H               ; Get the registers back
0465 C1                 POP     B

0466 C9                 RET

                ; Take a byte from the RS-232 output buffer

0467 C5         Tkrso   PUSH    B               ; Save registers
0468 E5                 PUSH    H

0469 3A0B3D             LDA     Robbc           ; Get the byte count
046C FE00               CPI     00H             ; Is it 0?
046E CA6105             JZ      Cr              ; Git if MT
0471 3D                 DCR     A               ; Decrement it
0472 320B3D             STA     Robbc           ; Put it back
0475 CDA603             CALL    Rsoep           ; Address the buffer's MT byte
0478 7E                 MOV     A,M             ; Put the byte into the accumulator
0479 218A3D             LXI     H,Robep         ; Get the address of the MT pointer
047C 46                 MOV     B,M             ; Get the MT pointer
047D 04                 INR     B               ; Increment it
047E 70                 MOV     M,B             ; Replace it
047F CD1602             CALL    Bufchk          ; Check the buffers

0482 E1                 POP     H               ; Get the registers back
```

```
0483 C1                 POP     B

0484 C9                 RET

                ; Add a byte to the IEEE-488 input buffer

0485 F5         Adiei   PUSH    PSW             ; Save registers
0486 E5                 PUSH    H

0487 CDB603             CALL    Ieifp           ; Address the buffer's fill byte
048A 77                 MOV     M,A             ; Put the byte into the buffer
048B 3A003C             LDA     Iibfp           ; Get the fill pointer
048E 3C                 INR     A               ; Increment it
048F 32003C             STA     Iibfp           ; Replace it
0492 3A023C             LDA     Iibbc           ; Get the byte count
0495 3C                 INR     A               ; Increment it
0496 32023C             STA     Iibbc           ; Put it back
0499 CD1602             CALL    Bufchk          ; Check the buffers

049C E1                 POP     H               ; Get the registers back
049D F1                 POP     PSW

049E C9                 RET

                ; Add a byte to the IEEE-488 output buffer

049F F5         Adieo   PUSH    PSW             ; Save registers
04A0 E5                 PUSH    H

04A1 CDD603             CALL    Ieofp           ; Address the buffer's fill byte
04A4 77                 MOV     M,A             ; Put the byte into the buffer
04A5 3A833C             LDA     Iobfp           ; Get the fill pointer
04A8 3C                 INR     A               ; Increment it
04A9 32833C             STA     Iobfp           ; Replace it
04AC 3A853C             LDA     Iobbc           ; Get the byte count
04AF 3C                 INR     A               ; Increment it
04B0 32853C             STA     Iobbc           ; Put it back
04B3 CD1602             CALL    Bufchk          ; Check the buffers

04B6 E1                 POP     H               ; Get the registers back
04B7 F1                 POP     PSW

04B8 C9                 RET

                ; Take a byte from the IEEE-488 input buffer

04B9 C5         Tiiei   PUSH    B               ; Save registers
04BA E5                 PUSH    H

04BC 3A023C             LDA     Iibbc           ; Get the byte count
04BF FE00               CPI     00H             ; Is it 0?
04C1 CA0504             JZ      Cr              ; Git if MT
04C4 3D                 DCR     A               ; Decrement it
04C5 32023C             STA     Iibbc           ; Put it back
04C8 CDE603             CALL    Ieiep           ; Address the buffer's MT byte
```

```
04CA 7E                 MOV     A,M             ; Put the byte into the accumulator
04CB 21013C             LXI     H,Iibep         ; Get the address of the MT pointer
04CE 46                 MOV     B,M             ; Get the MT pointer
04CF 04                 INR     B               ; Increment it
04D0 70                 MOV     M,B             ; Replace it
04D1 CD1602             CALL    Bufchk          ; Check the buffers

04D4 E1                 POP     H               ; Get the registers back
04D5 C1                 POP     B

04D6 C9                 RET

                ; Take a byte from the IEEE-488 output buffer

04D7 C5         Tbieo   PUSH    B               ; Save registers
04D8 E5                 PUSH    H

04D9 3A853C             LDA     Iobbc           ; Get the byte count
04DC FE00               CPI     00H             ; Is it 0?
04DE CA6105             JZ      Cr              ; Git if MT
04E1 3D                 DCR     A               ; Decrement it
04E2 32853C             STA     Iobbc           ; Put it back
04E5 CDE603             CALL    Ieoep           ; Address the buffer's MT byte
04E8 7E                 MOV     A,M             ; Put the byte into the accumulator
04E9 21843C             LXI     H,Iobep         ; Get the address of the MT pointer
04EC 46                 MOV     B,M             ; Get the MT pointer
04ED 04                 INR     B               ; Increment it
04EE 70                 MOV     M,B             ; Replace it
04EF CD1602             CALL    Bufchk          ; Check the buffers

04F2 E1                 POP     H               ; Get the registers back
04F3 C1                 POP     B

04F4 C9                 RET

                ; Add a byte to the character buffer

04F5 F5         Adch    PUSH    PSW             ; Save registers
04F6 E5                 PUSH    H

04F7 CDF603             CALL    Chfp            ; Address the buffer's fill byte
04FA 77                 MOV     M,A             ; Put the byte into the buffer
04FB 3A0C3E             LDA     Chbfp           ; Get the fill pointer
04FE 3C                 INR     A               ; Increment it
04FF 320C3E             STA     Chbfp           ; Replace it
0502 3A0E3E             LDA     Chbbc           ; Get the byte count
0505 3C                 INR     A               ; Increment it
0506 320E3E             STA     Chbbc           ; Put it back
0509 CD1602             CALL    Bufchk          ; Check the buffers

050C E1                 POP     H               ; Get the registers back
050D F1                 POP     PSW

050E C9                 RET
```

```
                ; Take a byte from the character buffer

050F C5         Tkch    PUSH    B               ; Save registers
0510 E5                 PUSH    H

0511 3A0E3E             LDA     Chbbc           ; Get the byte count
0514 FE00               CPI     00H             ; Is it 0?
0516 CA6105             JZ      Cr              ; Git if MT
0519 3D                 DCR     A               ; Decrement it
051A 320E3E             STA     Chbbc           ; Put it back
051D CD0604             CALL    Chep            ; Address the buffer's MT byte
0520 7E                 MOV     A,M             ; Put the byte into the accumulator
0521 210D3E             LXI     H,Chbep         ; Get the address of the MT pointer
0524 46                 MOV     B,M             ; Get the MT pointer
0525 04                 INR     B               ; Increment it
0526 70                 MOV     M,B             ; Replace it
0527 CD1602             CALL    Bufchk          ; Check the buffers

052A E1                 POP     H               ; Get the registers back
052B C1                 POP     B

052C C9                 RET

                ;*****************************************************************************
                ;*                          SPECIAL BUFFER ACCESS ROUTINES                   *
                ;*****************************************************************************

                ; These routines are for taking one byte from the buffer, but as addressed
                ; only by the fill pointer. (The last byte in) The fill pointer is the only
                ; pointer updated. CAUTION: be careful where you use these routines! You could
                ; easily cause massive data corruption if you're not!

                ; Take a byte from the IEEE-488 input buffer

052D C5         Tkieia  PUSH    B               ; Save the registers
052E E5                 PUSH    H

052F 3A023C             LDA     Iibbc           ; Get the byte count
0532 3D                 DCR     A               ; Decrement it
0533 FA6105             JM      Cr              ; Git if MT
0536 32023C             STA     Iibbc           ; Put it back
0539 CDB603             CALL    Ieifp           ; Address the buffer's filled byte
053C 2B                 DCX     H               ; Subtract 1 from the address
053D 7E                 MOV     A,M             ; Put the byte into the accumulator

053E 21003C             LXI     H,Iibfp         ; Get the address of the fill pointer
0541 46                 MOV     B,M             ; Put it into B
0542 05                 DCR     B               ; Decrement the pointer by 1
0543 70                 MOV     M,B             ; Replace it

0544 E1                 POP     H               ; Get the registers back
0545 C1                 POP     B

0546 C7                 RET
```

```
                ; Take a byte from the RS-232 input buffer

0547 C5         Tkrsia  PUSH    B               ; Save the registers
0548 E5                 PUSH    H

0549 3A083D             LDA     Ribbc           ; Get the byte count
054C 3D                 DCR     A               ; Decrement it
054D FA6105             JM      Cr              ; Git if MT
0550 32083D             STA     Ribbc           ; Put it back
0553 CD7603             CALL    Rsifp           ; Address the buffer's filled byte
0556 2B                 DCX     H               ; Subtract 1 from the address
0557 7E                 MOV     A,M             ; Put the byte into the accumulator

0558 21063D             LXI     H,Ribfp         ; Get the address of the fill pointer
055B 46                 MOV     B,M             ; Put it into B
055C 05                 DCR     B               ; Decrement the pointer by 1
055D 70                 MOV     M,B             ; Replace it

055E E1                 POP     H               ; Get the registers back
055F C1                 POP     B

0560 C9                 RET

0561 3E00       Cr      MVI     A,00H           ; Return a null unit

0563 E1                 POP     H               ; Get the registers back
0564 C1                 POP     B
0565 C9                 RET

                ;*******************************************************************************
                ;*                    INPUT CHARACTER "CLEANING" ROUTINE                       *
                ;*******************************************************************************

                ; This routine takes the last character in the input buffer, and does this:

                ;       a.) It looks to see if it is greater than a space. If not, it will
                ;           "throw out" the byte.

                ;       b.) It looks to see if the character is greater than, or equal to a
                ;           "rubout". If it is not, it is kept. Otherwise, the byte is
                ;           "thrown out".

                ;       c.) It looks to see if the character is a lowercase letter. If it is,
                ;           it capitalizes the letter.

0566 F5         Cln488  PUSH    PSW             ; Save the registers
0567 C5                 PUSH    B
0568 D5                 PUSH    D
0569 E5                 PUSH    H

056A CD2D05     Look    CALL    Tkiela          ; Get the byte
056D CD8805             CALL    Spcspn          ; Clean it
0570 D48504             CNC     Adiei           ; Put it back

0573 C38305             JMP     Retoin          ; Blow
```

```
0576 F5          Cln232    PUSH    PSW             ; Save the registers
0577 C5                    PUSH    B
0578 D5                    PUSH    D
0579 E5                    PUSH    H

057A CD4705      Lookit    CALL    Tkrsia          ; Get the byte
057D CDB805                CALL    Spcspn          ; Clean it
0580 D42F04                CNC     Adrsi           ; Put it back

0583 E1          Retoin    POP     H               ; Restore the registers
0584 D1                    POP     D
0585 C1                    POP     B
0586 F1                    POP     PSW

0587 C9                    RET                     ; Nightey-night

0588 47          Spcspn    MOV     B,A             ; Save a copy of the byte
0589
0589 3A993E                LDA     Systat          ; Get the system status byte
058C E640                  ANI     Pastru          ; Is the system in pass-thru?
058E CA9505                JZ      Noway           ; If not, continue

0591 78                    MOV     A,B             ; Get the byte again
0592 C3AC05                JMP     Noefct          ; Leave it alone

0595 78          Noway     MOV     A,B             ; Get the byte again
0596 FE21                  CPI     21H             ; Compare it with "!"
0598 DAAF05                JC      Delete          ; If it is lower, delete it
059B FE7E                  CPI     7EH             ; Compare it with a "~"
059D D2AF05                JNC     Delete          ; Delete it if it is higher
05A0 FE61                  CPI     61H             ; Compare it with "a"
05A2 DAAC05                JC      Noefct          ; Return to caller if it is less than
05A5 FE7B                  CPI     7BH             ; Compare it with "{"
05A7 D2AC05                JNC     Noefct          ; Return if it is higher or equal to
05AA D620                  SUI     20H             ; Convert to upper case
05AC
05AC 37          Noefct    STC                     ; Set the carry
05AD 3F                    CMC                     ; Clear the carry

05AE C9                    RET                     ; Go away

05AF 37          Delete    STC                     ; Set the carry flag (indicates no-go)

05B0 C9                    RET                     ; Go back to jail

                 ;*****************************************************************************
                 ;*                          PROCESSING SUBROUTINES                           *
                 ;*****************************************************************************

                 ; These routines are the "frame" routines which initiate input processing.
                 ; They determine whether or not the input is to be "passed-through", or if it
                 ; should be treated as a command.

05B1 3A993E      Frame     LDA     Systat          ; Get the system status byte
```

```
05B4 E680               ANI     Ptproc          ; Check to see if this is "Pass-thru"
05B6 CABD05             JZ      Dwon            ; Skip the next call if it's not

05B9 CDE405             CALL    Pt488           ; Call the 488 pass-through

05BC C9                 RET

05BD 3A993E     Dwon    LDA     Systat          ; Get the status byte again
05C0 47                 MOV     B,A             ; Save it in B
05C1 3E20               MVI     A,Act232        ; Get the 232 active flag
05C3 2F                 CMA                     ; Complement it
05C4 A0                 ANA     B               ; AND it into A
05C5 32993E             STA     Systat          ; Put it back
05C8 CD1806             CALL    Prcss           ; Process the input

05CB C9                 RET

05CC 3A993E     Pr232   LDA     Systat          ; Get the system status byte
05CF E680               ANI     Ptproc          ; Check to see if this is "Pass-thru"
05D1 CAD805             JZ      Tue             ; Do it To it

05D4 CDFA05             CALL    Pt232           ; Call the 232 pass-through

05D7 C9                 RET

05D8 3A993E     Tue     LDA     Systat          ; Get the status byte again
05DB F620               ORI     Act232          ; Set the 232 active flag
05DD 32993E             STA     Systat          ; Put it back
05E0 CD1806             CALL    Prcss           ; Process the input

05E3 C9                 RET

                ;*****************************************************************************
                ;*                          PASS-THROUGH HANDLERS                            *
                ;*****************************************************************************

                ; These routines just transfer the contents of one input buffer, to the output
                ; buffer of the opposite interface. If the output buffer is the 488 buffer, an
                ; opportunity to assert SRQ is provided.

                ; IEEE-488 in -> RS-232 out

05E4 3A023C     Pt488   LDA     Iibbc           ; Get the byte count
05E7 FE00               CPI     00H             ; Is it 0?
05E9 CAF505             JZ      Snd232          ; Start RS-232 output if all bytes done
05EC CDB904             CALL    Tkiei           ; Get a byte from the 488 input buffer
05EF CD1504             CALL    Adrso           ; Put it into the 232 output buffer
05F2 C3E405             JMP     Pt488           ; Loop until all gone

05F5 3E03       Snd232  MVI     A,Recda+Tbei    ; Enable transmit interrupts
05F7 D339               OUT     Int2            ; Place in interrupt buffer

05F9 C9                 RET

                ; RS-232 in -> IEEE-488 out
```

```
05FA 3A083D     Pt232   LDA     Ribbc           ; Get the byte count
05FD FE00               CPI     00H             ; Is it 0?
05FF CA0B06             JZ      Asserv          ; Return if all bytes done
0602 CD4904             CALL    Tkrsi           ; Get a byte from the 232 input buffer
0605 CD9F04             CALL    Adieo           ; Put it into the 488 output buffer
0608 C3FA05             JMP     Pt232           ; Loop until all gone

060B 3A973E     Asserv  LDA     Status          ; Load the GPIB status byte
060E 0602               MVI     B,Serl          ; Get the serial data output mask
0610 B0                 ORA     B               ; OR it into the status byte
0611 32973E             STA     Status          ; Replace the byte in RAM
0614 CDD20A             CALL    Srq_do          ; If legal, assert SRQ

0617 C9                 RET

                ;*****************************************************************************
                ;*                          COMMAND PROCESSING                               *
                ;*****************************************************************************

                ; These routines take the contents of the specified input buffer, and transfers
                ; them to the character buffer one command at a time (delimited by commas or
                ; semicolons). Then a generic (not input interface specific) routine is called
                ; that processes the command.

0618 21993E     Prcss   LXI     H,Systat        ; Get the address of the system status
061B 7E                 MOV     A,M             ; Bring it into the accumulator
061C E620               ANI     Act232          ; Check for 232 buffer active
061E C25106             JNZ     Rstr            ; If so, go somewhere else

                ; Transfer 488 buffer to character buffer

0621 0609       Lupe    MVI     B,09H           ; Maximum of 8 characters per command
0623 CDA702             CALL    Clchbf          ; Clear the character buffer
0626 05         Dek0    DCR     B               ; Decrement the byte count
0627 CA4506             JZ      Bye             ; Process the command if 0
062A
062A 3A023C             LDA     Iibbc           ; Check the byte count
062D FE00               CPI     00H             ; Is it 0?
062F CA4506             JZ      Bye             ; Cruise if so

0632 CDB904             CALL    Tkiei           ; Get a character from the 488 buffer

0635 FE3B               CPI     3BH             ; Is it a semicolon?
0637 CA4506             JZ      Bye             ; If so, go check the command

063A FE2C               CPI     2CH             ; Is it a comma?
063C CA4506             JZ      Bye             ; If so, go check the command

063F CDF504             CALL    Adch            ; Add it to the character buffer
0642 C32106             JMP     Dek0            ; Loop

0645 CD6A06     Bye     CALL    Comnd           ; Go process the command
0648 3A023C             LDA     Iibbc           ; Get the 488 byte count
064B FE00               CPI     00H             ; Check to see if it's 0
```

```
064D C8                 RZ
064E C32106             JMP     Lupe            ; Return if it is
                                                ; Get next command

                ; Transfer 232 buffer to character buffer

0651 0609       Rstr    MVI     B,09H           ; Maximum of 8 characters per command
0653 CDA702             CALL    Clchbf          ; Clear the character buffer
0656 05         Dek     DCR     B               ; Decrement the byte count
0657 CA7A06             JZ      Proc            ; Go process the input if 0

065A 3A083D             LDA     Ribbc           ; Check the byte count
065D FE00               CPI     00H             ; Is sit 0?
065F CA7A06             JZ      Proc            ; Git if it is
0662 CD4904             CALL    Tkrsi           ; Get a character from the 232 buffer

0665 FE3B               CPI     3BH             ; Is it a semicolon?
0667 CA7A06             JZ      Proc            ; If so, go check the command

066A FE2C               CPI     2CH             ; Is it a comma?
066C CA7A06             JZ      Proc            ; If so, go check the command

066F FE00               CPI     00H             ; See if it's a null unit
0671 CA7A06             JZ      Proc            ; Cruise if it is

0674 CDF504             CALL    Adch            ; Add it to the character buffer
0677 C35606             JMP     Dek             ; Loop

067A CD8606     Proc    CALL    Comnd           ; Go process the command
067D 3A083D             LDA     Ribbc           ; Get the 232 byte count
0680 FE00               CPI     00H             ; Check to see if it's 0
0682 C8                 RZ                      ; Return if it is
0683 C35106             JMP     Rstr            ; Get next command

                ;*****************************************************************************
                ;*                          MAIN COMMAND PROCESSOR                           *
                ;*****************************************************************************

                ; This routine checks to see if the input command matches any of the listed
                ; command strings as outlined in the lookup tables. If it does, the process
                ; associated with that command is carried out. If no commands match, then the
                ; error handler is called, and processing is aborted.

0686 3A0E3E     Comnd   LDA     Chbbc           ; Get char. buff. byte count
0689 FE00               CPI     0H              ; Check it for 0
068B C8                 RZ                      ; Null process if no command

068C 21993E             LXI     H,Systat        ; Get the address of the system status
068F 3E10               MVI     A,Nocmnd        ; Bring in the "no match" mask
0691 B6                 ORA     M               ; OR it into the byte
0692 77                 MOV     M,A             ; Put the updated byte back

0693 CDA506             CALL    Syscom          ; Check for system command match
0696 7E                 MOV     A,M             ; Get the system status again
0697 E610               ANI     Nocmnd          ; Check to see if a command matched
0699 C8                 RZ                      ; Return if it did
```

```
069A CDBC06             CALL    Patcom          ; Check for path command match
069D 7E                 MOV     A,M             ; Get the system status again
069E E610               ANI     Nocmnd          ; Check to see if a command matched
06A0 C8                 RZ                      ; Return if it did

06A1 CDB60A     Err     CALL    Error           ; Call the error handling routine

06A4 C9                 RET                     ; Get outta here

                ;*****************************************************************************
                ;*                          CHECK FOR SYSTEM COMMANDS                        *
                ;*****************************************************************************

                ; This routine loads the base address of the system command string match table
                ; into the H/L registers, then calls the generic string match subroutine. If a
                ; match occurs, the "Nocmd" flag is cleared, and the instruction number that is
                ; returned in the accumulator is processed.

06A5 F5         Syscom  PUSH    PSW             ; Save the registers
06A6 C5                 PUSH    B
06A7 D5                 PUSH    D
06A8 E5                 PUSH    H

06A9 21170B             LXI     H,System        ; Load the base address for system strs
06AC CDD306             CALL    Comtch          ; Look for an implicit command match

06AF 3A993E             LDA     Systat          ; Get the system status again
06B2 E610               ANI     Nocmnd          ; Check to see if a command matched
06B4 CC2307             CZ      Sysdo           ; Execute it if it did

06B7 E1                 POP     H               ; Get the registers back
06B8 D1                 POP     D
06B9 C1                 POP     B
06BA F1                 POP     PSW

06BB C9                 RET

                ;*****************************************************************************
                ;*                          CHECK FOR PATH COMMANDS                          *
                ;*****************************************************************************

06BC F5         Patcom  PUSH    PSW             ; Save the registers
06BD C5                 PUSH    B
06BE D5                 PUSH    D
06BF E5                 PUSH    H

06C0 215C0B             LXI     H,Pathst        ; Load the base address for paths
06C3 CDD306             CALL    Comtch          ; Look for an implicit command match

06C6 3A993E             LDA     Systat          ; Get the system status again
06C9 E610               ANI     Nocmnd          ; Check to see if a command matched
06CB CC4B0B             CZ      Patdo           ; Execute it if it did

06CE E1                 POP     H               ; Get the registers back
```

```
06CF D1                 POP     D
06D0 C1                 POP     B
06D1 F1                 POP     PSW

06D2 C9                 RET

                ;*****************************************************************************
                ;*                          LOOK FOR A STRING MATCH                          *
                ;*****************************************************************************

                ; This routine goes through the character buffer, one byte at a time, and
                ; compares each byte to it's corresponding byte in the specified lookup table.
                ; if a command makes it all the way through, the "Nocmnd" flag is cleared.
                ; Otherwise, the "Nocmnd" flag is set. If there is a match, the path number is
                ; returned in the accumulator.

06D3 3E00       Comtch  MVI     A,00H           ; Clear the path number holder
06D5 F5                 PUSH    PSW             ; Save it in the stack

06D6 7E         Nextpt  MOV     A,M             ; Bring the length count into A
06D7 23                 INX     H               ; Increment the pointer
06D8 FE00               CPI     00H             ; See if it's 0
06DA CAF006             JZ      Nomo            ; Scram if it is

06DD 4F                 MOV     C,A             ; Store it in C
06DE CDF306             CALL    Setmct          ; Check for a match
06E1 3A993E             LDA     Systat          ; Get the system status byte
06E4 E610               ANI     Nocmnd          ; Check for command valid
06E6 CAF006             JZ      Nomo            ; All right!

06E9 F1         Go_on   POP     PSW             ; Get the path number indicator
06EA C602               ADI     02H             ; Increment it by two
06EC F5                 PUSH    PSW             ; Put it back in the stack

06ED C3D606             JMP     Nextpt          ; Loop

06F0 F1         Nomo    POP     PSW             ; Get command number
06F1 47                 MOV     B,A             ; Save it in B

06F2 C9                 RET
06F3
                ; Defaults to command match

06F3 3E10       Setmct  MVI     A,Nocmnd        ; Get the No command mask
06F5 2F                 CMA                     ; Complement it
06F6 47                 MOV     B,A             ; Save it in B
06F7 3A993E             LDA     Systat          ; Get the system status byte
06FA A0                 ANA     B               ; AND it with the complemented mask
06FB 32993E             STA     Systat          ; Put it back

06FE 110F3E             LXI     D,Chbase        ; Load the base addr of the char buf

0701 0D         Geel    DCR     C               ; Decrement the count
0702 FA2207             JM      Yup             ; Yeah! Made it through!
```

```
0705 46                 MOV     B,M             ; Bring in the byte from the table
0706 23                 INX     H               ; Increment the pointer
0707 EB                 XCHG                    ; Exchange address registers
0708 7E                 MOV     A,M             ; Get the next character in the buffer
0709 23                 INX     H               ; Increment the buffer pointer
070A EB                 XCHG                    ; Re-exchange address registers

070B B8                 CMP     B               ; Compare the two bytes
070C C21207             JNZ     Nope            ; Break loop if they are not the same

070F C30107             JMP     Geek            ; Keep going, we're not through yet!

0712 3A993E     Nope    LDA     Systat          ; Get system status byte
0715 F610               ORI     Nocmnd          ; OR in the "not found" flag
0717 32993E             STA     Systat          ; Keep tryin'

071A 0D         Rfe     DCR     C               ; Continue to decrement the count
071B FA2207             JM      Yup             ; Break loop when through
071E 23                 INX     H               ; Continue incrementing pointer
071F C31A07             JMP     Rfe             ; Keep goin'

0722 C9         Yup     RET                     ; Go back

                ;*****************************************************************************
                ;*                          EXECUTE SYSTEM COMMANDS                          *
                ;*****************************************************************************

0723 78         Sysdo   MOV     A,B             ; Get the path number

0724 FE00       Clr     CPI     00H             ; Check for "CLR" command
0726 CA0100             JZ      Clrin           ; Restart if so

0729 FE02       Clp     CPI     02H             ; Check for "CLP" command
072B C23207             JNZ     Clc             ; Skip the next commands if it is not
072E CDF202             CALL    Dfault          ; Call the default path routine
0731 C9                 RET                     ; Goodbye!

0732 FE04       Clc     CPI     04H             ; Check for "CLC" command
0734 C23B07             JNZ     Ptm             ; Skip the next routine if it is not
0737 CD1603             CALL    Defsys          ; Clear comm ports
073A C9                 RET

073B FE06       Ptm     CPI     06H             ; Check for "PTM" command
073D C25507             JNZ     Rsb             ; Skip if not
0740 3A993E             LDA     Systat          ; Get the system status byte
0743 0640               MVI     B,Pastru        ; Get the pass-thru mask
0745 B0                 ORA     B               ; OR it into the byte
0746 32993E             STA     Systat          ; Put the byte back
0749 3A773E             LDA     Status          ; Get the GPIB status byte
074C F640               ORI     Pstr            ; OR in the pass-thru indicator
074E 32773E             STA     Status          ; Put the byte back
0751 CDD206             CALL    Srq_do          ; If legal, assert SRQ
0754 C9                 RET

0755 FE08       Rsb     CPI     08H             ; Check for "RSB" command
```

```
0757 C27A07             JNZ     Rbm             ; Skip if not
075A CDA702             CALL    Clchbf          ; Clear the character buffer
075D 3A973E             LDA     Status          ; Get the status byte
0760 CDEA0B             CALL    Bn_bcd          ; Call binary-to-bcd conversion routine
0763 3E0D               MVI     A,0DH           ; Carriage return
0765 CDF504             CALL    Adch            ; Put it into the output
0768 CD3A09             CALL    Outgo           ; Call the output routine
076B 3A943E             LDA     Byte5           ; Get the front panel byte
076E 47                 MOV     B,A             ; Store it in B
076F 3E0C               MVI     A,Srqon+Erron   ; Get the SRQ and error lamp masks
0771 2F                 CMA                     ; Complement them
0772 A0                 ANA     B               ; AND in the old byte
0773 32943E             STA     Byte5           ; Put the byte back
0776 CD4C03             CALL    Setprt          ; Turn off the lamp(s)
0779 C9                 RET

077A FE0A       Rbm     CPI     0AH             ; Check for "RBM" command
077C C2A407             JNZ     Rps             ; Skip if not
077F CDA702             CALL    Clchbf          ; Clear the character buffer
0782 3A8F3E             LDA     Byte0           ; Get the first bit map byte
0785 CDF504             CALL    Adch            ; Put it into the character buffer
0788 3A903E             LDA     Byte1           ; Get the second bit map byte
078B CDF504             CALL    Adch            ; Put it into the character buffer
078E 3A913E             LDA     Byte2           ; Ditto for third...
0791 CDF504             CALL    Adch
0794 3A923E             LDA     Byte3
0797 CDF504             CALL    Adch
079A 3A933E             LDA     Byte4
079D CDF504             CALL    Adch

07A0 CD3A09             CALL    Outgo           ; Call the output routine
07A3 C9                 RET

07A4 FE0C       Rps     CPI     0CH             ; Check for "RPS" command
07A6 C2B307             JNZ     Ieo             ; Skip if not
07A9 CDA702             CALL    Clchbf          ; Clear character buffer
07AC CD8309             CALL    Getset          ; Put the setup into the buffer
07AF CD3A09             CALL    Outgo           ; Output it
07B2 C9                 RET

07B3 FE0E       Ieo     CPI     0EH             ; Check for the "IEO" command
07B5 C2C607             JNZ     Rso             ; Skip if not
07B8 3A993E             LDA     Systat          ; Get the system status byte
07BB F604               ORI     Ieomsk          ; OR the IEEE-488 output mask in
07BD 47                 MOV     B,A             ; Put the accumulator into B
07BE 3E08               MVI     A,Rsomsk        ; Get the RS-232 output mask
07C0 2F                 CMA                     ; Complement it
07C1 A0                 ANA     B               ; AND it into the accumulator
07C2 32993E             STA     Systat          ; Put the new system status back
07C5 C9                 RET

07C6 FE10       Rso     CPI     10H             ; Check for the "RSO" command
07C8 C2D907             JNZ     Bot             ; Skip if not
07CB 3A993E             LDA     Systat          ; Get the system status byte
07CE F608               ORI     Rsomsk          ; OR the RS-232 output mask in
```

```
07D0 47                 MOV     B,A             ; Put the accumulator into B
07D1 3E04               MVI     A,Ieomsk        ; Get the IEEE-488 output mask
07D3 2F                 CMA                     ; Complement it
07D4 A0                 ANA     B               ; AND it into the accumulator
07D5 32993E             STA     Systat          ; Put the new system status back
07D8 C9                 RET

07D9 FE12       Bot     CPI     12H             ; Check for the "BOT" command
07DB C2E907             JNZ     Sts             ; Skip if not
07DE 3A993E             LDA     Systat          ; Get the system status byte
07E1 F608               ORI     Rsomsk          ; OR in the RS-232 mask
07E3 F604               ORI     Ieomsk          ; OR in the IEEE-488 mask
07E5 32993E             STA     Systat          ; Put the system status byte back
07E8 C9                 RET

07E9 FE14       Sts     CPI     14H             ; Check for "STS" command
07EB C2F507             JNZ     Ser             ; Skip if not
07EE CD9508             CALL    Bcd_bn          ; Get the input number
07F1 32983E             STA     Srqmsk          ; Store the result in the SRQ mask
07F4 C9                 RET

07F5 FE16       Ser     CPI     16H             ; Check for "SER" command
07F7 C20208             JNZ     Bdr             ; Skip if not
07FA CD9508             CALL    Bcd_bn          ; Get the input number
07FD E67F               ANI     7FH             ; Make sure that the divisor isn't enab
07FF D33B               OUT     Linctl          ; Send it to the chip
0801 C9                 RET

0802 FE18       Bdr     CPI     18H             ; Check for "BDR" command
0804 C20B08             JNZ     Stm             ; Skip if not
0807 CD8F0A             CALL    Set_br          ; Set the Baud rate
080A C9                 RET

080B FE1A       Stm     CPI     1AH             ; Check for "STM" command
080D C21708             JNZ     Rad             ; Skip if not
0810 CD9508             CALL    Bcd_bn          ; Get the input number
0813 32953E             STA     Sertrm          ; Store the number of the ch in RAM
0816 C9                 RET

0817 FE1C       Rad     CPI     1CH             ; Check for "RAD" command
0819 C22E08             JNZ     Adr             ; Skip if not
081C CDA702             CALL    Clchbf          ; Clear the character buffer
081F 3A9A3E             LDA     Adgpib          ; Get the address
0822 CDEA0B             CALL    Bn_bcd          ; Put it into the character buffer
0825 3E0D               MVI     A,0DH           ; Carriage return
0827 CDF504             CALL    Adch            ; Put it into the output
082A CD3A09             CALL    Outgo           ; Output it
082D C9                 RET

082E FE1E       Adr     CPI     1EH             ; Check for "ADR" command
0830 C23B08             JNZ     Att             ; Skip if not
0833 CD9508             CALL    Bcd_bn          ; Get the new address
0836 E61F               ANI     1FH             ; AND out the address (0-31)
0838 D374               OUT     Addreg          ; Send it to the address register
083A 329A3E             STA     Adgpib          ; Save it in RAM
```

```
083D C9                 RET

083E CD9508     Att     CALL    Bcd_bn          ; Get the attenuator value
0841 32933E             STA     Byte4           ; Store the set value
0844 CD4C03             CALL    Setprt          ; Set the ports
0847 C9                 RET

                ;*****************************************************************************
                ;*                          PATH COMMAND EXECUTION                           *
                ;*****************************************************************************

                ; This routine takes the system command number passed in by the accumulator,
                ; and offsets it against the base address of the path code pointer table. This
                ; addresses a pointer that indicates the location of the start of the port set
                ; control codes for that particular command. The codes are then loaded in and
                ; deciphered, and the bit map is set accordingly. After the bit map has been
                ; set, the routine to set the ports is executed.

0848 78         Patdo   MOV     A,B             ; Get the path number
0849 21E60C             LXI     H,Patptr        ; Set pointer to code table ptr base
084C 85                 ADD     L               ; Add the command number to L
084D 6F                 MOV     L,A             ; Store it into L
084E D25208             JNC     Xcute           ; Skip H increment if no carry
0851 24                 INR     H               ; Increment H if carry

0852 5E         Xcute   MOV     E,M             ; Get the LSB of code address
0853 23                 INX     H               ; Increment memory pointer
0854 56                 MOV     D,M             ; Get the MSB of code address
0855 EB                 XCHG                    ; Set the new pointer

0856 118F3E     Getcod  LXI     D,Byte0         ; Load the base address of bit map
0859 7E                 MOV     A,M             ; Get the code byte
085A 47                 MOV     B,A             ; Keep the original in B
085B E638               ANI     38H             ; AND out the byte address
085D 0F                 RRC                     ; Shift the address over 3 places
085E 0F                 RRC
085F 0F                 RRC
0860 83                 ADD     E               ; Add it to E
0861 5F                 MOV     E,A             ; Store it in E
0862 D26608             JNC     Ghi             ; Skip increment if no carry
0865 14                 INR     D               ; Increment MSB of byte address

0866 78         Ghi     MOV     A,B             ; Get the original byte again
0867 E607               ANI     07H             ; AND out the bit address
0869 4F                 MOV     C,A             ; Store it in C
086A 3E01               MVI     A,01H           ; Set the accumulator to 1

086C 0D         Rptr    DCR     C               ; Decrement C
086D FA7408             JM      Escp            ; Break loop if result is negative
0870 07                 RLC                     ; Rotate one place left
0871 C36C08             JMP     Rptr            ; Continue loop

0874 4F         Escp    MOV     C,A             ; Store the result in C
0875 EB                 XCHG                    ; Exchange address pointers
0876 78                 MOV     A,B             ; Bring in the original byte again
```

```
0877 E680                ANI     80H             ; AND out the last bit
0879 C28308              JNZ     Setbit          ; Skip the next routine if not clear

087C 79          Clrbit  MOV     A,C             ; Bring the parsed bit into A
087D 2F                  CMA                     ; Complement the byte
087E A6                  ANA     M               ; AND the code bit into bit map byte
087F 77                  MOV     M,A             ; Put the bit map byte back
0880 C38608              JMP     Exiter          ; Go on

0883 79          Setbit  MOV     A,C             ; Bring in the parsed bit
0884 B6                  ORA     M               ; OR it into the bit map byte
0885 77                  MOV     M,A             ; Put it into the bit map

0886 EB          Exiter  XCHG                    ; Re-exchange the address registers
0887 78                  MOV     A,B             ; Get the original byte again
0888 E640                ANI     40H             ; Check the "Last code" bit
088A C29108              JNZ     Setter          ; Break loop if it's set
088D 23                  INX     H               ; Increment the address registers
088E C33608              JMP     Getcod          ; Continue setting the bit map

0891 CD4C03      Setter  CALL    Setprt          ; Set the ports
0894 C9                  RET                     ; Return

;*****************************************************************************
;*                        BCD/BINARY CONVERSION ROUTINES                     *
;*****************************************************************************

; This routine goes through the character buffer from the beginning on, and
; looks for numeric characters. It then processes the first numeric character
; and each one after that until it comes upon the first non-numeric character
; or the end of the string. The result is returned in the accumulator. If the
; result is greater than 255, it is handled as an error.

0895 C5          Bcd_bn  PUSH    B               ; Save registers
0896 D5                  PUSH    D
0897 E5                  PUSH    H

0898 210F3E              LXI     H,Chbase        ; Get the base address of char. buff
089B 0600                MVI     B,0             ; Put 0 into the "Collector"
089D 3A0C3E              LDA     Chbfp           ; Get the character buffer fill ptr
08A0 5F                  MOV     E,A             ; Save it in E as a character count
08A1 3A993E              LDA     Systat          ; Get the system status byte
08A4 F601                ORI     Scrtch          ; Set a flag to indicate no numerics
08A6 32993E              STA     Systat          ; Put it back

08A9 7E          Do_it   MOV     A,M             ; Get the pointed character in buffer
08AA 23                  INX     H               ; Increment the pointer

08AB D630                SUI     30H             ; "Normalize" it
08AD FAB208              JM      Oops            ; Go away if non-numeric
08B0 FE0A                CPI     0AH             ; Compare it to 10
08B2 D2B208              JNC     Oops            ; Go away if non-numeric

08B5 4F                  MOV     C,A             ; Copy it into C
08B6 78                  MOV     A,B             ; Get previous result
```

```
08B7 FE00               CPI     00H             ; Compare it to 0
08B9 CAC308             JZ      Fgs             ; Skip multiply if 0
08BC 1609               MVI     D,09H           ; Place multiplier into D

08BE 80         Mu      ADD     B               ; Add the previous result to itself
08BF 15                 DCR     D               ; 10 times
08C0 C2BE08             JNZ     Mu              ; Keep goin'

08C3 81         Fgs     ADD     C               ; Add the present digit to the result
08C4 DAB60A             JC      Error           ; If there is a carry, there is an err.
08C7 47                 MOV     B,A             ; Put it into the "Collector"

08C8 3E01               MVI     A,Scrtch        ; Get the Numeric flag
08CA 2F                 CMA                     ; Complement it
08CB 4F                 MOV     C,A             ; Store it in C
08CC 3A993E             LDA     Systat          ; Get the system status byte
08CF A1                 ANA     C               ; AND in the mask
08D0 32993E             STA     Systat          ; Put the byte back
08D3 C3DE08             JMP     Fgsdi           ; Go keep counting

08D6 3A993E     Oops    LDA     Systat          ; Get the system status byte
08D9 E601               ANI     Scrtch          ; See if the numeric character flag is
08DB CAE508             JZ      Reslt           ; Blow town if it is

08DE 1D         Fgsdi   DCR     E               ; Decrement the character count
08DF FAE508             JM      Reslt           ; Deliver result if all done
08E2 C3A908             JMP     Do_it           ; Go back to loop

08E5 78         Reslt   MOV     A,B             ; Put the result into the accumulator

08E6 E1                 POP     H               ; Retrieve registers
08E7 D1                 POP     D
08E8 C1                 POP     B

08E9 C9                 RET                     ; Return to caller

                ;*****************************************************************************
                ;*                          BINARY TO BCD ROUTINE                            *
                ;*****************************************************************************

                ; This routine takes the number passed in through the accumulator, and puts
                ; it's ASCII equivalent into the character buffer. Leading 0's are ignored.
                ; The value cannot, of course, be greater than 255.

08EA F5         Bn_bcd  PUSH    PSW             ; Save the registers
08EB C5                 PUSH    B
08EC D5                 PUSH    D
08ED E5                 PUSH    H

08EE 1600               MVI     D,00H           ; Clear result locations
08F0 0E00               MVI     C,00H
08F2 0600               MVI     B,00H

08F4 FE64       Hund    CPI     64H             ; Compare with 100
08F6 DAFF08             JC      Tens            ; If less than 100, skip subtract
```

```
08F9 D664                       SUI     64H             ; Subtract 100
08FB 04                         INR     B               ; Increment 100's register
08FC C3F408                     JMP     Hund            ; Repeat until dun

08FF FE0A       Tens            CPI     0AH             ; Compare with 10
0901 DA0A09                     JC      Ones            ; If less than 10, skip subtract
0904 D60A                       SUI     0AH             ; Subtract 10
0906 0C                         INR     C               ; Increment the 10's register
0907 C3FF08                     JMP     Tens            ; Keep it up

090A FE01       Ones            CPI     01H             ; Compare it with 1
090C DA1509                     JC      Dunnit          ; Skip subtract if less than
090F D601                       SUI     01H             ; Subtract 1
0911 14                         INR     D               ; Increment the 1's register
0912 C30A09                     JMP     Ones            ; Keep going

0915 78         Dunnit          MOV     A,B             ; Get the MSD
0916 FE00                       CPI     00H             ; Check for 0
0918 CA2409                     JZ      Md              ; Skip if 0
091B C630                       ADI     30H             ; Normalize it to ASCII
091D CDF504                     CALL    Adch            ; Add it to the character buffer
0920 79                         MOV     A,C             ; Get next digit
0921 C32A09                     JMP     Jrk             ; Skip 0 check for middle digit

0924 79         Md              MOV     A,C             ; Get middle digit
0925 FE00                       CPI     00H             ; Check for 0
0927 CA2F09                     JZ      Ld              ; Skip if 0

092A C630       Jrk             ADI     30H             ; Normalize it to ASCII
092C CDF504                     CALL    Adch            ; Add it to the character buffer

092F 7A         Ld              MOV     A,D             ; Get least significant digit
0930 C630                       ADI     30H             ; Normalize it to ASCII
0932 CDF504                     CALL    Adch            ; Add it to the character buffer

0935 E1                         POP     H               ; Retrieve registers
0936 D1                         POP     D
0937 C1                         POP     B
0938 F1                         POP     PSW

0939 C9                         RET                     ; Go back
```

```
;*****************************************************************************
;*                          SELECTIVE OUTPUT ROUTINE                         *
;*****************************************************************************

; This routine determines which of the interfaces is enabled as a reply channel
; for system replies. It then transfers the contents of the character buffer to
; the output buffer of the designated port. After doing this, it enables the
; output interrupts for that port, and returns to the caller.

093A 210F01     Outgo           LXI     H,Chbbc         ; Get the address of the byte count
093D 46                         MOV     B,M             ; Get the character buffer byte count
093E 04                         INR     B               ; Increment it by 1
093F 210E01                     LXI     H,Chbase        ; Get the base address of the char buff
```

```
0942 3A993E             LDA     Systat          ; Get the system status byte
0945 E604               ANI     Ieomsk          ; Check to see if 488 is valid output
0947 CA6809             JZ      Rsdoit          ; Skip next buffer transfer if not

094A 3A973E     Iedoit  LDA     Status          ; Get GPIB status byte
094D F604               ORI     Replie          ; OR in the reply ready bit
094F 32973E             STA     Status          ; Replace the status byte

0952 210F3E             LXI     H,Chbase        ; Get the base address of the ch bf
0955 3A0E3E             LDA     Chbbc           ; Get the byte count
0958 47                 MOV     B,A             ; Save a copy in B

0959 05         Fte     DCR     B               ; Decrement it
095A FA6509             JM      X_s_1           ; Break loop if awdun
095D 7E                 MOV     A,M             ; Get the next byte
095E CD9F04             CALL    Adieo           ; Put the character into the 488 buffer
0961 23                 INX     H               ; Increment character buffer address
0962 C35909             JMP     Fte             ; Continue loop

0965 CDD20A     X_s_1   CALL    Srq_do          ; If legal, assert SRQ

0968 3A993E     Rsdoit  LDA     Systat          ; Get the system status byte
096B E60B               ANI     Rsomsk          ; Check to see if 232 is valid output
096D C8                 RZ                      ; Skip next buffer transfer if so

096E 3A0E3E     Ftf     LDA     Chbbc           ; Get the character buffer byte count
0971 FE00               CPI     00H             ; Compare it to 0
0973 CA7F09             JZ      X_s_r           ; Break loop if awdun
0976 CD0F05             CALL    Tkch            ; Get a byte from the character buffer
0979 CD1504             CALL    Adrso           ; Put the character into the 232 buffer
097C C36E09             JMP     Ftf             ; Continue loop

097F CDF505     X_s_r   CALL    Snd232          ; Set up output enables

0982 C9                 RET                     ; Beat it

                ;*************************************************************************
                ;*            ROUTINE TO READ THE PATH SETUP INTO CHARACTER BUFFER        *
                ;*************************************************************************

                ; This routine reads the bit map, and forms an easily readable ASCII reply.
                ; The first part of the reply indicates which relay control line numbers are
                ; in an "active" state. The second part returns the attenuator vale in dB, and
                ; the third part returns the status of the 5 external control lines.

0983 CDA702     Getset  CALL    Clchbf          ; Clear the character buffer
0986 3E4C               MVI     A,4CH           ; Put header into character buffer
0988 CDF504             CALL    Adch            ; Put in the letter "L"
098B 3E49               MVI     A,49H           ; "I"
098D CDF504             CALL    Adch
0990 3E4E               MVI     A,4EH           ; "N"
0992 CDF504             CALL    Adch
0995 3E45               MVI     A,45H           ; "E"
0997 CDF504             CALL    Adch
```

```
099A 3E53              MVI     A,53H           ; "S"
099C CDF504            CALL    Adch
099F 3E3A              MVI     A,3AH           ; ":"
09A1 CDF504            CALL    Adch
09A4 3E20              MVI     A,20H           ; " "
09A6 CDF504            CALL    Adch

09A9 218F3E            LXI     H,Byte0         ; Address the first bit map byte
09AC 0E01              MVI     C,01H           ; Initialize the path count

09AE 46        Bnb     MOV     B,M             ; Put bit map byte into the B register
09AF 1601              MVI     D,01H           ; Initialize the variable mask
09B1 CD750A            CALL    Vrnr            ; Check the byte
09B4 79                MOV     A,C             ; Get the line count
09B5 FE1C              CPI     1CH             ; Was this an early break?
09B7 CABE09            JZ      Attset          ; Check the attenuator if so
09BA 23                INX     H               ; Increment pointer
09BB C3AE09            JMP     Bnb             ; Continue to next byte

09BE 3E20      Attset  MVI     A,20H           ; Put a space into buffer
09C0 CDF504            CALL    Adch
09C3 3E2A              MVI     A,2AH           ; "*" (Indicates end of list)
09C5 CDF504            CALL    Adch
09C8 3E0D              MVI     A,0DH           ; Put carr. return into the accumulator
09CA CDF504            CALL    Adch            ; Store it
09CD 3E0A              MVI     A,0AH           ; Put a line feed into the accumulator
09CF CDF504            CALL    Adch            ; Put it into the character buffer
09D2 3E41              MVI     A,41H           ; Put "A" into the accumulator
09D4 CDF504            CALL    Adch            ; Add it to the character buffer
09D7 3E54              MVI     A,54H           ; Two "T"'s
09D9 CDF504            CALL    Adch
09DC CDF504            CALL    Adch
09DF 3E3D              MVI     A,3DH           ; "="
09E1 CDF504            CALL    Adch

09E4 3A933E            LDA     Byte4           ; Get attenuator value
09E7 CDEA0B            CALL    Bn_bcd          ; Add it to the character buffer
09EA 3E0D              MVI     A,0DH           ; Put carr. return into accumulator
09EC CDF504            CALL    Adch            ; Store it
09EF 3E0A              MVI     A,0AH           ; Put a line feed into the accumulator
09F1 CDF504            CALL    Adch            ; Add it to the character buffer
09F4 3E45              MVI     A,45H           ; Get "E"
09F6 CDF504            CALL    Adch            ; Put it into the character buffer
09F9 3E58              MVI     A,58H           ; "X"
09FB CDF504            CALL    Adch
09FE 3E54              MVI     A,54H           ; "T"
0A00 CDF504            CALL    Adch
0A03 3E3A              MVI     A,3AH           ; ":"
0A05 CDF504            CALL    Adch
0A08 3E20              MVI     A,20H           ; " "
0A0A CDF504            CALL    Adch

0A0D 3A923E            LDA     Byte3           ; Get the fourth bit map byte
0A10 47                MOV     B,A             ; Put it into B
0A11 0E01              MVI     C,01H           ; Start line count at 1
```

```
0A13 160B               MVI     D,08H           ; Start mask at 08
0A15 CD750A             CALL    Vrnr            ; Check lines

0A18 3E20               MVI     A,20H           ; Put a space into buffer
0A1A CDF504             CALL    Adch
0A1D 3E2A               MVI     A,2AH           ; "*" (Indicates end of list)
0A1F CDF504             CALL    Adch
0A22 3E0D               MVI     A,0DH           ; Carriage return
0A24 CDF504             CALL    Adch
0A27 3E0A               MVI     A,0AH           ; Line feed
0A29 CDF504             CALL    Adch

0A2C 3E53               MVI     A,53H           ; "S"
0A2E CDF504             CALL    Adch
0A31 3E54               MVI     A,54H           ; "T"
0A33 CDF504             CALL    Adch
0A36 3E4D               MVI     A,4DH           ; "M"
0A38 CDF504             CALL    Adch
0A3B 3E3D               MVI     A,3DH           ; "="
0A3D CDF504             CALL    Adch
0A40 3A953E             LDA     Sertrm          ; Get the serial termination character
0A43 CDEA08             CALL    Bn_bcd          ; Put it into the buffer
0A46 3E0D               MVI     A,0DH           ; Carriage return
0A48 CDF504             CALL    Adch
0A4B 3E0A               MVI     A,0AH           ; Line feed
0A4D CDF504             CALL    Adch

0A50 3E41               MVI     A,41H           ; "A"
0A52 CDF504             CALL    Adch
0A55 3E44               MVI     A,44H           ; "D"
0A57 CDF504             CALL    Adch
0A5A 3E52               MVI     A,52H           ; "R"
0A5C CDF504             CALL    Adch
0A5F 3E3D               MVI     A,3DH           ; "="
0A61 CDF504             CALL    Adch
0A64 3A9A3E             LDA     Adgpib          ; Get the GPIB address
0A67 CDEA08             CALL    Bn_bcd          ; Put it into the character buffer
0A6A 3E0D               MVI     A,0DH           ; Carriage return
0A6C CDF504             CALL    Adch
0A6F 3E0A               MVI     A,0AH           ; Line feed
0A71 CDF504             CALL    Adch

0A74 C9                 RET

0A75 7A         Vrnr    MOV     A,D             ; Put the mask into the accumulator
0A76 A0                 ANA     B               ; AND it with the byte
0A77 CA830A             JZ      Vby             ; Skip next procedure if not active
0A7A 79                 MOV     A,C             ; Get the line count
0A7B CDEA08             CALL    Bn_bcd          ; Add it to the character buffer
0A7E 3E2C               MVI     A,2CH           ; Put a comma into the accumulator
0A80 CDF504             CALL    Adch            ; Put it into the character buffer

0A83 0C         Vby     INR     C               ; Increment the path count
0A84 79                 MOV     A,C             ; Bring it into A
0A85 FE1C               CPI     1CH             ; See if all relay lines were checked
```

```
0AB7 C8                 RZ                              ; Return if so
0AB8 7A                 MOV     A,D                     ; Get the movable mask
0AB9 07                 RLC                             ; Rotate it one place left
0ABA 57                 MOV     D,A                     ; Put it back
0ABB D2750A             JNC     Vrnr                    ; Keep checking this byte if not thru
0ABE C9                 RET                             ; Return to caller

                ;*****************************************************************************
                ;*                       ROUTINE TO SET THE BAUD RATE                        *
                ;*****************************************************************************

                ; This routine simply uses the input number as an index to address the divisor
                ; table. It then loads the UART's divisor latch with the given codes.

0ABF CD9508     Set_br  CALL    Bcd_bn                  ; Get the code from the input
0AC2 FE12               CPI     12H                     ; Compare it to 18 (cannot be >17)
0AC4 D2B60A             JNC     Error                   ; It's a boo-boo if it's greater!
0AC7 87                 ADD     A                       ; Double it (2 bytes per code)
0AC8 11F30A             LXI     D,Baudtb                ; Get the base address of divisor table
0ACB 83                 ADD     E                       ; Add the code to the LSB
0ACC 5F                 MOV     E,A                     ; Put the LSB Back in E
0ACD D2D10A             JNC     Grab                    ; Skip the D-increment if no carry
0AD0 14                 INR     D                       ; Increment MSB

0AD1 0680       Grab    MVI     B,Dlab                  ; Mask to enable the divisor latch
0AD3 DB3B               IN      Linctl                  ; Get the current line control register
0AD5 B0                 ORA     B                       ; Set the latch enable
0AD6 D33B               OUT     Linctl                  ; Enable the divisor
0AD8 1A                 LDAX    D                       ; Get the MSB of the divisor byte
0AD9 D339               OUT     Dlmsb                   ; Put it into the MSB register
0ADB 13                 INX     D                       ; Increment the address by one
0ADC 1A                 LDAX    D                       ; Get the LSB
0ADD D338               OUT     Dllsb                   ; Store it
0ADF DB3B               IN      Linctl                  ; Load the contents of line control reg
0AE1 E67F               ANI     7FH                     ; AND out the last bit (Latch enable)
0AE3 D33B               OUT     Linctl                  ; Put the new value into the register
0AE5 C9                 RET

                ;*****************************************************************************
                ;*                        ROUTINE TO HANDLE ERRORS                           *
                ;*****************************************************************************

                ; This routine is a generic error handler. It turns on the front panel lamp,
                ; sets SRQ if allowed, and then forces the program to the interrupt wait.

0AE6 3A974E     Error   LDA     Byte5                   ; Get the front panel lamp byte
0AE9 F604               ORI     Erron                   ; Set the error lamp bit
0AEB 32974E             STA     Byte5                   ; Put the byte back
0AEE CD4C07             CALL    Setprt                  ; Turn on the lamp
0AF1 CD0C02             CALL    Clribf                  ; Clear the 488 input buffer
0AF4 CD1502             CALL    Clribf                  ; Clear the 232 input buffer
0AF7 CD1E02             CALL    Clchbf                  ; Clear the character buffer
0AFA 3A974E             LDA     Status                  ; Get the GPIB status byte
0AFD F601               ORI     Errr                    ; Set the error bit
0AFF 32974E             STA     Status                  ; Put the byte back
```

```
                ; Just "drop through" into the next routine

                ;*****************************************************************************
                ;*                    ROUTINE TO CHECK OUT AND ASSERT SRQ                    *
                ;*****************************************************************************

                ; This routine is called when an SRQ is requested. It checks the current GPIB
                ; status byte, and compares it against the operator-specified mask in RAM. If
                ; the mask allows it, an SRQ interrupt is initiated on the 488 bus.

0AD2 F5         Srq_do  PUSH    PSW             ; Save the registers
0AD3 E5                 PUSH    H

0AD4 F3                 DI                      ; Disable interrupts for a bit
0AD5 3A973E             LDA     Status          ; Get the current status byte
0AD8 D335               OUT     Spoll           ; Store it in the 488 chip's register
0ADA 21983E             LXI     H,Srqmsk        ; Set memory pointer to the SRQ mask
0ADD A6                 ANA     M               ; AND them together
0ADE CAF00A             JZ      By              ; Skip SRQ if the mask does not allow

0AE1 3E98               MVI     A,Rsv2          ; Assert SRQ
0AE3 D333               OUT     Auxcmd          ; Put into control register
0AE5 3A943E             LDA     Byte5           ; Get the front panel lamp byte
0AE8 F608               ORI     Srqon           ; Turn on the SRQ lamp
0AEA 32943E             STA     Byte5           ; Put the byte back
0AED CD4C03             CALL    Setprt          ; Send it to the ports

0AF0 E1         By      POP     H               ; Get registers back
0AF1 F1                 POP     PSW
0AF2 C9                 RET

                ;*****************************************************************************
                ;*                        LOOKUP TABLES AND POINTERS                         *
                ;*****************************************************************************

                ; Baud rate divisor latch table

0AF3 0F00       Baudtb  DW      000FH           ; 50 Baud
0AF5 0A00               DW      000AH           ; 75
0AF7 06D1               DW      0D106H          ; 110
0AF9 0594               DW      9405H           ; 134.5
0AFB 0500               DW      0005H           ; 150
0AFD 0280               DW      8002H           ; 300
0AFF 0140               DW      4001H           ; 600
0B01 00A0               DW      0A000H          ; 1,200
0B03 006B               DW      6B00H           ; 1,880
0B05 0060               DW      6000H           ; 2,000
0B07 0050               DW      5000H           ; 2,400
0B09 0035               DW      3500H           ; 3,600
0B0B 0028               DW      2800H           ; 4,800
0B0D 001B               DW      1B00H           ; 7,200
0B0F 0014               DW      1400H           ; 9,600 (Default)
0B11 000A               DW      0A00H           ; 19,200
0B13 0005               DW      0500H           ; 38,400
```

```
0B15 0003                       DW      0300H                   ; 56,000

                        ;*****************************************************************************
                        ;*                          COMMAND MATCH STRINGS                            *
                        ;*****************************************************************************

                        ; System commands:

0B17 03434C52   System  DB      03H,"CLR"               ; String match for "CLR" command
0B1B 03434C50           DB      03H,"CLP"               ; Etc...
0B1F 03434C43           DB      03H,"CLC"
0B23 0350544D           DB      03H,"PTM"
0B27 03525342           DB      03H,"RSB"
0B2B 0352424D           DB      03H,"RBM"
0B2F 03525053           DB      03H,"RPS"
0B33 0349454F           DB      03H,"IEO"
0B37 0352534F           DB      03H,"RSO"
0B3B 03424F54           DB      03H,"BOT"
0B3F 03535453           DB      03H,"STS"
0B43 03534552           DB      03H,"SER"
0B47 03424452           DB      03H,"BDR"
0B4B 0353544D           DB      03H,"STM"
0B4F 03524144           DB      03H,"RAD"
0B53 03414452           DB      03H,"ADR"
0B57 03415454           DB      03H,"ATT"
0B5B 00                 DB      00H                     ; End of list indicator

                        ; Path commands:

0B5C 07474E313D Pathst  DB      07H,"GN1=RO1"           ; String match for "GN1=RO1"
     524F31
0B64 07474E323D         DB      07H,"GN2=RO1"           ; Etc...
     524F31
0B6C 075249313D         DB      07H,"RI1=RO1"
     524F31
0B74 074E4F533D         DB      07H,"NOS=RO1"
     524F31
0B7C 075350413D         DB      07H,"SPA=RO1"
     524F31
0B84 07474E313D         DB      07H,"GN1=RO2"
     524F32
0B8C 07474E323D         DB      07H,"GN2=RO2"
     524F32
0B94 075249323D         DB      07H,"RI2=RO2"
     524F32
0B9C 074E4F533D         DB      07H,"NOS=RO2"
     524F32
0BA4 075350413D         DB      07H,"SPA=RO2"
     524F32
0BAC 075350413D         DB      07H,"SPA=RI3"
     524F33
0BB4 075350413D         DB      07H,"SPA=RI4"
     524F34
0BBC 075350413D         DB      07H,"SPA=RI5"
     524F35
```

```
0BC4 03545231              DB        03H,"TR1"
0BC8 03545232              DB        03H,"TR2"
0BCC 03454631              DB        03H,"EF1"
0BD0 03564631              DB        03H,"VF1"
0BD4 034C4631              DB        03H,"LF1"
0BD8 03454632              DB        03H,"EF2"
0BDC 03564632              DB        03H,"VF2"
0BE0 034C4632              DB        03H,"LF2"
0BE4 0341494E              DB        03H,"AIN"
0BE8 03535031              DB        03H,"SP1"
0BEC 03535032              DB        03H,"SP2"
0BF0 0343424E              DB        03H,"CBN"
0BF4 03434246              DB        03H,"CBF"
0BF8 03534946              DB        03H,"SIF"
0BFC 034D414E              DB        03H,"MAN"
0C00 034D4146              DB        03H,"MAF"
0C04 03414F54              DB        03H,"AOT"
0C08 03414F46              DB        03H,"AOF"
0C0C 034E4F4E              DB        03H,"NON"
0C10 034E4F46              DB        03H,"NOF"
0C14 03504D4E              DB        03H,"PMN"
0C18 03504D46              DB        03H,"PMF"
0C1C 054C58313D            DB        05H,"LX1=1"
     31
0C22 054C58313D            DB        05H,"LX1=0"
     30
0C28 054C58323D            DB        05H,"LX2=1"
     31
0C2E 054C58323D            DB        05H,"LX2=0"
     30
0C34 054C58333D            DB        05H,"LX3=1"
     31
0C3A 054C58333D            DB        05H,"LX3=0"
     30
0C40 054C58343D            DB        05H,"LX4=1"
     31
0C46 054C58343D            DB        05H,"LX4=0"
     30
0C4C 054C58353D            DB        05H,"LX5=1"
     31
0C52 054C58353D            DB        05H,"LX5=0"
     30
0C58 00                    DB        00H                    ; End of table indicator

                ;*****************************************************************************
                ;*                          PATH BIT MAP CONTROL TABLES                      *
                ;*****************************************************************************

                ; Control code table:

0C59 070B0F1011 G1_rol     DB        07H,08H,0FH,10H,11H,12H,0D3H
     12D3
0C60 B70F101112 G2_rol     DB        B7H,0FH,10H,11H,12H,53H
     53
0C66 070F101192 R1_rol     DB        07H,0FH,10H,11H,92H,53H
```

```
      53
0C6C 070D8F1011 No_ro1    DB        07H,0DH,8FH,10H,11H,12H,53H
      1253
0C73 070F109112 Sp_ro1    DB        07H,0FH,10H,91H,12H,13H,15H,18H,19H,5AH
      131518195A
0C7D 880D141516 G1_ro2    DB        88H,0DH,14H,15H,16H,57H
      57
0C83 07080A0D14 G2_ro2    DB        07H,08H,0AH,0DH,14H,15H,16H,0D7H
      1516D7
0C8B 080D141596 R2_ro2    DB        08H,0DH,14H,15H,96H,57H
      57
0C91 088D141516 No_ro2    DB        08H,8DH,14H,15H,16H,57H
      57
0C97 080D111495 Sp_ro2    DB        08H,0DH,11H,14H,95H,16H,17H,18H,19H,5AH
      161718195A
0CA1 111598195A Sp_ri3    DB        11H,15H,98H,19H,5AH
0CA6 111518995A Sp_ri4    DB        11H,15H,18H,99H,5AH
0CAB 11151819DA Sp_ri5    DB        11H,15H,18H,19H,0DAH
0CB0 070F901112 Tr1       DB        07H,0FH,90H,11H,12H,53H
      53
0CB6 080D941516 Tr2       DB        08H,0DH,94H,15H,16H,57H
      57
0CBC C3         Ef1       DB        0C3H
0CBD 0143       Vf1       DB        01H,43H
0CBF 810243     Lf1       DB        81H,02H,43H
0CC2 8546       Ef2       DB        85H,46H
0CC4 040546     Vf2       DB        04H,05H,46H
0CC7 840546     Lf2       DB        84H,05H,46H
0CCA 818243     Ain       DB        81H,82H,43H
0CCD 8B4C       Sp1       DB        8BH,4CH
0CCF CC         Sp2       DB        0CCH
0CD0 CA         Cbn       DB        0CAH
0CD1 4A         Cbf       DB        4AH
0CD2 4B         Sif       DB        4BH
0CD3 C0         Man       DB        0C0H
0CD4 40         Maf       DB        40H
0CD5 864A       Aot       DB        86H,4AH
0CD7 46         Aof       DB        46H
0CD8 CE         Non       DB        0CEH
0CD9 4E         Nof       DB        4EH
0CDA C9         Pmn       DB        0C9H
0CDB 49         Pmf       DB        49H
0CDC DB         L=1_1     DB        0DBH
0CDD 5B         L=1_0     DB        5BH
0CDE DC         L=2_1     DB        0DCH
0CDF 5C         L=2_0     DB        5CH
0CE0 DD         L=3_1     DB        0DDH
0CE1 5D         L=3_0     DB        5DH
0CE2 DE         L=4_1     DB        0DEH
0CE3 5E         L=4_0     DB        5EH
0CE4 DF         L=5_1     DB        0DFH
0CE5 5F         L=5_0     DB        5FH

                ; Code pointer table:
```

```
0CE6 590C       Patptr  DW      G1_ro1          ; Point to location of first code set
0CE8 600C               DW      G2_ro1
0CEA 660C               DW      R1_ro1
0CEC 6C0C               DW      No_ro1
0CEE 730C               DW      Sp_ro1
0CF0 7D0C               DW      G1_ro2
0CF2 830C               DW      G2_ro2
0CF4 8B0C               DW      R2_ro2
0CF6 910C               DW      No_ro2
0CF8 970C               DW      Sp_ro2
0CFA A10C               DW      Sp_ri3
0CFC A60C               DW      Sp_ri4
0CFE AB0C               DW      Sp_ri5
0D00 B00C               DW      Tr1
0D02 B60C               DW      Tr2
0D04 BC0C               DW      Ef1
0D06 BD0C               DW      Vf1
0D08 BF0C               DW      Lf1
0D0A C20C               DW      Ef2
0D0C C40C               DW      Vf2
0D0E C70C               DW      Lf2
0D10 CA0C               DW      Ain
0D12 CD0C               DW      Sp1
0D14 CF0C               DW      Sp2
0D16 D00C               DW      Cbn
0D18 D10C               DW      Cbf
0D1A D20C               DW      Sif
0D1C D30C               DW      Man
0D1E D40C               DW      Maf
0D20 D50C               DW      Aot
0D22 D70C               DW      Aof
0D24 D80C               DW      Non
0D26 D90C               DW      Nof
0D28 DA0C               DW      Pmn
0D2A DB0C               DW      Pmf
0D2C DC0C               DW      Lx1_1
0D2E DD0C               DW      Lx1_0
0D30 DE0C               DW      Lx2_1
0D32 DF0C               DW      Lx2_0
0D34 E00C               DW      Lx3_1
0D36 E10C               DW      Lx3_0
0D38 E20C               DW      Lx4_1
0D3A E30C               DW      Lx4_0
0D3C E40C               DW      Lx5_1
0D3E E50C               DW      Lx5_0
0D40 0000               DW      0000H           ; Indicates end of pointer list

                ;*****************************************************************************

    0 errors
```

```
0020 Act232
0032 Addsta
0485 Adiei
042F Adrsi
0CD7 Aof
083E Att
0AF3 Baudtb
0802 Bdr
08EA Bn_bcd
07D9 Bot
0AF0 By
3E90 Byte1
3E93 Byte4
0CD0 Cbn
3E0D Chbep
0406 Chep
02A7 Clchbf
028F Cliobf
0729 Clp
087C Clrbit
029F Clrobf
0686 Comnd
02AE Countr
0037 Datin
0000 Defbt1
0000 Defbt4
0656 Dek
05AF Delete
0038 Dllsb
0001 Drbit
0915 Dunnit
0CC2 Ef2
06A1 Err
0001 Errr
0886 Exiter
01F1 Fin
020A Fploff
0C59 G1_ro1
0C83 G2_ro2
0983 Getset
0057 Go_0
0041 Gpib
0001 Hpibon
03C6 Ieiep
07B3 Ieo
0004 Ieomsk
3C01 Iibep
0071 Int1
02BB Int488
0021 Intr5
0076 Intrid
3C05 Iobbc
0270 Irl
0004 Isr
0417 Loose
04F5 Adch
0034 Addswt
049F Adieo
0415 Adrso
0CD5 Aot
09BE Attset
0895 Bcd_bn
0000 Begin
09AE Bnb
0030 Bport
0645 Bye
3E91 Byte2
3E94 Byte5
3E0F Chbase
3E0C Chbfp
03F6 Chfp
02AC Clearr
0576 Cln232
0724 Clr
0297 Clribf
0000 Clrst
06D3 Comtch
0561 Cr
0037 Datout
0088 Defbt2
0000 Defbt5
0626 Dek0
02F2 Dfault
0039 Dlmsb
0020 Dsr
05BD Dwon
0008 Eoim
0004 Erron
0001 Erstat
08C3 Fgs
00A3 Finish
0959 Fte
0C7D G1_ro2
0701 Geek
01EB Gh
015D Go_1
0AA1 Grab
08F4 Hund
03B6 Ieifp
03E6 Ieoep
3C03 Iibase
3C00 Iibfp
0039 Int2
002C Int5_5
0034 Intr6
000B Intwrd
3C04 Iobep
0178 Leepup
0179 Looter
0CBF Lf1
0034 Addreg
3E9A Adgpib
082E Adr
0CCA Ain
060B Asserv
0033 Auxcmd
0014 Bd9600
0020 Bi
0010 Bo
0216 Bufchk
3E8F Byte0
3E92 Byte3
0CD1 Cbf
3E0E Chbbc
0273 Checkr
0732 Clc
0287 Cliibf
0566 Cln488
00BB Clr_it
0001 Clrin
0129 Cmpr
0037 Contrl
0010 Cts
0000 Defbt0
0004 Defbt3
0316 Defsys
0002 Delays
0080 Dlab
08A9 Do_it
0001 Dtr
0CBC Ef1
0097 Eoiset
0AB6 Error
0874 Escp
08DE Fgsdi
02B8 Fns
096E Ftf
0C60 G2_ro1
0856 Getcod
0866 Ghi
06E9 Go_on
0205 Hgt
094A Iedoit
00D9 Ieinto
03D6 Ieofp
3C02 Iibbc
0030 Int0
02E3 Int232
0034 Int6_5
0279 Intram
3C06 Iobase
3C83 Iobfp
0115 Ip
092F Ld
0CC7 Lf2
```

```
0038 Linctl     00AB Linefd     003D Linsta
056A Lool       057A Loolit     0010 Lpas
0002 Lstone     0621 Lupe       0CDD L:1_0
0CDC L:1_1      0CDF L:2_0      0CDE L:2_1
0CE1 L:3_0      0CE0 L:3_1      0CE3 L:4_0
0CE2 L:4_1      0CE5 L:5_0      0CE4 L:5_1
0CD4 Maf        0CD3 Man        0924 Md
0020 Mla        003C Modctl     003E Modsta
0040 Mta        08BE Mu         06D6 Nextpt
0C6C No_ro1     0C91 No_ro2     0010 Nocmnd
05AC Noefct     0CD9 Nof        06F0 Nomo
0CD8 Non        0712 Nope       01DB Normal
0595 Noway      090A Ones       08D6 Oops
093A Outgo      00BF Outie      0040 Pastru
06BC Patcom     0848 Patdo      0B5C Pathst
0CE6 Patptr     0CDB Pmf        0CDA Pmn
0040 Port40     0041 Port41     0042 Port42
0043 Port43     0044 Port44     0045 Port45
013B Pr1        05CC Pr232      05B1 Pr488
0618 Prcss      067A Proc       01E5 Proc2
0008 Pstr       05FA Pt232      05E4 Pt488
073B Ptm        011E Ptpg       0080 Ptproc
0002 Ptstat     0C66 R1_ro1     0C8B R2_ro2
0B17 Rad        027F Rambot     3C00 Ramstr
077A Rbm        0004 Rcv        0189 Rcvser
003B Recbuf     0001 Recda      0004 Replie
08E5 Reslt      0583 Retoin     071A Rfe
3D09 Ribase     3D08 Ribbc      3D07 Ribep
3D06 Ribfp      3D8C Robase     3D8B Robbc
3D8A Robep      3D89 Robfp      07A4 Rps
086C Rptr       0755 Rsb        0968 Rsdoit
0386 Rsiep      0376 Rsifp      07C6 Rso
03A6 Rsoep      0396 Rsofp      0008 Rsomsk
0651 Rstr       0098 Rsv2       0002 Rts
01FA Scat       0001 Scrtch     07F5 Ser
0147 Serial     0002 Serl       0002 Serlon
005F Serpll     01CD Serpth     3E95 Sertrm
0A8F Set_br     0883 Setbit     06F3 Setmct
034C Setprt     0891 Setter     0CD2 Sif
05F5 Snd232     0CCD Sp1        0CCF Sp2
0CA1 Sp_ri3     0CA6 Sp_ri4     0CAB Sp_ri5
0C73 Sp_ro1     0C97 Sp_ro2     0004 Spas
0588 Spcspn     0035 Spoll      0001 Srmask
0AD2 Srq_do     3E98 Srqmsk     0008 Srqon
0004 Srstat     3FFF Stack      0000 Start
3E97 Status     0006 Stkint     080B Stm
007F Stoppt     07E9 Sts        0080 Swrst
0004 Sysbyt     06A5 Syscom     0723 Sysdo
3E99 Systat     0B17 System     0002 Tads
007E Talk       0002 Tbei       08FF Tens
0020 Thre       050F Tkch       04B9 Tkiei
052D Tkieia     04D7 Tkieo      0449 Tkrsi
0547 Tkrsia     0467 Tkrso      0077 Tlklst
0008 Tpas       0CB0 Tr1        0CB6 Tr2
0040 Tsre       05D8 Tue        0002 Txm
```

```
0A83 Vby                0CBD Vf1                0CC4 Vf2
0A75 Vrnr               003D Wait               3E96 Wordse
0965 X_s_l              097F X_s_r              0852 Xcute
0038 Xmtbuf             0722 Yup
```

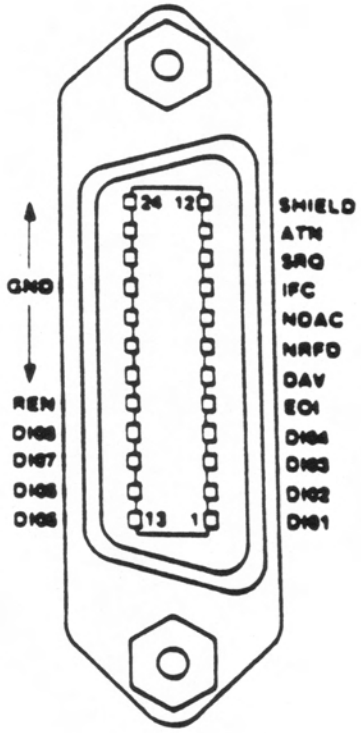
TF-30194 MICROWAVE ATE INTERFACE

Hardware References

II.

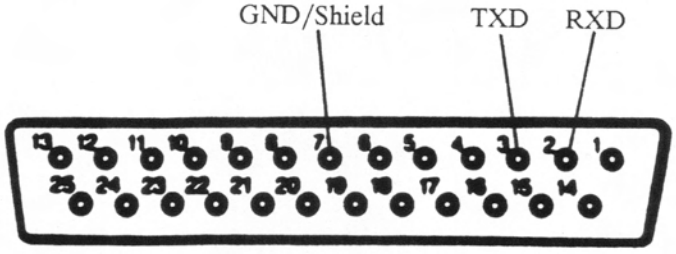
TF-30194 MICROWAVE ATE INTERFACE

IEEE-488 Connector



J20 IEEE-488

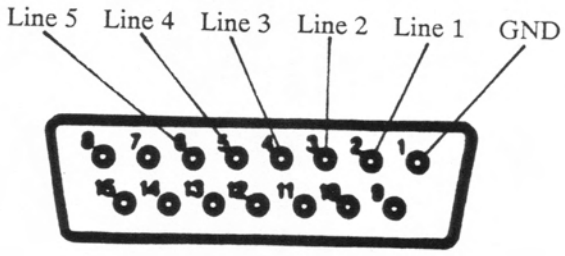
RS-232 Connector



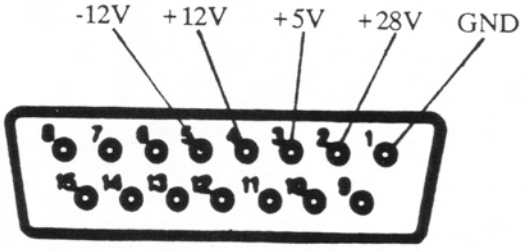
Front Panel RS-232 Connector

TF-30194 MICROWAVE ATE INTERFACE

Auxilliary Output Connectors



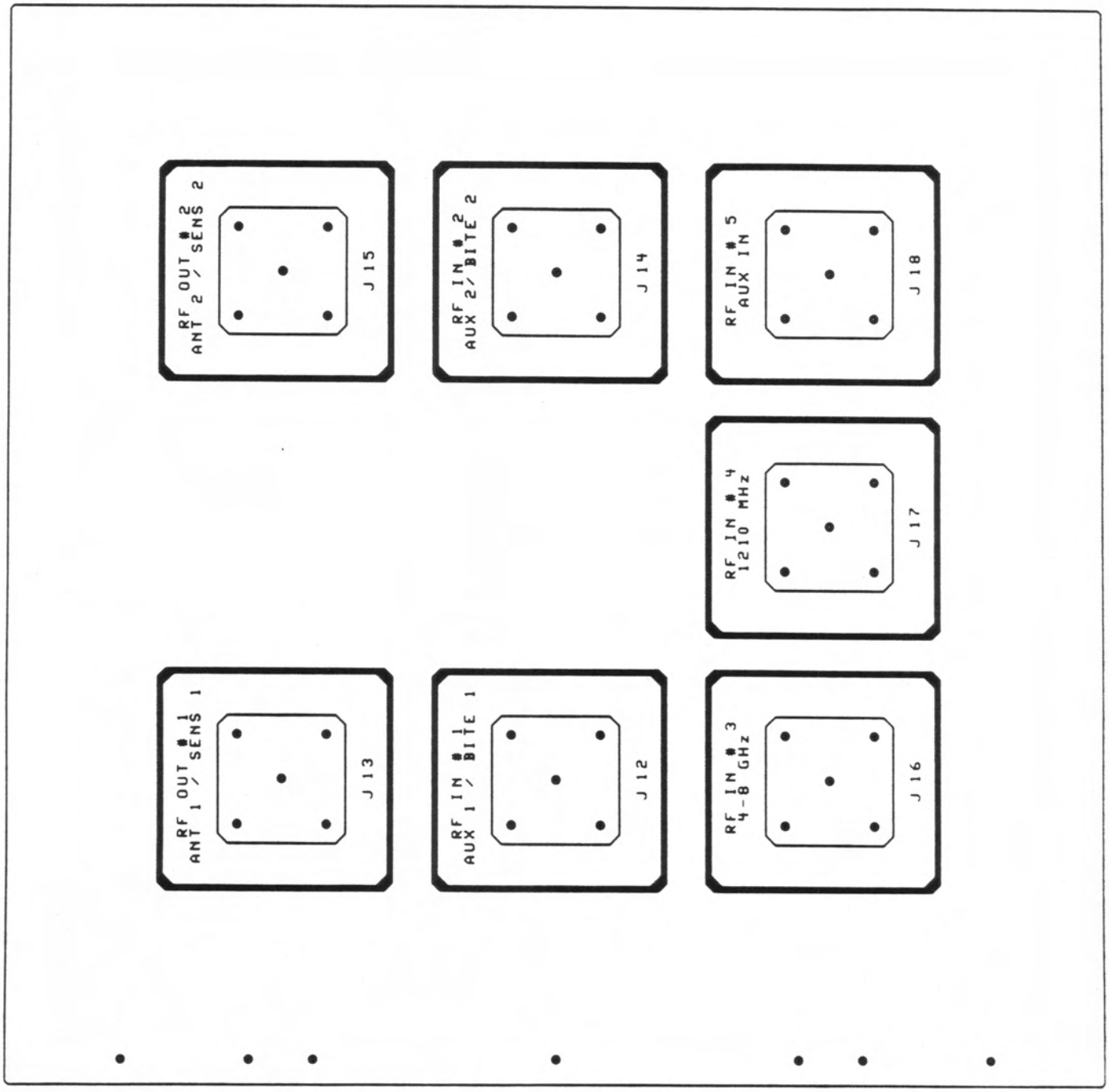
J18 External Control



J19 External DC Power

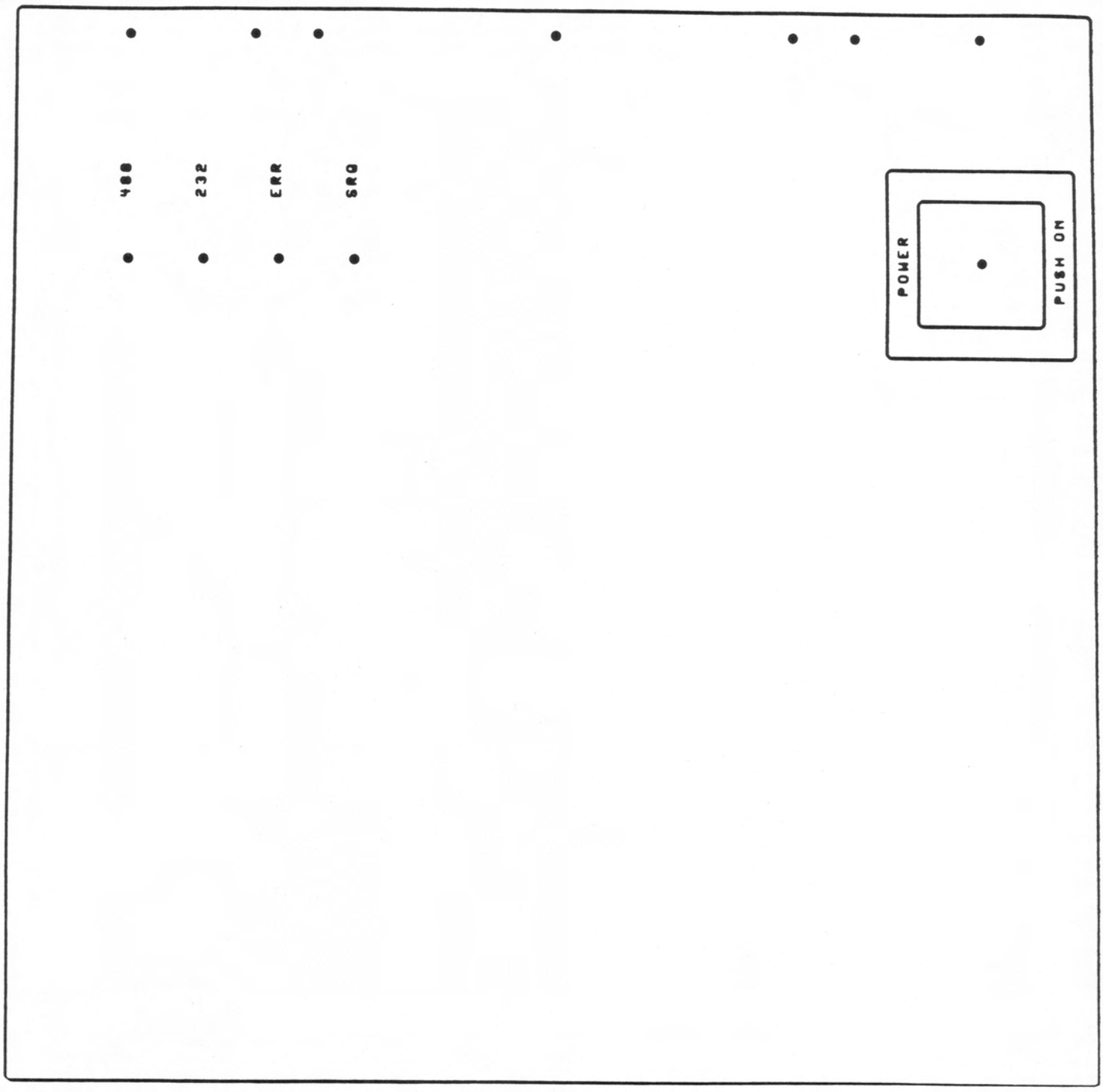
TF-30194 MICROWAVE ATE INTERFACE

Layout of Left Side of Front Panel



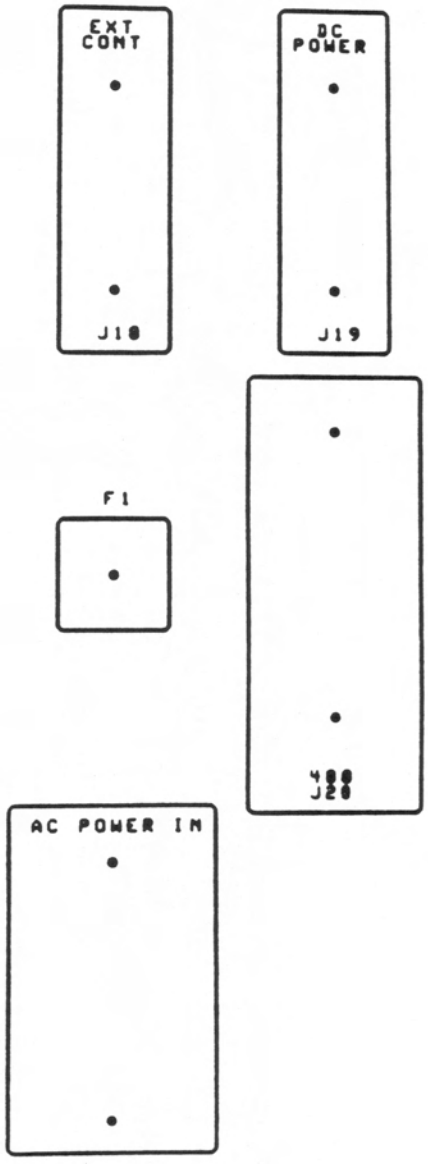
TF-30194 MICROWAVE ATE INTERFACE

Layout of Right Side of Front Panel



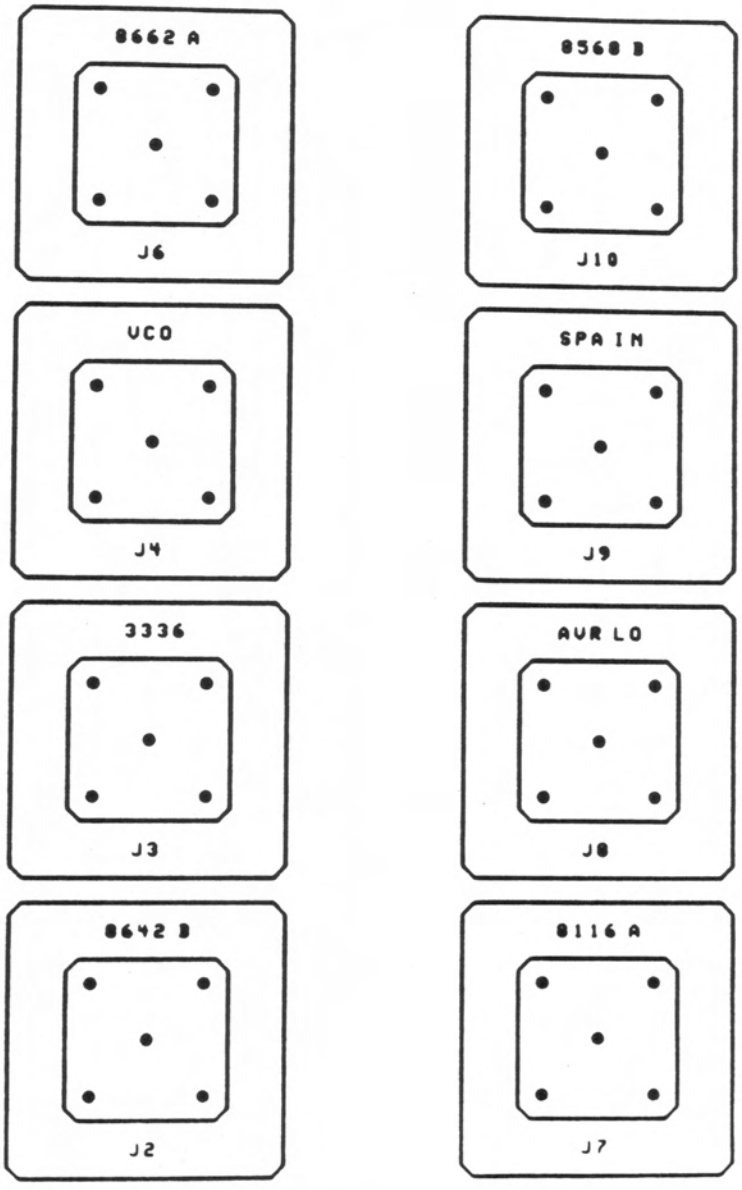
TF-30194 MICROWAVE ATE INTERFACE

Layout of Left Side of Rear Panel (facing rear from behind)



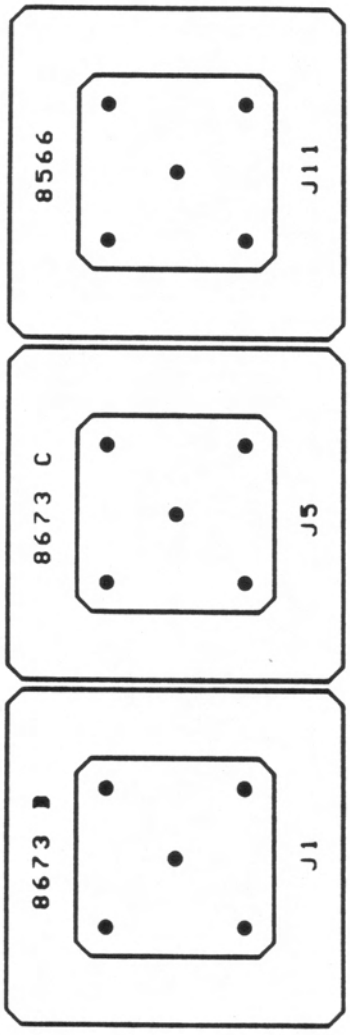
TF-30194 MICROWAVE ATE INTERFACE

Layout of Right Side of Rear Panel (facing rear from behind)



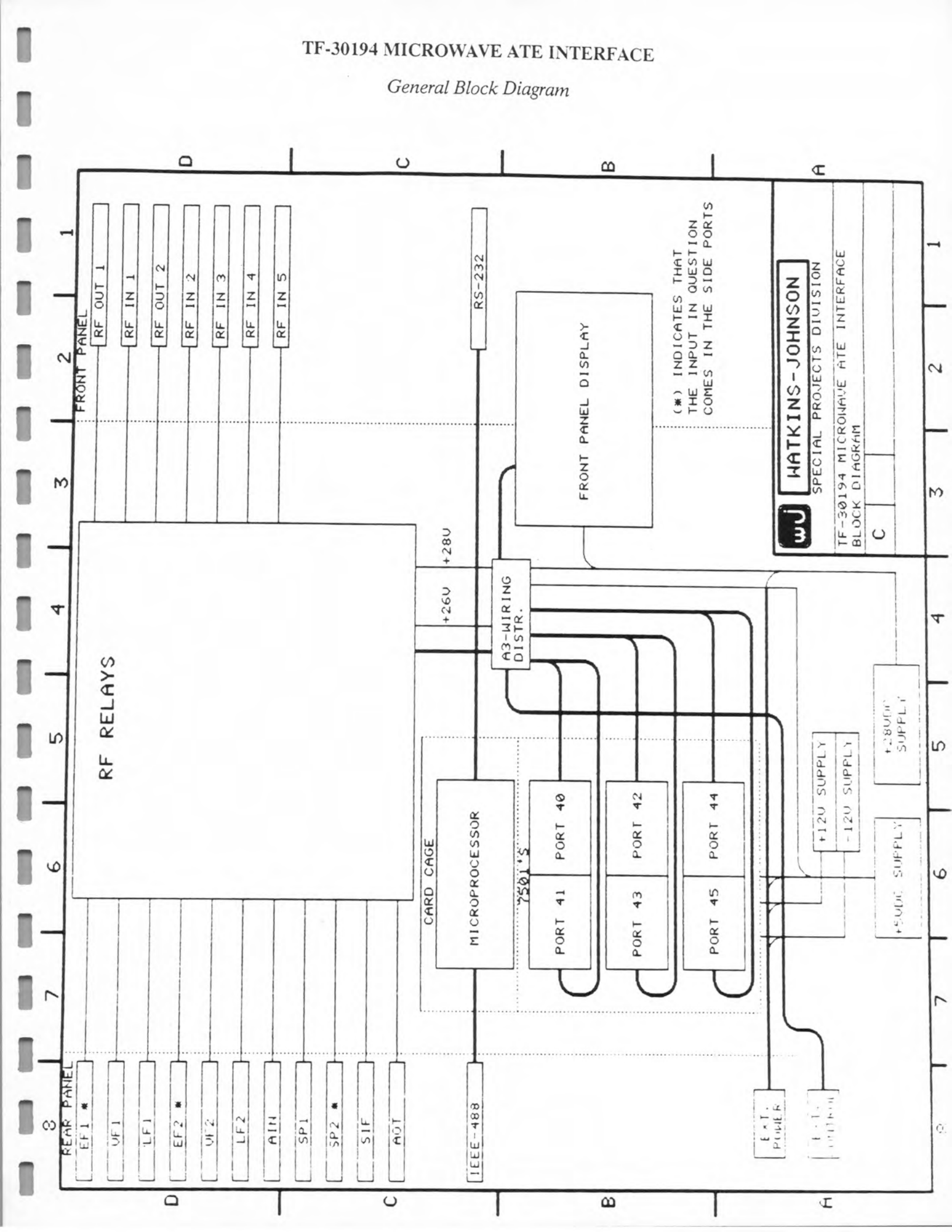
TF-30194 MICROWAVE ATE INTERFACE

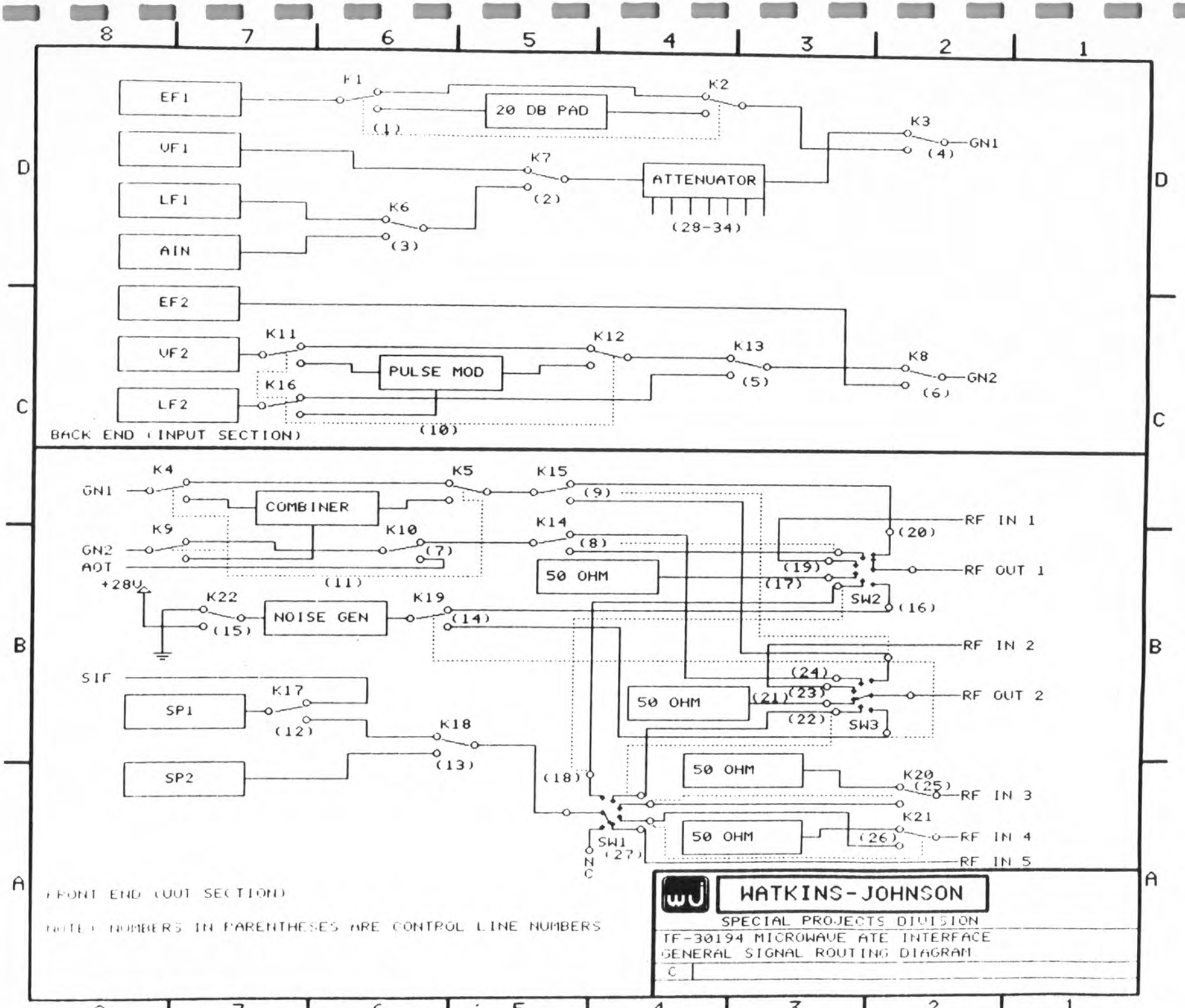
Side Panel Inset Layout



TF-30194 MICROWAVE ATE INTERFACE

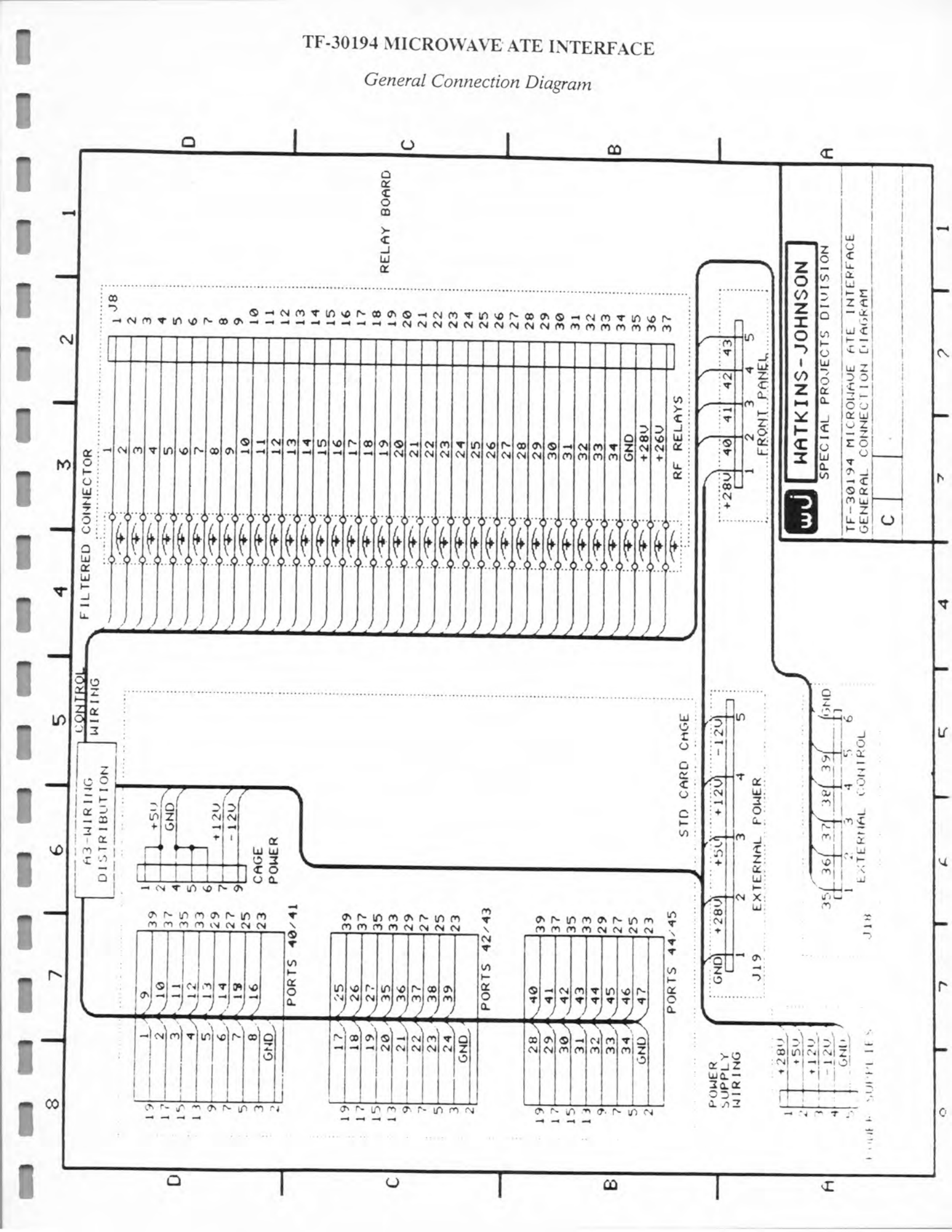
General Block Diagram




EF1
VF1
LF1
AIN
EF2
VF2
LF2
K1
20 DB PAD
(1)
K2
K3
GN1
(4)
K7
ATTENUATOR
(2)
(28-34)
K6
(3)
K11
K12
K13
K8
GN2
(6)
PULSE MOD
(5)
K16
(10)
BACK END (INPUT SECTION)
K4
K5
K15
GN1
COMBINER
(9)
K9
K10
K14
GN2
AUT
(7)
(8)
(11)
+28V
K22
(15)
NOISE GEN
K19
(14)
50 OHM
(19)
(17)
SW2
(16)
(20)
RF IN 1
RF OUT 1
RF IN 2
(24)
(21)
(23)
(22)
SW3
RF OUT 2
SIF
SP1
K17
(12)
K18
(13)
SP2
(18)
50 OHM
K20
(25)
RF IN 3
K21
(26)
RF IN 4
SW1
(27)
NC
RF IN 5
FRONT END (OUT SECTION)
NOTE: NUMBERS IN PARENTHESES ARE CONTROL LINE NUMBERS
WJ
WATKINS-JOHNSON
SPECIAL PROJECTS DIVISION
TF-30194 MICROWAVE ATE INTERFACE
GENERAL SIGNAL ROUTING DIAGRAM
C

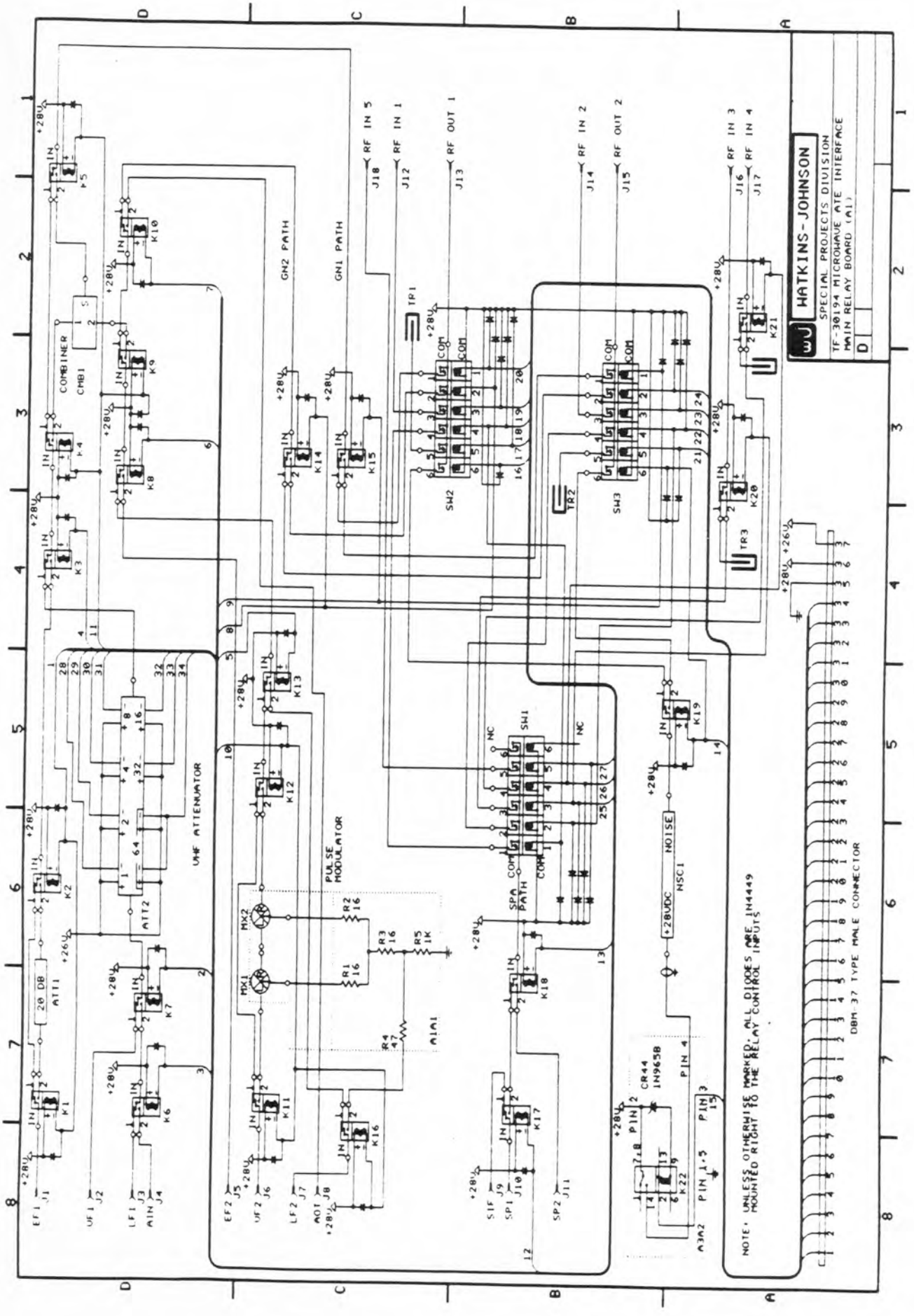
TF-30194 MICROWAVE ATE INTERFACE

General Connection Diagram



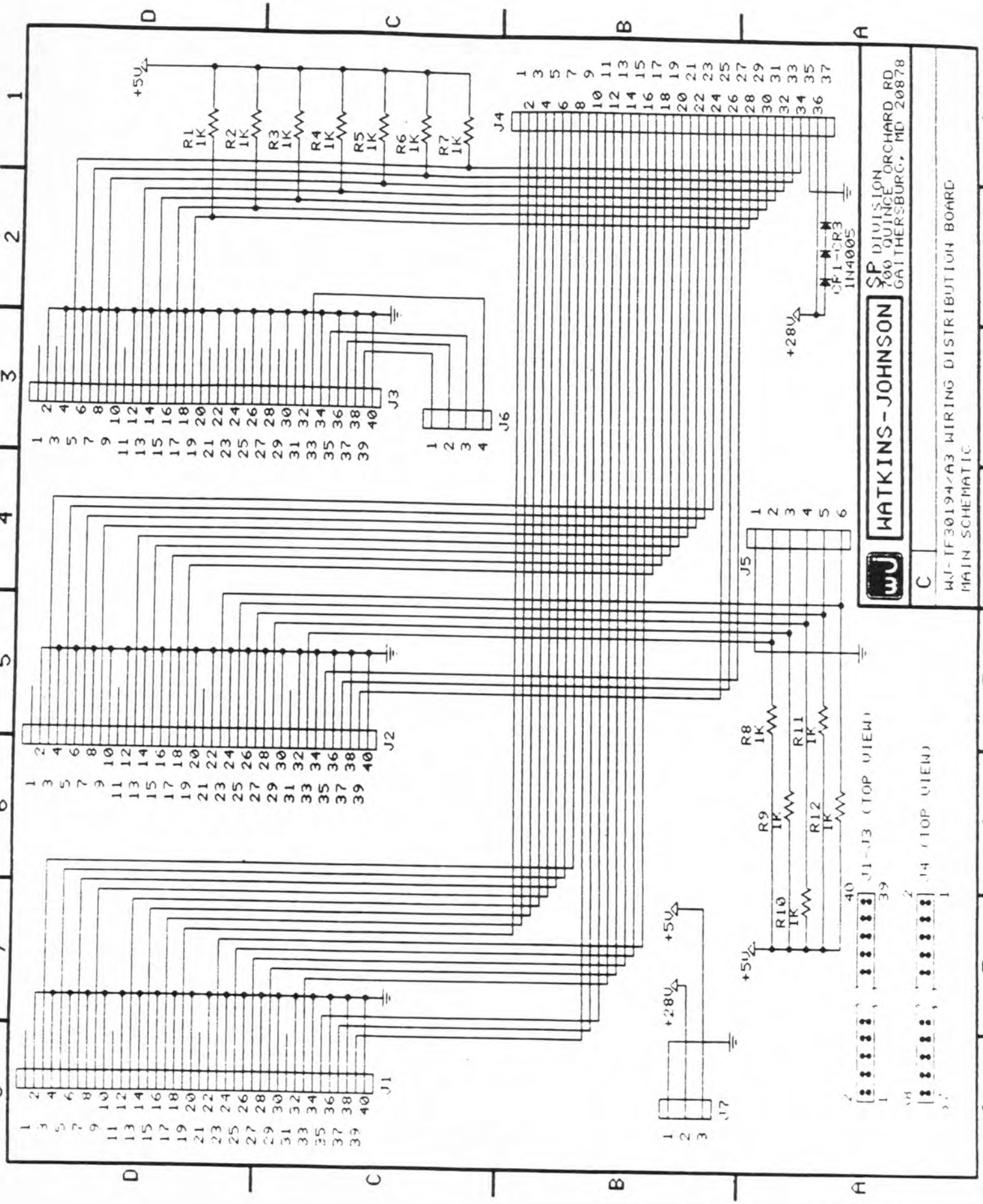
TF-30194 MICROWAVE ATE INTERFACE

Schematic of The Main Relay Plate



TF-30194 MICROWAVE ATE INTERFACE

Schematic of A3 Wiring Distribution Board



TF-30194 MICROWAVE ATE INTERFACE

Parts List
Main Chassis/RF Relays

Control #	Qty.	Part No.	Description	Notes/Vendor	Desig.
	1	444-20DB	20 dB SMA Pad	Midwest Mic.	ATT1
	1	PA-54	Prog. Attenuator	Texscan	ATT2
	1		Resistor Network	In-house	A1A1
	1		DC Relay Board	In-house	A1A2
	1		Wiring Dist. Board	In-house	A3
	1	1515	Microwave Comb.	Weinschel	CB1
	1	BR-16H(1)	STD Card Cage	PRO-LOG	CC1
S190115	38	1N4449	Diode		CR1-38
	3	PL-7501	DC Driver Card	PRO-LOG	DR1-3
S250145	1	342004	Fuse Holder		F1
	21	CS33S10	Microwave SPDT	Teledyne	K1-21
	17	3006-7941-00	Bulkhead N-conn.	Omni-Spect.	J1-17
S440046	2	842926-3	Filtered DBM-15	AMP	J18-19
S380448	1	553122-1	IEEE-488 Conn.		J20
	2	WJ-M1A	Mixer	WJ	MX1-2
	1	346B	Noise Source	HP	NSC1
	1	HB5-3/OVP-A	5V Power Supply	Power-One	PS1
	1	HD28-4	28V Power Supply	Condor	PS2
	1	HAD-12-0.4-A	+-12V Power Sup.	Power-One	PS3
	3	CS38S16	6-pole Coax Switch	Teledyne	SW1-3
N380087	4	2444	50 Ohm SMA Trm	Midwest Mic.	TR1-4
	1	ZT7805	GPIB Computer	Ziatech	UP1
S200298	94	201-1	SMA/Semi-Rigid		
N340256	2	217	SMA Male-Male	Omni-Spect.	
N580481	1	3082-2240-00	N-to-SMA Adpt.	Omni-Spect.	
	6	746094-9	40-Pin Rib. Conn.	AMP	
	1	746094-6	26-Pin Rib. Conn.	AMP	
	1	102387-1(2)	10-Pin Keyed Soc.	AMP	
	1	102387-9(3)	40-Pin Keyed Soc.	AMP	
N370171	3	DCM-37P	DBM-37 Male	ITT	
N370172	2	DCM-37S	DBM-37 Female	ITT	
N490110	1	56-733-004(4)	EMI DBM-37S	Metuschen	
S140492	1	DBM-25S	DBM-25 Female		
	3	745174-3	Cover, DBM-37	AMP	
S300113	1	1EF1	EMI Line Filter		
S250096	1	671-6-1	Pushbutton Sw.		
S200056	AR	AA50141	.141 Semi-Rigid		
N320419	24"	171-40	40-Cond. Rib. Ca.		
S380113	18"	1-499116-1	24-Cond. Rib. Ca.		

(1)This was a 16-position cage that was cut in half, and shortened vertically. You may also use a PRO-LOG^{tm} BR-08T 8-position card cage.

(2)The prototype uses a handmade "home-brew" connector.

(3)The prototype uses a handmade connector filed down to 38 pins. You may use the 40-pin connector offset by 1 row of pins.

(4)This was soldered straight (pin-to-pin) to one of the DBM-37P connectors and mounted to make a filtered male-to-female adaptor.

TF-30194 MICROWAVE ATE INTERFACE

Parts List
A3 Wiring Distribution Board

Control #	Qty.	Part No.	Description	Notes/Vendor	Desg.
S120378	3	1N4005	Diode		CR1-3
	4	2-102845-1	Pin Header assy.	AMP	J1-4
	3	102844-7	Pin Header assy.	AMP	J5-7
S360311	12	CF1/4-1K/J	Resistor, 1/4W, 1K		R1-12

TF-30194 MICROWAVE ATE INTERFACE

Parts List
A1A2 Noise Diode DC Switch

Control #	Qty.	Part No.	Description	Notes/Vendor	Desg.
S190117	1	1N965B	Zener Diode, 12V		CR44
	1	102844-7	Pin Header assy.	AMP	J1
	1	835C	Relay, DIP	GORDOS	K22

TF-30194 MICROWAVE ATE INTERFACE

Parts List
A1A1 Resistor Network

Control #	Qty.	Part No.	Description	Notes/Vendor	Desg.
S410274	3	CF1/4-16OHMS/J	Resistor, 1/4W, 16		R1-R3
S360297	1	CF1/4-47OHMS/J	Resistor, 1/4W, 47		R4
S360311	1	CF1/4-1K/J	Resistor, 1/4W, 1K		R5