DRC190 Remote Control System

Installation, Operation & Maintenance

Revised 17 November 1989

Hallikainen & Friends, Inc.
141 Suburban Road, Building E4
San Luis Obispo, CA 93401-7590, USA
Telephone (805) 541-0200
Fax (805) 544-6715
Telex 4932775 HFI UI

Table of Contents

Warranty

Hallikainen & Friends, a California corporation (CORP) hereby warrants, subject to the conditions herein below set forth, that should this product prove defective by reason of improper workmanship or defective materials within one (1) year from date of original purchase, Corp will repair or, at its option, replace the defective unit without charge for either parts or labor.

Contitions of Warranty

1. Notice. Purchaser shall notify Corp at its principal place of business by telephone within three (3) days after malfunction of the product. Time is deemed of essence.
2. Proper Delivery. The unit must be shipped, freight prepaid, or delivered to the manufacturing plant of Corp located at San Luis Obispo, CA 93401, in either its original package or a similar package affording an equal degree of protection.
3. The unit must not have been previously altered, repaired or serviced by anyone other than Corp, except for the replacement of plug in conponents with electrically identical components, or routine adjustments as outlined in the accompanying manual. Upon repair by customer, the Corp shall replace defective plug in parts returned to Corp, but shall not be liable for any labor expenses incurred in a field repair.
4. The serial number on the unit must not have been altered or removed; the unit must not have been subject to accident, misuse, or operated contrary to the instructions contained in the accompanying manual.
5. This warranty does not cover peripheral devices of other manufacturers supplied as part of a system by Corp (such as CRT terminals, printers, etc.); Purchaser's only remedies for malfunction with respect to such devices are with the equipment's manufacturer.
6. This warranty does not cover transporation expenses to and from service facility.
7. This warranty is in lieu of any other oral, written, or implied warranty, whether made by salesmen, agents, or other representatives of Corp.

Except to the extent prohibited by applicable law, all implied warranties made by Corp in connection with the product, including the warranty of merchantability are limited in duration to a period of one (1) year from the date of original purchase, and no warranties, whether expressed or implied, including said warranty of merchantability shall apply to this product after said period. Should this product prove defective in workmanship or material, the consumer's sole remedy shall be such repairs or replacements as hereinabove expressly provided; and under no circumstances shall Corp be liable for any loss or damage, direct or consequential, arising out of the use or inability to use this product.

FCC Notice

Warning: This equipment generates and uses radio frequency energy and if not installed and used properly, i.e. in strict accordance with the instructions manual, may cause harmful interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment.

Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Direct Connect Modem FCC Notice

FCC rules and regulations under part 68, requires the following information be provided to the user of FCC-registered terminal equipment such as the Cermetek CH1770 (used on the DRC190 direct connect modem card).

Section 68.100 GENERAL

Terminal equipment may be directly connected to the telephone network in accordance with the rules and regulations...of this part.

Section 68.104 STANDARD PLUGS AND JACKS

(a) General

"Except for telephone company provided ringers, all connections to the telephone network shall be made through standard (USOC) plugs and standard telephone company provided jacks, in such a manner as to allow for easy and immediate disconnection of the terminal equipment. Standard jacks shall be so arranged that if the plug connected thereto is withdrawn, no interference to the operation of equipment at the customer's premises which remains connected to the telephone network shall occur by reason of such withdrawal."

Section 68.106 NOTIFICATION TO TELEPHONE COMPANY

"Customers connecting terminal equipment or protective circuitry to the telephone network shall, before such connection is made, give notice to the telephone company of the particular line(s) to which such connection is to be made, and shall provide the telephone company the FCC Registration Number and Ringer Equivalence of the registered terminal equipment or protective circuitry. The customer shall give notice to the telephone company upon final disconnection of such equipment or circuitry from the particular line(s)."

Section 68.108 INCIDENCE OF HARM

"Should terminal equipment or protective circuitry cause harm to the telephone network, the telephone company shall, where practicable, notify the customer that temporary discontinuance of service may be required; however, where prior notice is not practicable, the telephone company may temporarily disconnect service forthwith, if such action is reasonable in the circumstances. In case of such temporary discontinuance, the telephone company shall (1) promptly notify the customer of such temporary discontinuance, (2) afford the customer the opportunity to correct the situation which gave rise to the temporary discontinuance, and (3) inform the customer of the right to bring a complaint to the Commission pursuant to the procedures set forth in Subpart E of this Part."

Section 68.216 REPAIR OF REGISTERED TERMINAL EQUIPMENT AND REGISTERED PROTECTIVE CIRCUITRY

"Repair of registered terminal equipment and registered protective circuitry shall be accomplished only by the manufacturer or assembler thereof or by their authorized agent; however, routine repairs may be performed by a user, in accordance with the instruction manual if the applicant certifies that such routine repairs will not result in non-compliance with the rules in Subpart D of this Part."

Section 68.218(b) ADDITIONAL INSTRUCTIONS TO USER

1. "...registered terminal equipment or protective circuitry may not be used with coin lines."

2. "...when trouble is experienced, the customer shall disconnect the registered equipment from the telephone line to determine if the registered equipment is manfunctioning, and...if the registered equipment is malfunctioning, the use of such equipment shall be discontinued until the problem has been corrected."

3. "...the user must give notice to the telepnone company in accordance with the requirements of Section 68.1Ø6..." for connecting the H&F Direct Connect Modem (Cermetek CH177Ø) to the telephone line.

NOTE:

As part of the H&F agreement with Cermetek, repairs to the H&F direct connect modem board should be referred to Hallikainen & Friends, NOT to Cermetek.

Copyright Notice

System Registration

To comply with software licensing agreements and to insure that the user receives notices regarding software and hardware updates, please complete the form below. Leave this page in the instruction manual, but photocopy it, sign the photocopy, and return the signed copy to Hallikainen & Friends. Thank you!

Non-Disclosure Agreement

The party below agrees that it is receiving a copy of Hallikainen & Friends DRC190 Firmware for use on a single computer only, as designated below. The party agrees to fill out and mail in this registration form before making use of Hallikainen & Friends DRC190 Firmware. The party agrees that all copies will be strictly safeguarded against disclosure to or use by persons not authorized by Hallikainen & Friends to use the DRC190 Firmware, and that the location of all copies will be reported to Hallikainen & Friends at its request. The party agrees that copying or unauthorized disclosure will cause great damage to Hallikainen & Friends, and this damage is far greater than the value of the copies involved. The party agrees that this agreement shall inure to the benefit of any third party holding any right, title or interest in the Hallikainen & Friends DRC190 Firmware or any software from which it was derived.

DRC190 Unit Serial Number: ____________________

User Name: ______________________________________

User Company: ___________________________________

Company Address: ________________________________

Company City, State, Zip: ____________________

Company Country: ________________________________

Company Telephone Number: ____________________

User Signature: _________________________________

Title: ____

Date: ____

Please return the signed copy of this form to:

Hallikainen & Friends, Inc.
141 Suburban Road, Building E4
San Luis Obispo, CA 93401-7590
USA

Introduction

The DRC19Ø is a data acquisition system optimized for use in the broadcast industry. The software and hardware have been generalized as much as possible, allowing any unit to act as a remote or control unit.

The system can be operated with anywhere between 1 and 1ØØ units in a system. A one unit system might be used to automatically log and control a local transmitter. A two unit system could be used to manually or automatically log and control a single remote transmitter site. A three or four unit system could be used to automatically or manually log and control an AM/FM or AM/FM/TV system with separate transmitter sites. Larger systems could be used to monitor broadcast microwave networks, or other systems with a large number of remote unattended sites.

The basic DRC19Ø box is a specialized microcomputer. It includes the standard microcomputer components (processor, memory and Input/Output) plus a 12ØØ bit/second half duplex modem and optional analog to digital converters.

In a simple remote control system, the "studio unit" includes a DRC19Ø with the processor, memory, modem, and a front panel display and keyboard. The operator keys in the desired site and channel that s/he wishes to check or adjust.

A remote site includes all the functions of a "studio unit" plus up to ten analog to digital converter boards. These boards select the appropriate metering sample and convert it to digital for use by the microprocessor. In addition, the processor can instruct the A/D boards to output control signals to adjust the external equipment.

Since the sample voltages provided to the DRC19Ø are proportional to the actual parameter to be measured, a scaling factor is established for each channel of metering. This scaling factor is established in the calibration procedure, and is stored in non-volatile memory at the remote site. Non-volatile memory also holds setup information regarding the unit (site number, communication speeds, labels and units for each metering channel, and a Morse Code identifier). Non-volatile memory (battery backed RAM) can also hold about 1 Kbyte of a Basic applications program. This program is automatically loaded and run on system reset. This program might be a simple logging program, or a "boot" loader program that loads a larger program from disc.

The sample voltages provided to the DRC19Ø can have up to +/-1ØØ volts of common mode voltage. The differential voltage must be limited to less than 2 volts. Analog to digital converter boards have a provision to install a voltage divider after the analog multiplexer allowing a higher differential sample voltage. Such a voltage divider can be added to the A/D board if required to avoid building separate voltage dividers for each sample. The temperature coefficeint of the voltage divider resistors will decrease the stability of the indicated sample, so low tempco resistors should be used. The control outputs are open collector and can drive 5ØØ mA with an open circuit voltage of less than 3Ø volts.

Optional status transceivers are also available. Each status transceiver transmits and receives 32 channels of status. Up to 3 status transceivers may be connected to each DRC19Ø, allowing the monitoring of 96 channels of status per site. When more than one status transceiver is required, the STX191 status expander is used. Each STX191 holds two status transceivers.

The transmit portion of the status transceiver reads TTL compatible input lines. Each input includes a 1ØK pullup resistor, allowing the input to be driven by a contact closure to ground or an open collector transistor. All transmitted status is identified at this and other site by site number (programmed in the DRC190) and channel number.

The receive portion of the status transceiver displays the received status on front panel LEDs and provides open collector outputs on the rear panel. Program jumpers on the status transceivers determine the site number the receiver portion responds to. A specific site number may be programmed in, or if the status receive site number is set to 100, the receive portion of the status transceiver displays the status of the site that has been selected using the front panel keyboard and display. This feature is useful when there are several sites that have the same status assignments on each status channel.

The DRC190 units communicate "half duplex". All units transmit and receive on the same frequencies, using Bell 202 standard FSK 1200 bit per second coding. The firmware includes a multiple access anti-contention system that assignes each unit a time slot when it is allowed to start transmitting data, if it has any data. If no data needs to be transmitted, no carrier is brought up, and no data transmitted. This absence of data is detected by other units in the system. Permission to transmit is then passed to the next unit in the system. On a system where it is not necessary to wait for external key-up and squelch delays (such as wire line or dedicated subcarrier), the time slot allocated to each unit typically runs 50 mS. If a site starts transmitting data in its allocated 50 mS time slot, the advancing of the site counter in all units is inhibited until the data transmission is completed. If no data is transmitted in the 50 mS time slot, the site counter at each site is advanced, granting permission to the next site to transmit, if it has any data. This forms a modified token passing permission to transmit to the next unit in the system. The use of absence of data as a token yields several improvements over typical token passing systems. Since no data is to be transmitted, the carrier bring up and shut down delays can be eliminated from the system when no data is to be transmitted. This results in higher speed token passing. Since the absence of data is the token, there is no danger of the token being "smashed" or lost due to a data error. This improves system reliability and reduces software complexity. When a site fails, it transmits no data, which is equivalent to always passing the token. This avoids system failures and system reconfiguration software requirements.

The DRC190 can use almost any audible communications link. These include standard "3002" data circuits, microwave and broadcast subcarriers, and separate radio links. In radio linked systems, all sites can transmit and receive on the same frequency using the same anti-contention firmware as the audio communications portion of the DRC190 system.

Since a voice-grade communications link is used by the DRC190, an intercom feature was included. When the COM button on the front panel is pressed, the DRC190 sends an FSK code telling other units to enable their front panel speakers. The operator is then allowed to talk into the front panel speaker for up to 30 seconds. That voice information will be heard from the speaker of each other unit in the system.

The intercom also includes a "listen" feature. When the decimal point key on the front panel keyboard is pressed, a "listen" command is sent to the site selected on the front panel display (the display is showing the site, channel and analog meter reading). That site responds by sending the intercom "talk" code, followed by 30 seconds of audio from the site, followed by an intercom off code. This allows an operator to listen to ambient audio at any site from any other site. The same command is also available in Basic. The command is LISTEN N where N is the site number to be listened to.

A Morse code identifier is included with the system to allow automatic identification of transmitters in radio linked systems. This firmware generates the FCC required 2Ø WPM 1.5 KHz Morse code station identification. The number of minutes between station identifications and the actual identifying code are programmable by the user through the DRC19Ø front panel.

As an option, the DRC19Ø includes a 32 channel "status transceiver". Each transceiver inludes 32 inputs and outputs. The 32 inputs are programmable through the DRC19Ø front panel as to whether each is active high or active low. On a change in status, the updated status is transmitted to all sites in the system. The status is available at each site through the Basic interpreter and through front panel LEDs. The front panel LEDs can be programmed to always display the status of a specific site, or can be programmed to display the status of the site selected with the front panel keyboard. In addition, a rear panel connector provides open collector outputs that track the front panel LEDs. These outputs can be used to drive remote indicators or alarms.

Each DRC19Ø includes a Basic interpreter. This program, along with the RS-232 port included in each unit, allows the user to write programs in Basic that can display the readings on a CRT terminal, log the readings on a printer, or log and control the operation of the station. If a printer and CRT are used, the CRT must have a peripheral port capable of driving the printer, and the peripheral port enable and disable control codes must be programmed into the DRC19Ø.

The DRC19Ø also includes optional subcarrier generation and demodulation (2Ø KHz to 2ØØ KHz) and a serial interface to the Commodore 1541 disc drive for program storage.

The DRC19Ø also includes a port for controlling IEEE488 automatic test equipment. At this writing, the firmware to support this item has not been written. Registered system users will be notified when that firmware is available.

Remote Diagnostics

The ability to program the DRC190 in a standard high level language gives the station great flexibility in the design of the transmitter control system. One of the most interesting possibilities is remote transmitter diagnostics.

When weekly transmitter site inspections were required, it was fairly common practice to log all the available system parameters during that weekly inspection. This once a week sample of the system parameters proved useful in detecting problems that were not revealed in the parameters that were normally remote metered (generally the minimum the FCC required). With the large number of metering channels available on the DRC190 along with the ability to program the unit, it is quite possible to remotely read all the parameters that could be read at the transmitter site. These parameters are continuously scanned by the system. The readings for each parameter are used in the calculation of minimum, maximum and average for each parameter. In addition, limits on all these parameters can be programmed. Exceeding the limits could cause the reading to be displayed, an alarm message to be displayed, and the terminal to "beep" to warn the operator of out of tolerance conditions. In addition, such a program could have a "detailed reading screen" that shows all these parameters with those out of tolerance in reverse video. It is now possible to do a site inspection without going to the site.

Using the report generation capability of the Basic in the DRC190, it is quite simple to print a daily report on each transmitter site. This daily report shows the minimum, maximum and average of all the remoted parameters. This daily report is simple for the designated chief operator to review. This review insures that the station operated within FCC required parameters during that day. In addition, trends in parameters not traditionally remoted (such as final grid current) can point out potential problems (driver tube getting weak) before they become serious. A sample daily report and log for an installed DRC190 system follows.

A list of possible remote parameters to remote follows. These are listed just to give some ideas. In general, the more you have remoted, the more you can determine from the comfort of your office.

For each STL Receiver

Power supply voltages
First and second oscillator outputs
Mixer drives
Received signal levels*
Discriminator voltage

*Often the "received signal level" indications on the STL receivers use arbitrary scales. For example, the Moseley PCL303 rectifies the signal level at a couple of places in the IF chain and calls them "Sig. 1" and "Sig. 2". The final test data relates these signal levels to received signal levels in uV. A program could display both the actual signal level indications along with calculated RF input voltages based on linear interpolation using a table of values from the final test data.

For each exciter

Power supply voltages
AFC voltage
RF amplifier parameters
Output forward and reflected power

For each transmitter

Voltages and currents for each element of each amplifier (grids, plates)
Forward and reflected power for each amplifier

For each RF combiner

Forward and reflected power for each input and output

For the DA antenna monitor

All loop currents and phase angles. These can be used to calculate loop current ratios, loop current ratio and phase deviations from licensed.

and more. . .

Other general site parameters might include line voltage, temperature, transmission line gas pressures, etc.

Since the DRC19Ø can also have analog inputs at the studio, additional system parameters can be gathered there for inclusion in the detailed reading screens and in the daily reports. These readings might include the various STL transmitter parameters and readings derived from modulation monitors (carrier, pilot and subcarrier frequencies, pilot and subcarrier injection levels, an indication of received signal strength, overmodulation peak counts, etc.

To summarize, meter all you can. From your desk, with a CRT terminal, you can view the whole station. With the daily report, you can view the operation of the station over the previous day, noting any trends or exceeded limits. The use of the daily report averages for DA parameters is especially useful in the adjustment of the array. The array should be adjusted so that the average deviations from licensed are zero. If tower three's phase is averaging one degree low, yet when you go to adjust the array it is reading two degrees low, you should adjust the array to bring the phase up one degree (to a present one degree low). This will result in the array swinging approximately equally above and below the licensed parameter, resulting in the operation of the system on the whole being as close to licensed as possible. It is possible, of course, that the swing of the parameter is asymmetrical, resulting in an average deviation of zero, but the minimum and maximum not being equal and opposite. If there is a serious discrepancy between the average and the median, the array could also be adjusted so that the median rather than the average (mean) is zero. In either case, it would appear that this would be a superior adjustment procedure to the typical procedure of adjusting the array to "right on" at 3:ØØ am.

Kiis AM/FM Transmitter Log
For Monday, August 4, 1986
All Times PDT
Page 1

Time	AM Readings									FM Readings				Comments
	EP	IP	IA	ICP	Power	DA Parameter	T1	T2	T3	EP	IP	In.TPO	Reflc.	
1:00:34 AM	5075.	1.32		9.9	98.0	Phase (deg)	51.2	0.0	140.6	6332.	1.96	97.3	1.1	
						Loop Current	46.6	97.3	45.5					
3:01:35 AM	5085.	1.31		9.8	97.8	Phase (deg)	51.2	0.0	140.7	6288.	1.94	95.4	1.1	
						Loop Current	46.5	97.8	45.9					
5:00:38 AM	5078.	1.33		10.1	103.1	Phase (deg)	51.3	0.0	140.7	6251.	1.93	94.6	1.1	
						Loop Current	47.6	98.7	46.9					
														Changed to day pattern: 6:15:35 AM
6:16:27 AM	4977.	1.31	11.5		102.1					6236.	1.92	93.6	1.1	
														Trimming AM power down: 6:22:57 AM
8:00:44 AM	4943.	1.30	11.4		99.1					6236.	1.91	93.2	1.1	
														Trimming AM power up: 8:48:30 AM
10:00:19 AM	4949.	1.31	11.4		100.1					6288.	1.92	94.7	1.1	
12:00:44 PM	4954.	1.29	11.3		98.3					6302.	1.93	95.2	1.1	
2:00:14 PM	4959.	1.30	11.4		98.8					6251.	1.91	93.4	1.1	
3:46:02 PM	4966.	1.31	11.3		97.5					6229.	1.90	92.4	1.1	
5:00:42 PM	4975.	1.29	11.4		99.5					6251.	1.90	93.2	1.1	
7:00:21 PM	4962.	1.30	11.3		98.4					6244.	1.91	93.6	1.1	
														Changed to night pattern: 7:45:21 PM
7:46:16 PM	5039.	1.31		9.8	96.4	Phase (deg)	51.0	0.0	140.6	6192.	1.89	91.4	1.1	
						Loop Current	46.1	97.1	45.6					
														Trimming AM power up: 8:18:16 PM
9:00:46 PM	5133.	1.32		9.9	99.9	Phase (deg)	51.2	0.0	140.4	6295.	1.94	95.6	1.1	
						Loop Current	46.9	99.3	46.4					
														Tower lights OK at10:00:14 PM
11:00:35 PM	5124.	1.33		10.0	100.8	Phase (deg)	51.1	0.0	140.7	6229.	1.91	93.3	1.1	
						Loop Current	47.2	98.3	46.0					

Printed at12:00:15 AM KIIS AM/FM Daily Report for August 4, 1986

KIIS-AM Readings

Label	DAY Average Reading	DAY Max Reading	DAY Min Reading	DAY Last Reading	NIGHT Average Reading	NIGHT Max Reading	NIGHT Min Reading	NIGHT Last Reading
Pri. Volts	119.9	120.3	119.6	120.0	120.1	120.4	119.8	120.0
Plate Volts	4948.	5036.	4839.	5108.	5087.	5154.	5000.	5108.
Plate Amps	1.30	1.34	1.26	1.31	1.33	1.36	1.29	1.31
AM Power %	99.3	104.1	95.7	97.2	99.1	104.5	95.1	97.2
AM Freq Dev	-.2	-.2	-.2	-.2	-.2	-.2	-.2	-.2
AM Room Temp	73.3	74.9	70.4	70.2	71.2	73.5	69.5	70.2
ND Base Amps	11.4	11.7	11.2	11.1	11.3	11.6	11.0	11.1
DA Com. Amps					9.9	10.2	9.7	9.8
T1 Loop Curr					46.8	48.4	45.5	46.3
T1 Lp Rtio %					47.6	49.2	46.7	47.4
T1 Rtio Dev%					-1.4	1.7	-3.4	-2.0
T1 Phase Deg					51.2	51.3	50.9	51.0
T1 Phs Dv Dg					.9	1.0	.6	.7
T2 Loop Curr					98.2	101.3	95.8	97.7
T2 Phase Deg					1.42723E-04	.1	0.0	0.0
T3 Loop Curr					46.0	47.5	45.0	45.9
T3 Lp Ratio%					46.8	47.8	45.4	46.9
T3 Rat Dev %					-1.3	.6	-4.3	-1.0
T3 Phase Deg					140.6	140.8	139.7	140.5
T3 Phs Dv Dg					1.2	1.4	.3	1.1

KIIS-FM Readings

Label	Average Reading	Max Reading	Min Reading	Last Reading
Pri. Volts	113.3	115.2	111.3	112.7
Plate Volts	6260.	6354.	6177.	6229.
Plate Amps	1.92	1.97	1.88	1.91
Indrct Pwr %	94.1	97.6	91.1	93.3
FM Rfl Pwr %	1.1	1.1	1.0	1.1
Direct Pwr %	100.6	102.3	98.7	100.1
Final G1 Ma	96.0	96.3	95.6	95.9
Final G2 Ma	217.2	217.8	216.4	217.1
Final G2 Vlts	666.4	668.1	663.7	665.8
Drvr G1 Ma	5.14	5.16	5.12	5.14
Drvr G2 Ma	5.04	5.07	5.02	5.04
Drvr Cat Ma	206.9	207.4	206.1	206.7
Drvr G2 Volts	232.3	232.9	231.4	232.1
Main Pres Lbs	4.14	4.15	4.12	4.13
Aux Press Lbs	4.14	4.15	4.12	4.13
Deicer Amps	0.00	0.00	0.00	0.00
FM Room Temp	80.7	89.6	72.5	74.5

<u>Review of FCC Rules</u>

The FCC is continually making updates to the rules regarding remote control and automatic control of broadcast stations. A good review of the appropriate rules is suggested prior to beginning the installation of a new transmitter control system.

In addition, there is considerable FCC policy that must be evaluated in determining the legality of a particlular installation approach. This is particularly the case with "off premises control".

We've included a copy of our book <u>Insight On Rules</u> with each DRC19Ø system. This book is a reprint of all the <u>Insight On Rules</u> articles published in Radio World Newspaper. You may want to review the sections of this book that relate to remote control.

The installer should also review a current copy of the following sections of the FCC rules:

73.51	AM Station Power Determination
73.57	AM Remote Reading Antenna or Common Point Ammeters
73.62	AM Directional Antenna System Tolerances
73.69(a)(2)	Antenna Monitor Requirements with Remote Control
73.267	FM Station Power Determination
73.293	Use of FM Subcarriers
73.295(a)	Definition of SCA
73.319	FM Subcarrier Tehnical Standards
73.663	TV Station Power Determination
73.665	Use of TV Subcarriers
73.667(a)	Definition of TV SCA
73.932	EBS Monitor and Generator Requirements
73.933	EBS Operation During National Emergency
73.935	EBS Operation During State or Local Emergency
73.936	EBS Operation During State Emergency
73.937	EBS Operation During Local Emergency
73.94Ø(j)	EBS Generator Switch Guard Requirement
73.1215	Specifications for Indicating Instruments
73.123Ø	Station and Operator License Posting Requirements
73.14ØØ	Remote Control Authorizations
73.141Ø	Remote Control Operation
73.15ØØ	Automatic Transmission System
73.156Ø	Operating Power Tolerance
73.157Ø(b)(2)	Modulation Limit on FM Stations Using Subcarriers
73.158Ø	Required Transmission System Inspections
73.169Ø(d)(2)	Commencement of Remote Conrol Operation
73.169Ø(e)(5)	Installation or Replacement of Subcarrier Generator
73.18ØØ	General Requirements Related to Station Log
73.182Ø	More Station Log Requirements
73.184Ø	Retention of Station Logs
73.186Ø	Transmitter Duty Operators
73.187Ø(c)(3)	Chief Operator Review of Station Log
73.3544(b)(4)	Change in Control Point
73.4Ø97	EBS Attention Signals on Automated Stations
74.24	Short-term Operation of Auxiliary Stations
74.4Ø2(a)(7)	Frequencies Reserved For Operational Communications

74.436(c)(7) Group P Frequencies Licensed for Remote Control or ATS
74.434 Remote Control of Remote Pickup Stations
74.451(a) Use of Equipment Type Accepted Under Part 90 Acceptable
74.462(b) TRL Transmitters Limited to 1.5 KHz Deviation
74.464 TRL Frequency Tolerance
74.465 TRL Frequency Measurements
74.467 Posting of Station License
74.482(b) Hourly Identification of TRL Transmitters
74.482(d) International Morse Code Identification of TRL Transmitters
74.531(d) Use of Subcarriers for Remote Control on Aural STLs
74.533(a) Remote Control of Aural STLs
74.535(b) Maximum Deviation For Subcarriers on Aural STLs
74.634 Remote Control of TV STLs
74.734(a) Operator Required at Control Point for LPTV Local Originatio
n
73.933 Remote Control of ITFS Stations

Installation

The DRC19Ø includes copyrighted software in its EPROM. Prior to starting installation of the system, please complete and sign the system registration form as instructed in the FCC and Copyright Notices section of this manual.

When unpacking the DRC19Ø remote control system, first remove the cover of each unit. Insure that all circuit boards are firmly in place and that all connectors are firmly in place. If the system was shipped with the battery backup option, one of the battery leads was disconnected for shipping. This battery lead should be connected when the system is ready for installation.

WARNING

Once the battery or line cord is connected, the power supply portion of the DRC19Ø has high voltage present. Be sure to observe safety precautions!

Once you are satisfied that all the pieces are connected together properly, you can apply AC power by plugging the supplied AC cord into the rear panel and a AC outlet. The rear panel is marked with the required line voltage. A "site-channel" message should appear on the front panel LCD. This initial power-up procedure should be repeated for each unit provided as part of the system.

Premliminary Communications Test

Once each unit has been powered up, try connecting a communications link between the units. A twisted pair cable with a 25 pin D connector on each end is provided to simulate a telephone line, if the system is set up for wire line operation. This should be plugged into J21 of each unit. If the system is set up for other communications circuits, similar simulations can be set up (ie, a coax from subcarrier out to subcarrier in of the other unit). The twisted pairs from each unit should be connected together, allowing each unit to transmit and receive data to and from the other units.

For testing the system, H&F programs the site number in each unit. The unit with the lowest serial number is programmed as site Ø. The next lowest site number is programmed as site 1. This process continues until all units are covered. Note that the site number of each unit can be changed by the user.

With all units connected together, try keying in a site and channel number into each unit. These are keyed in as a four digit number. For example, keying in Ø123 results in selection of site Ø1, channel 23. The DRC checks to see if the requested site is the same as the site number where the entry is being made. If so, the appropriate channel of the A/D converter is selected and the reading displayed. If the requested channel does not exist in the selected site number (for example, channel 23 in a unit with only one A/D board set up to be channels Ø through 9), a reading of zero is returned. If the selected site is not the same as the site the request is being requested from, the DRC sends a request out to the appropriate site on the communications link. Once that site responds, the reading is displayed.

Note that with no samples connected, you'll get random readings displayed. You should be able to step through the channels using the up arrow and down arrow keys. Once a site and channel are selected, a reading should show up within a couple of seconds, assuming the selected site is in the system.

Another preliminary test to complete is the testing of the intercom. On each unit try pressing the COM key. Within a second or so the display on that unit should instruct you to start talking. Talking into the unit with the COM key pressed should cause your voice to be heard at each of the other units. To prevent the failsafe system from dropping out, intercom transmissions are limited to 3Ø seconds each.

Status Panel Test

On systems shipped with status indicators, a test of this portion of the system can also be completed. The status panel is programmable as to which site's status is to be displayed. If a system has a single status transceiver, we will generally set the receive portion of the status transceiver to monitor that site. If a system has two status transceivers, we will set the receive portion of each to monitor the other site (ie, site Ø's LEDs show the status of site 1, and vice versa). In systems with more than two status transceivers, we will generally set each status transceiver to monitor "site 1ØØ". This setting results in the status LEDs at each site displaying the status of the site that is shown on the DRC front panel LCD. With this in mind, grounding pin 2 of J18 should result in the status Ø LED (top left) at the appropriate site changing state. Note that the transmit portion of each status transceiver is programmable through the front panel (in set up mode) as to whether each status line is active high or low. If the line is programmed active high, the LED will light and the STATUS function will return a -1 when the status input is released. The LED will go out and the STATUS function return a Ø when the line is grounded.

Preliminary RS232 Test

If you have a standard CRT terminal, it can be plugged into J22 of a DRC unit. The DRC is shipped with the baud rate set to match the CRT terminal provided. The speed of the DRC or terminal can be changed as desired. Due to a lack of handshaking or turn on delays in the peripheral port of some CRT terminals (such as Qume QVT1Ø1), it is sometimes necessary to run the DRC at 12ØØ bits per second. The Qume QVT1Ø1+ terminal usually shipped with the DRC19Ø includes a buffered printer port with X-ON/X-OFF handshaking, allowing the DRC19Ø and the terminal main and peripheral ports to all be set to 96ØØ bits per second. Terminal and printer interface are covered later in this section. Other CRT data characteristics include:

1 start bit, always space
7 data bits
1 parity bit, always mark (marking parity)
1 stop bit, always mark

Once again, DRC systems that were shipped with terminal and printer should require no changes in the set up of the DRC, the terminal, or the printer.

On pressing the reset button on the rear panel of a DRC unit, a sign on message should appear on the CRT. Typing the sample one line command listed below should cause a meter reading to be displayed, substituting the appropriate site number for S.

```
DISPLAY METER$(S,Ø)
```

Installation

Once these functions on the DRC have been tested, it is time to start the actual installation. It is suggested that each DRC19Ø be rack mounted so that the display is at eye level, or slightly above. The display contrast is optimum when the display is viewed from slightly below horizontal. The following tables indicate the connections to be made to the rear panel connectors.

Metering & Control Connections

The metering inputs are isolated from ground. The metering sample can be up to 1ØØ volts above or below ground (1ØØ volts maximum common mode voltage). The sample voltage itself must be less than 2 volts maximum. If the sample voltages to be read are higher than the allowed 2 volts, external voltage dividers can be added to each sample, or a voltage divider can be added to the A/D converter board (sockets to plug in voltage divider resistors are provided on each A/D board). This voltage divider will be applied to all samples presented to that board. Refer to the A/D board component placement drawing for placement of the series (R2Ø) and shunt (R21) resistors in the voltage divider. To allow for a 2Ø volt sample, the series resistor might be 9 K while the shunt resistor is 1 K. While the actual value of the resistors is non-critical, the stability of the resistors is quite critical. It is suggested that low tempco resistors be used (such as the 5 ppm/C degree resistors used in the reference voltage divider). Prior to connecting samples, each sample voltage should be measured with the highest expected sample to insure that the sample voltage and the 1ØØ volt common mode voltages are not exceeded.

The control outputs are open collector outputs. When the output is activated, the output is pulled to ground. When the control is released, the maximum voltage applied to the control output should be 3Ø volts or less. When the output is active, the maximum current should not exceed 5ØØ mA. Again, each control line should be tested prior to connecting it to the DRC.

Once it has been determined that the sample and control lines are within requirements, they can be connected to the DRC. Provided with the DRC are the required connectors, connector pins, connector shells, and a pin insertion/extraction tool. The following table lists the supplied parts and the suggested crimp tool. The use of shielded cables for all connections to the DRC is suggested to insure immunity from high RF fields. The DRC has been tested in 1 MHz fields exceeding 5 volts/meter and high VHF RF fields (the test site includes 4 FM stations and a TV station) without shielded cables with no problem, but shielded cables are good insurance. The shield (or drain wire) of the cable should be connected to pin 1 of the connector.

We have not had any problems with RF at FM stations. On AM stations with extremely high RF fields (ie, 50 KW TPO, open panel phasor a couple of feet away and one tower right outside the building), it may be necessary to add external low pass filters on some lines. Metering and control lines that go to monitoring equipment in the same equipment rack (such as an antenna monitor) have not required additional filtering. Control, metering, and communications lines leaving the equipment rack that the DRC is mounted in may require external filtering. If there is a problem with unstable readings when the station's RF is brought up, or the communications link appears to fail when the RF is brought up, simple LC filters can be put in the offending lines. A series 10 uH choke with a 0.1 uF capacitor to ground (on the DRC side of the choke) has been successful in curing RF problems. The filter assembly can be placed anywhere in the same equipment rack as the DRC. Shielded cables should be used between the DRC and the filter assembly. If resistors are substituted for the chokes, and a shunt resistor is added, an external RF filter can also provide voltage division to bring the sample voltage within range of the A/D.

Several sections follow on the additional connections required for system operation.

J21 Connections

J21 connects to the communications system linking the DRC190 units. This link is to be a half duplex (two way, but only one way at a time) voice grade link meeting the requirements of a Bell 3002 circuit with basic conditioning as described in Bell System publication 41004. Telephone lines, subcarrier links and radio links are normally available to meet these requirements. Should a link capable of handling the 1200 Baud data not be available, the DRC data rate can be reduced as required. See the Setup and Calibration Section for information on changing the communications data rate.

Pin	Description
1	Shield
3	Audio input data +
16	Audio input data -
4	Audio output data +
17	Audio output data -
6	Transmit key*
19	Ground (Transmit key return)
7	External Speaker
20	Speaker return
12	Direct Connect Modem Tip
25	Direct Connect Modem Ring
13	Direct Connect Modem Shield

The above list is all that is normally required for connection of the DRC system. If working with a 2 wire telephone line, connect pins 3 and 4 to one side of the line and pins 16 and 17 to the other side of the line. If a 4 wire telephone line or other 4 wire circuit (microwave subcarrier, RF link, etc.) is used, pins 3 and 16 should be connected to the receiver and pins 4 and 17 to the transmitter.

If a subcarrier transceiver inside the DRC190 is used, the lines driven by it should not be connected outside the DRC. For example, if this site is receiving a 110 KHz control subcarrier, the internal subcarrier transceiver puts the received audio on pins 3 and 16. No outside connection should be made to these pins.

The Transmit key* (* indicates the line is active low) line is pulled low whenever this site is transmitting data. This line can be used to key a "TRL" transmitter. Note that the transmit key* output can only sink 40 mA with an open circuit voltage of 15 volts.

An external speaker with a series volume control can be connected to pins 7 and 20. This speaker is driven by the speaker driver amplifier on the processor board. The keyboard clicks and intercom voice will be heard through this speaker. This can be useful at noisey transmitter sites. The front panel speaker remains the microphone for intercom operation. The impedance presented to these pins should not be below 8 ohms. An 8 ohm speaker with a series pot as a volume control is suggested.

The direct connect modem connections are used only if a direct connect modem was ordered. These pins should be connected to the dial-up telephone line so that the Basic applications program can receive and place data calls.

If the DRC was ordered with the "Dual Audio Option", there is an additional audio input and output on each unit. This allows the DRC to be used at an intermediate microwave site in a large microwave system. A unit with the dual audio option has the following connections on J21.

Pin	Description
1	Shield
2	Audio input number 1 +
15	Audio input number 1 -
3	Audio input number 2 +
16	Audio input number 2 -
4	Audio output number 1 +
17	Audio output number 1 -
5	Audio output number 2 +
18	Audio output number 2 -
6	Transmit key*
19	Ground
7	External Speaker
20	Speaker return
12	Direct Connect Modem Tip
25	Direct Connect Modem Ring
13	Direct Connect Modem Shield

Note that an internal subcarrier generator will be drive by audio output number 1 (pin 4 high, pin 17 grounded). An internal subcarrier receiver will drive audio input 2 (pin 3 high, pin 16 grounded).

Subcarrier Transceiver Connections

The subcarrier transceiver input and output connections appear above the reset button on the rear panel. The left BNC connector is the subcarrier output, the right BNC connector is the subcarrier input.

If a subcarrier is being used as a control uplink, the subcarrier output at the studio unit should be connected to the multiplex input on the STL transmitter, and the subcarrier input on the transmitter site unit should be connected to the multiplex output of the STL receiver.

If a subcarrier is being used as a metering downlink, the subcarrier output at the transmitter site should be connected to the exciter SCA input (connection for external SCA generator), and the subcarrier input at the studio should be connected to a composite output of the modulation monitor or wideband receiver.

Various portions of the subcarrier system can be replaced with external subcarrier equipment, if desired. For example, some FM exciters have a built in SCA generator. The audio output from the DRC190 could be connected directly to the audio input of this SCA generator. In addition, the modulation monitor/subcarrier demodulator combination at the studio could be replaced with a standard SCA (background music) receiver.

Since STL and exciter specifications vary, some adjustment of the subcarrier transceiver boards will probably be required. The adjustments require the use of an oscilloscope, a digital voltmeter and a frequency counter.

The only control that should require adjustment will be the subcarrier output level (R4). With an oscilloscope connected to the output of the subcarrier generator, adjust R4 for the level required by the STL or exciter multiplex input. STL equipment supplied by Moseley Associates requires a subcarrier level of 1.5 volts P-P. STL equipment supplied by Micro Controls requires a subcarrier level of 1 volt RMS.

Other subcarrier transmitter controls include:

Course Frequency - R6: Adjust this control to the approximate subcarrier frequency desired.

Fine Frequency - R7: Adjust this control to the exact subcarrier frequency desired.

Deviation - R9: Adjust this control to the desired deviation. Deviation is normally set to 1 KHz/Volt. To make this adjustment, ground the audio input and count the output frequency. Apply 5 volt DC (from the DRC logic supply) to the input and adjust R1 so the frequency is 5 KHz lower than the previously measured value. With the subcarrier deviation control adjusted in this manner, the standard 0 dBm output level of the modem in the DRC190 will result in +/- 1 KHz peak deviation of the subcarrier. Note that once the deviation is set, adjustment of the subcarrier frequency will result in the same percentage deviation. Multiplying the subcarrier frequency by two results in double the deviation. Since the subcarrier demodulator is designed for a constant frequency deviation (rather than a constant percentage deviation), the deviation control will have to be readjusted after a frequency change to maintain the desired 1 KHz/volt deviation.

Symmetry - R1: With an oscilloscope conntected to the subcarrier output, adjust R1 for a symmetrical waveform.

Distortion - R2: Connect an audio distortion analyzer to the subcarrier output. Adjust R1 and R2 for minimum distortion (minimum subcarrier harmonic content). It should be possible to get distortion below 1% (sum of harmonics 4Ø dB below subcarrier level).

Subcarrier Receiver Controls include:

Local Oscillator Null - R11: With no input to the subcarrier receiver, adjust R11 for a null in the AC level on pin 2 of the XR22Ø6.

Local Oscillator Frequency - R17: A jumper is supplied on PØ5. This jumper improves the local oscillator null. When adjusting the local oscillator frequency, remove this jumper. When the local oscillator frequency has been adjusted, return the jumper. With no input to the subcarrier receiver, connect a frequency counter to pin 1 of PØ5 and adjust R17 for the desired local oscillator frequency. The desired local oscillator frequency is 455 KHz - SCA frequency. Typical local oscillator frequencies are listed below:

Subcarrier Frequency (KHz)	Local Oscillator Frequency (KHz)
26	429
67	388
92	363
11Ø	345
152	3Ø3

On installing the subcarrier equipment, it would be a good idea to try sending the transmitter directly into the appropriate receiver after the required adjustments have been made (typically only the transmitter output level). Feed a tone into the subcarrier transmitter. If a CRT terminal is available, this can be easily accomplished by typing MODEMTST. A clean sine wave should be apparent at the subcarrier receiver. A distorted sine wave is normally due to excessive deviation of the subcarrier. A noisey signal is normally due to excessively low subcarrier level. The minimum acceptable subcarrier level at the receiver input is approximately 15Ø mV P-P.

Once the direct connection has been shown to work, connect the subcarrier equipment to the normal RF link and check end to end operation. Again, a distorted signal is probably due to over deviation, although this should not have changed since the direct connection test. A noisy signal may be due to crosstalk from the main channel. To simplify manufacturing and adjustment, no input filtering is included in the subcarrier receiver. Instead, the whole base band is hetrodyned up to 455 KHz (and frequency inverted due to the low side local oscillator) and filtered in a standard ceramic filter. The ceramic filter gives excellent rejection characteristics, but it is possible to get interference from the main channel due to actual main channel harmonics (due to non-linearity of the STL or exciter), or various cross products due to the non-linearity in the mixer. Interference from the main channel can be checked by dropping the main channel and seeing if the subcarrier signal cleans up. Interference from the main channel can sometimes be fixed by adjusting the main channel and subcarrier levels (subcarrier injection).

As an aid in checking the system, the front panel speaker of the DRC19Ø is enabled when the calibrate mode is first entered and remains active until another key is pressed, or a speaker off command is received (such as at the end of a CW ID or the end of an intercom message). To enter the calibrate mode, key in 1 2 3 4 followed by a decimal point. Leaving the speaker enabled allows one to hear the received data, test tones, etc. as an aid in checking subcarrier operation.

Once the subcarrier equipment is operating properly, the remainder of the DRC19Ø system should be tested with the subcarrier equipment.

Direct Connect Modem Connections

If a direct connect modem was ordered with the system, the telephone line tip and ring connections appear on J21 pins 12 and 25, as indicated earlier in this section. It is suggested that shielded cable be used for this telephone line connection, and that the shield be connected to pin 13 of J21.

Prior to connecting the equipment to the telephone line, the local telephone company should be notified. You should give the telephone company which line or lines the modem is to be used on, the FCC Registration Number (B468NR-68618-DM-E) and the Ringer Equivalence (Ø.4B). For further information regarding direct connection to dial-up telephone lines, see the FCC Notice section of the manual.

The direct connect modem board also makes some more serial and parallel input/output available.

The additional serial input/output is an RS232 port on J23. This port can be used to drive another printer or terminal from Basic using the statements PRINT#3, DISPLAY#3, INPUT#3, INKEY$(3), LINE(3) and MAXLINE(3). The speed of this serial port is set up in the set up mode. It is identified as port 3 and can be programmed to various speeds between 5Ø and 19.2K bits/second.

The RS232 port at J23 has the following pin out:

1 - Shield
2 - Data from terminal
3 - Data to terminal
4&5 are jumpered giving RTS-CTS handshake
7 - Signal Ground

Many people have trouble remembering which device in an RS232 system is driving which pin. The phrase "Terminal Talks on Two" may help (data from terninal is on pin 2).

The parallel input/output of the direct connect modem card is available on J2Ø. J2Ø has the below pin out when used with the direct connect modem card:

1 - Shield
2 - Input 1
3 - Input 2 (also timer input)
4 - Input 3
5 - Input 4
6 - Input 5
7 - Input 6
8 - Output 1
9 - Output 2
1Ø - Output 3
11 - Output 4
12 - Output 5
13 - Output 6
14 - Output 7
19 - +5 volts thru 1Ø ohms
21 - 33 Signal Ground

Pin 19 is a current limited 5 volt output that may be used to drive external low power logic circuitry (such as CMOS).

These I/O lines are from the 2681 DUART on the modem card. The inputs and outputs are at TTL levels. These lines can be used as status/control lines, although precautions must be taken to protect the I/O lines from voltage transients. In addition, these lines are not supported direclty by Basic. They must be accessed using PEEK and POKE statements. As the firmware of the DRC19Ø is revised, the address of the direct connect modem DUART may vary. To find the address, refer to the symbol table in the Firmware section of the manual. Listed there is the hexadecimal address of DCMDM (Direct Connect Modem). Convert this to decimal and substitute it in the below listed PEEK and POKE statements.

To read the state of an input line from Basic, use the expression (2^N) AND PEEK(DCMDM+13). N is the bit number (N=6 for input 6). The expression will have the value Ø if the input was low, and a non-zero value if the input was high.

To program an output pin high, use the statement POKE DCMDM+14, (2^N). To program an output low, use the statement POKE DCMDM+15, (2^N). For further information, see the 2681 data sheet.

Metering & Control Connections

J1 provides control and metering interface for the first five channels. J2 provides the next five channels, J3 provides the next five channels, and so on.

The below table indicates which channels show up on which connector.

Connector	Channels	A/D Board
J1	Ø - 4	Ø
J2	5 - 9	Ø
J3	1Ø - 14	1
J4	15 - 19	1
J5	2Ø - 24	2
J6	25 - 29	2
J7	3Ø - 34	3
J8	35 - 39	3
J9	4Ø - 44	4
J1Ø	45 - 49	4
J11	5Ø - 54	5
J12	55 - 59	5
J13	6Ø - 64	6
J14	65 - 69	6
J15	7Ø - 74	7
J16	75 - 79	7
J17	8Ø - 84	8
J18	85 - 89	8
J19	9Ø - 94	9
J2Ø	95 - 99	9

Remember the precautions regarding the sample differential and common mode voltages, and the control voltage and current limits. The sample differential voltage is to be less than 2 volts (unless voltage dividers have been added to the A/D board as indicated in the adjustment section). The sample common mode voltage is to be +/- 1ØØ volts from ground. The control circuit open circuit voltage is to be between Ø volts and +3Ø volts. The control circuit short circuit current is to be limited to 5ØØ mA.

The Raise* or Lower* lines are enabled (pulled low) when a site in the system has the appropriate site and channel selected and presses Raise or Lower. These control lines can be used to adjust the reading being displayed or to make mode changes (ie, transmitter on/off, power change, pattern change, etc.).

The ChanOut* lines are enabled (pulled low) when a site in the system has the appropriate site and channel selected. These lines indicate to external equipment that an A/D sample is being taken. These lines can be used to drive external sample selecting equipment, such as the tower and parameter select lines on a directional antenna monitor. The calibration mode includes a sample delay provision to allow for the settling time of an antenna monitor.

The failsafe1* line is enabled (pulled low) when all sites that have failsafe enabled are operating in the system. The failsafe requirements of each site can be set in the setup mode. See the section on calibration and setup. The failsafe1* line will be released if one of the required other DRC units in the system is not responding or if the power supply in this DRC fails. Note that the Failsafe 1* appearing on J1 and J2 are the same signal. If different circuits are to be driven, steering diodes will be required and the current limit (500 mA) will have to be watched closely. The failsafe signal is duplicated on each A/D board, so it is available on each metering/control connector.

The "Failsafe 2*" output is pulled low when this site is "in local". In system set up, the operator is allowed to lock out control from all other sites for up to 599 minutes. This "local" mode is brought out as FS2* to drive an external indicator to avoid the possibility of the site being left with control locked out. Since the control lockout is driven by a timer, no serious harm is done if the site is left with control locked out.

J1 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample Ø	21	-Sample Ø
3	Raise Ø*	22	Ground (Control return)
4	Lower Ø*	23	ChanOut Ø*
5	+Sample 1	24	-Sample 1
6	Raise 1*	25	Ground (Control return)
7	Lower 1*	26	ChanOut 1*
8	+Sample 2	27	-Sample 2
9	Raise 2*	28	Ground (Control return)
1Ø	Lower 2*	29	ChanOut 2*
11	+Sample 3	3Ø	-Sample 3
12	Raise 3*	31	Ground (Control return)
13	Lower 3*	32	ChanOut3*
14	+Sample 4	33	-Sample 4
15	Raise 4*	34	Ground (Control return)
16	Lower 4*	35	ChanOut 4*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J2 Connections

Pin	Description	Pin	Description
1	Snield	2Ø	Ground (Control return)
2	+Sample 5	21	-Sample 5
3	Raise 5*	22	Ground (Control return)
4	Lower 5*	23	ChanOut 5*
5	+Sample 6	24	-Sample 6
6	Raise 6*	25	Ground (Control return)
7	Lower 6*	26	ChanOut 6*
8	+Sample 7	27	-Sample 7
9	Raise 7*	28	Ground (Control return)
1Ø	Lower 7*	29	ChanOut 7*
11	+Sample 8	3Ø	-Sample 8
12	Raise 8*	31	Ground (Control return)
13	Lower 8*	32	ChanOut8*
14	+Sample 9	33	-Sample 9
15	Raise 9*	34	Ground (Control return)
16	Lower 9*	35	ChanOut 9*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J3 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 1Ø	21	-Sample 1Ø
3	Raise 1Ø*	22	Ground (Control return)
4	Lower 1Ø*	23	ChanOut 1Ø*
5	+Sample 11	24	-Sample 11
6	Raise 11*	25	Ground (Control return)
7	Lower 11*	26	ChanOut 11*
8	+Sample 12	27	-Sample 12
9	Raise 12*	28	Ground (Control return)
1Ø	Lower 12*	29	ChanOut 12*
11	+Sample 13	3Ø	-Sample 13
12	Raise 13*	31	Ground (Control return)
13	Lower 13*	32	ChanOut 13*
14	+Sample 14	33	-Sample 14
15	Raise 14*	34	Ground (Control return)
16	Lower 14*	35	ChanOut 14*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J4 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 15	21	-Sample 15
3	Raise 15*	22	Ground (Control return)
4	Lower 15*	23	ChanOut 15*
5	+Sample 16	24	-Sample 16
6	Raise 16*	25	Ground (Control return)
7	Lower 16*	26	ChanOut 16*
8	+Sample 17	27	-Sample 17
9	Raise 17*	28	Ground (Control return)
1Ø	Lower 17*	29	ChanOut 17*
11	+Sample 18	3Ø	-Sample 18
12	Raise 18*	31	Ground (Control return)
13	Lower 18*	32	ChanOut 18*
14	+Sample 19	33	-Sample 19
15	Raise 19*	34	Ground (Control return)
16	Lower 19*	35	ChanOut 19*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J5 Connections

Pin	Description	Pin	Description
1	Shield	20	Ground (Control return)
2	+Sample 20	21	-Sample 20
3	Raise 20*	22	Ground (Control return)
4	Lower 20*	23	ChanOut 20*
5	+Sample 21	24	-Sample 21
6	Raise 21*	25	Ground (Control return)
7	Lower 21*	26	ChanOut 21*
8	+Sample 22	27	-Sample 22
9	Raise 22*	28	Ground (Control return)
10	Lower 22*	29	ChanOut 22*
11	+Sample 23	30	-Sample 23
12	Raise 23*	31	Ground (Control return)
13	Lower 23*	32	ChanOut 23*
14	+Sample 24	33	-Sample 24
15	Raise 24*	34	Ground (Control return)
16	Lower 24*	35	ChanOut 24*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J6 Connections

Pin	Description	Pin	Description
1	Shield	20	Ground (Control return)
2	+Sample 25	21	-Sample 25
3	Raise 25*	22	Ground (Control return)
4	Lower 25*	23	ChanOut 25*
5	+Sample 26	24	-Sample 26
6	Raise 26*	25	Ground (Control return)
7	Lower 26*	26	ChanOut 26*
8	+Sample 27	27	-Sample 27
9	Raise 27*	28	Ground (Control return)
10	Lower 27*	29	ChanOut 27*
11	+Sample 28	30	-Sample 28
12	Raise 28*	31	Ground (Control return)
13	Lower 28*	32	ChanOut 28*
14	+Sample 29	33	-Sample 29
15	Raise 29*	34	Ground (Control return)
16	Lower 29*	35	ChanOut 29*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J7 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 3Ø	21	-Sample 3Ø
3	Raise 3Ø*	22	Ground (Control return)
4	Lower 3Ø*	23	ChanOut 3Ø*
5	+Sample 31	24	-Sample 31
6	Raise 31*	25	Ground (Control return)
7	Lower 31*	26	ChanOut 31*
8	+Sample 32	27	-Sample 32
9	Raise 32*	28	Ground (Control return)
1Ø	Lower 32*	29	ChanOut 32*
11	+Sample 33	3Ø	-Sample 33
12	Raise 33*	31	Ground (Control return)
13	Lower 33*	32	ChanOut 33*
14	+Sample 34	33	-Sample 34
15	Raise 34*	34	Ground (Control return)
16	Lower 34*	35	ChanOut 34*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J8 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 35	21	-Sample 35
3	Raise 35*	22	Ground (Control return)
4	Lower 35*	23	ChanOut 35*
5	+Sample 36	24	-Sample 36
6	Raise 36*	25	Ground (Control return)
7	Lower 36*	26	ChanOut 36*
8	+Sample 37	27	-Sample 37
9	Raise 37*	28	Ground (Control return)
1Ø	Lower 37*	29	ChanOut 37*
11	+Sample 38	3Ø	-Sample 38
12	Raise 38*	31	Ground (Control return)
13	Lower 38*	32	ChanOut 38*
14	+Sample 39	33	-Sample 39
15	Raise 39*	34	Ground (Control return)
16	Lower 39*	35	ChanOut 39*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J9 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 4Ø	21	-Sample 4Ø
3	Raise 4Ø*	22	Ground (Control return)
4	Lower 4Ø*	23	ChanOut 4Ø*
5	+Sample 41	24	-Sample 41
6	Raise 41*	25	Ground (Control return)
7	Lower 41*	26	ChanOut 41*
8	+Sample 42	27	-Sample 42
9	Raise 42*	28	Ground (Control return)
1Ø	Lower 42*	29	ChanOut 42*
11	+Sample 43	3Ø	-Sample 43
12	Raise 43*	31	Ground (Control return)
13	Lower 43*	32	ChanOut 43*
14	+Sample 44	33	-Sample 44
15	Raise 44*	34	Ground (Control return)
16	Lower 44*	35	ChanOut 44*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J1Ø Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 45	21	-Sample 45
3	Raise 45*	22	Ground (Control return)
4	Lower 45*	23	ChanOut 45*
5	+Sample 46	24	-Sample 46
6	Raise 46*	25	Ground (Control return)
7	Lower 46*	26	ChanOut 46*
8	+Sample 47	27	-Sample 47
9	Raise 47*	28	Ground (Control return)
1Ø	Lower 47*	29	ChanOut 47*
11	+Sample 48	3Ø	-Sample 48
12	Raise 48*	31	Ground (Control return)
13	Lower 48*	32	ChanOut 48*
14	+Sample 49	33	-Sample 49
15	Raise 49*	34	Ground (Control return)
16	Lower 49*	35	ChanOut 49*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J11 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 5Ø	21	-Sample 5Ø
3	Raise 5Ø*	22	Ground (Control return)
4	Lower 5Ø*	23	ChanOut 5Ø*
5	+Sample 51	24	-Sample 51
6	Raise 51*	25	Ground (Control return)
7	Lower 51*	26	ChanOut 51*
8	+Sample 52	27	-Sample 52
9	Raise 52*	28	Ground (Control return)
1Ø	Lower 52*	29	ChanOut 52*
11	+Sample 53	3Ø	-Sample 53
12	Raise 53*	31	Ground (Control return)
13	Lower 53*	32	ChanOut 53*
14	+Sample 54	33	-Sample 54
15	Raise 54*	34	Ground (Control return)
16	Lower 54*	35	ChanOut 54*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J12 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 55	21	-Sample 55
3	Raise 55*	22	Ground (Control return)
4	Lower 55*	23	ChanOut 55*
5	+Sample 56	24	-Sample 56
6	Raise 56*	25	Ground (Control return)
7	Lower 56*	26	ChanOut 56*
8	+Sample 57	27	-Sample 57
9	Raise 57*	28	Ground (Control return)
1Ø	Lower 57*	29	ChanOut 57*
11	+Sample 58	3Ø	-Sample 58
12	Raise 58*	31	Ground (Control return)
13	Lower 58*	32	ChanOut 58*
14	+Sample 59	33	-Sample 59
15	Raise 59*	34	Ground (Control return)
16	Lower 59*	35	ChanOut 59*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J13 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 6Ø	21	-Sample 6Ø
3	Raise 6Ø*	22	Ground (Control return)
4	Lower 6Ø*	23	ChanOut 6Ø*
5	+Sample 61	24	-Sample 61
6	Raise 61*	25	Ground (Control return)
7	Lower 61*	26	ChanOut 61*
8	+Sample 62	27	-Sample 62
9	Raise 62*	28	Ground (Control return)
1Ø	Lower 62*	29	ChanOut 62*
11	+Sample 63	3Ø	-Sample 63
12	Raise 63*	31	Ground (Control return)
13	Lower 63*	32	ChanOut 63*
14	+Sample 64	33	-Sample 64
15	Raise 64*	34	Ground (Control return)
16	Lower 64*	35	ChanOut 64*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J14 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 65	21	-Sample 65
3	Raise 65*	22	Ground (Control return)
4	Lower 65*	23	ChanOut 65*
5	+Sample 66	24	-Sample 66
6	Raise 66*	25	Ground (Control return)
7	Lower 66*	26	ChanOut 66*
8	+Sample 67	27	-Sample 67
9	Raise 67*	28	Ground (Control return)
1Ø	Lower 67*	29	ChanOut 67*
11	+Sample 68	3Ø	-Sample 68
12	Raise 68*	31	Ground (Control return)
13	Lower 68*	32	ChanOut 68*
14	+Sample 69	33	-Sample 69
15	Raise 69*	34	Ground (Control return)
16	Lower 69*	35	ChanOut 69*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J15 Connections

Pin	Description	Pin	Description
1	Shield	20	Ground (Control return)
2	+Sample 70	21	-Sample 70
3	Raise 70*	22	Ground (Control return)
4	Lower 70*	23	ChanOut 70*
5	+Sample 71	24	-Sample 71
6	Raise 71*	25	Ground (Control return)
7	Lower 71*	26	ChanOut 71*
8	+Sample 72	27	-Sample 72
9	Raise 72*	28	Ground (Control return)
10	Lower 72*	29	ChanOut 72*
11	+Sample 73	30	-Sample 73
12	Raise 73*	31	Ground (Control return)
13	Lower 73*	32	ChanOut 73*
14	+Sample 74	33	-Sample 74
15	Raise 74*	34	Ground (Control return)
16	Lower 74*	35	ChanOut 74*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J16 Connections

Pin	Description	Pin	Description
1	Shield	20	Ground (Control return)
2	+Sample 75	21	-Sample 75
3	Raise 75*	22	Ground (Control return)
4	Lower 75*	23	ChanOut 75*
5	+Sample 76	24	-Sample 76
6	Raise 76*	25	Ground (Control return)
7	Lower 76*	26	ChanOut 76*
8	+Sample 77	27	-Sample 77
9	Raise 77*	28	Ground (Control return)
10	Lower 77*	29	ChanOut 77*
11	+Sample 78	30	-Sample 78
12	Raise 78*	31	Ground (Control return)
13	Lower 78*	32	ChanOut 78*
14	+Sample 79	33	-Sample 79
15	Raise 79*	34	Ground (Control return)
16	Lower 79*	35	ChanOut 79*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J17 Connections

Pin	Description	Pin	Description
1	Shield	20	Ground (Control return)
2	+Sample 80	21	-Sample 80
3	Raise 80*	22	Ground (Control return)
4	Lower 80*	23	ChanOut 80*
5	+Sample 81	24	-Sample 81
6	Raise 81*	25	Ground (Control return)
7	Lower 81*	26	ChanOut 81*
8	+Sample 82	27	-Sample 82
9	Raise 82*	28	Ground (Control return)
10	Lower 82*	29	ChanOut 82*
11	+Sample 83	30	-Sample 83
12	Raise 83*	31	Ground (Control return)
13	Lower 83*	32	ChanOut 83*
14	+Sample 84	33	-Sample 84
15	Raise 84*	34	Ground (Control return)
16	Lower 84*	35	ChanOut 84*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J18 Connections

Pin	Description	Pin	Description
1	Shield	20	Ground (Control return)
2	+Sample 85	21	-Sample 85
3	Raise 85*	22	Ground (Control return)
4	Lower 85*	23	ChanOut 85*
5	+Sample 86	24	-Sample 86
6	Raise 86*	25	Ground (Control return)
7	Lower 86*	26	ChanOut 86*
8	+Sample 87	27	-Sample 87
9	Raise 87*	28	Ground (Control return)
10	Lower 87*	29	ChanOut 87*
11	+Sample 88	30	-Sample 88
12	Raise 88*	31	Ground (Control return)
13	Lower 88*	32	ChanOut 88*
14	+Sample 89	33	-Sample 89
15	Raise 89	34	Ground (Control return)
16	Lower 89*	35	ChanOut 89*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J19 Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 9Ø	21	-Sample 9Ø
3	Raise 9Ø*	22	Ground (Control return)
4	Lower 9Ø*	23	ChanOut 9Ø*
5	+Sample 91	24	-Sample 91
6	Raise 91*	25	Ground (Control return)
7	Lower 91*	26	ChanOut 91*
8	+Sample 92	27	-Sample 92
9	Raise 92*	28	Ground (Control return)
1Ø	Lower 92*	29	ChanOut 92*
11	+Sample 93	3Ø	-Sample 93
12	Raise 93*	31	Ground (Control return)
13	Lower 93*	32	ChanOut 93*
14	+Sample 94	33	-Sample 94
15	Raise 94	34	Ground (Control return)
16	Lower 94*	35	ChanOut 94*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

J2Ø Connections

Pin	Description	Pin	Description
1	Shield	2Ø	Ground (Control return)
2	+Sample 95	21	-Sample 95
3	Raise 95*	22	Ground (Control return)
4	Lower 95*	23	ChanOut 95*
5	+Sample 96	24	-Sample 96
6	Raise 96*	25	Ground (Control return)
7	Lower 96*	26	ChanOut 96*
8	+Sample 97	27	-Sample 97
9	Raise 97*	28	Ground (Control return)
1Ø	Lower 97*	29	ChanOut 97*
11	+Sample 98	3Ø	-Sample 98
12	Raise 98*	31	Ground (Control return)
13	Lower 98*	32	ChanOut 98*
14	+Sample 99	33	-Sample 99
15	Raise 99	34	Ground (Control return)
16	Lower 99*	35	ChanOut 99*
17	Failsafe 1*	36	Ground (Control return)
18	Failsafe 2* ("Local")	37	Ground (Control return)
19	No connection		

Status Transceiver Connections

The status transceiver appears on the rear panel of the DRC190 on J18 and J19. The use of the status transceiver (and the direct connect modem also) reduces the number of A/D boards that the system can hold, since we run out of room on the rear panel.

Status inputs appear on J18. Each input is pulled to +5 volts through a 10K resistor. Grounding the input activates the input if it is programmed active low, or de-activates the input if it is programmed active high (see the section on set up). These inputs may be driven by a contact closure to ground, an open collector output, a TTL level signal, or a 5 volt CMOS signal. The input must be kept within the 0 volts to 5 volts range to avoid damage to the circuitry.

J19 provides outputs from the status transceiver. These outputs follow the front panel LEDs. If the LED is lit, the output is low. These outputs are driven by a Sprague 5832 peripheral driver. This limits the outputs to a maximum of +40 volts open circuit, and sink a maximum of 100 mA on each output. These outputs are suitable for driving small displays, "Sonalerts", or other devices. If the outputs are used to drive relays, transient suppression diodes are required.

The connections to J18 and J19 are identical, except that J18 has inputs and J19 has outputs. The pin-outs are listed below:

1 - Shield	20 - Status 18
2 - Status 0	21 - Status 19
3 - Status 1	22 - Status 20
4 - Status 2	23 - Status 21
5 - Status 3	24 - Status 22
6 - Status 4	25 - Status 23
7 - Status 5	26 - Status 24
8 - Status 6	27 - Status 25
9 - Status 7	28 - Status 26
10 - Status 8	29 - Status 27
11 - Status 9	30 - Status 28
12 - Status 10	31 - Status 29
13 - Status 11	32 - Status 30
14 - Status 12	33 - Status 31
15 - Status 13	34 - Ground (Status Return)
16 - Status 14	35 - Ground
17 - Status 15	36 - Ground
18 - Status 16	37 - Ground
19 - Status 17	

Note: Early status transceiver boards "skipped" status 18 in the sequence, putting status 20 on pin 20, status 20 on pin 21. . . and, finally, status 18 on pin 33.

J22 Connections

J22 is an RS-232/C subset connector. This connector gives the user access to the Basic interpreter included in the DRC190. The baud rate on the port is programmed through the calibration and setup procedure. Other data parameters include: 1 start bit, 7 data bits, mark parity, 1 stop bit.

Pin	Description
2	Data from terminal
3	Data to terminal
4&5	Shorted to form RTS/CTS loop back
7	Data ground
20	DTR/DCD to DRC

Pin 20 is an input to the DRC190 that can be driven by the terminal plugged into J22. When this is done, a Basic program can see if the terminal is ready to receive data by testing DTR(0) [Data Terminal Ready for port 0]. DTR(0) returns a TRUE (-1) when pin 20 is above +3 volts (TRUE) and returns a FALSE (0) when pin 20 is below -3 volts (FALSE).

Some users plug an external modem into J22. When this is the case, pin 20 can be used as a Data Carrier Detect line. A Basic program can then see if the modem data carrier is present (indicating a call has been received or an outgoing call has made a connection). If DCD is TRUE (carrier indeed detected and pin 20 above +3 volts), DCD(0) [Data Carrier Detect port 0] will be true (-1). If modem carrier data is not present (DCD FALSE or below -3 volts), then DCD(0) will be false (0).

Null Modem Cable

When using an external modem plugged into J22, a null modem cable is required. To take advantage of DCD, the below wiring is suggested. Note that the cable is symmetrical, allowing either end to be connected to J22 of the DRC, and either end of the cable connected to the modem.

```
 1 --------------------- shield --------------------- 1
 2 --------------------- Data > --------------------- 3
 3 --------------------- Data < --------------------- 2
 7 --------------------- Ground --------------------- 7
 8 ---------------------  DCD < --------------------- 20
20 ---------------------  DCD > --------------------- 8
```

Connector Supplies

The below listed connector parts are supplied or are available from H&F.

H&F P/N	Description	Manufacturer Part Number
2100-0016	25 pin male D connector	AMP 205208-1
2100-0017	pins for D connectors	AMP 205202-4
2100-0018	25 pin connector back shell	AMP 205718-1
2100-0091	37 pin male D connector	AMP 205210-1
2100-0092	37 pin connector back shell	AMP 205731-1
2100-0093	pin insertion/extraction tool	AMP 91067-2
2100-0094	crimp tool (not supplied)	AMP 29004-1

Connecting the DRC19Ø to Antenna Monitors

The existence of the CHANOUT* lines on the DRC19Ø simplifies connections to an antenna monitor in directional AM stations. In most cases, no interface circuitry is required. The loop current indication output of the antenna monitor drives all the loop current sample inputs of the DRC19Ø (sample inputs all bridge the single antenna monitor output). The phase angle indication output of the antenna monitor drives all the phase angle sample inputs of the DRC19Ø. The CHANOUT* lines directly drive the tower select lines of the antenna monitor.

A wiring example for a five tower DA using a Potomac Instruments AM-19(2Ø4) antenna monitor is shown below. The five loop currents will show up on the DRC19Ø channels 1Ø through 14. The five phase angles will show up on channels 15 through 19. It is suggested that the sample lines that need to be bridged be so connected in the DRC19Ø connectors. This minimizes the number of conductors required in the cable.

It is suggested that R6 (phase sample adjust) on the AM19 be adjusted for a sample voltage of +1.8ØØ volts when the +18Ø.Ø degree calibrate button on the AM19 is pressed. With the antenna monitor power off, the mechanical zero (assuming an AM19 instead of an AM19D) of each meter should be adjusted to exactly zero. With the power on and the reference tower selected, adjust the zero and phase null controls as instructed in the AM19 manual for an indication of Ø on the DRC19Ø. When the DRC19Ø reads zero, the AM19 meter should also read zero, since the 1ØØ uV resolution of the DRC19Ø exceeds the resolution of the meter on the AM19. Press the 18Ø.Ø degree calibrate button on the AM-19 and adjust the 18Ø degree calibrate control on the AM19. While continuing to hold the 18Ø degree button, calibrate each DRC19Ø phase channel to read 18Ø.ØØ degrees (or, if a negative phase indication is desired, calibrate to -18Ø.Ø degrees). Due to the extremely good linearity of the AM19 meters and the analog to digital converters in the DRC19Ø, the DRC19Ø should indicate phase within Ø.1 degrees of the AM19 indication throughout the range of the AM19. Note also that each antenna monitor channel on the DRC19Ø should have the sample delay set to three, allowing the AM19 one second to settle on each sample. Note, however, that you may wish to modify the below shown schematic to put loop and phase on alternate channels (ie, tower 1 loop current on channel 1Ø, tower 1 phase on channel 11, etc.) and set the sample delays for the first sample of a specific tower to three. The second sample from that tower (ie, phase one on channel 11) could then have a sample delay of zero since meter readings are normally taken in an "upward" direction (and so is "SCAN" in Basic). Once the AM19 has settled on the first reading of a tower, no settling time is required for the second, since both samples are actually available from the AM19 simultaneously.

Similar to the above mentioned procedure for calibrating phase indications, the loop current indications can be calibrated. However, since there is no "1ØØ% calibrate" button on the AM-19, it is suggested that each loop current channel be calibrated using the highest available loop current indication for a particular channel (although, not over 1ØØ%). In addition, R7 of the AM19 should be adjusted to yield 1.ØØØ volts sample when the indicated loop current is 1ØØ.Ø%. As each tower loop current is selected by the DRC19Ø, adjust the AM19 loop current calibrate control to give an indication near 1ØØ%. Then, without modulation, calibrate the DRC19Ø to indicate the same as the AM19. This procedure is repeated on each tower. The sample delay for each should be set to three unless the "alternate sample" scheme of above is implimented. Once calibration is complete, the loop current calibrate control should be returned to its normal setting (generally 100% on the reference tower) and all channels of calibration be checked under each pattern.

A suggested schematic for connecting the DRC190 to the AM19 is shown below. You may wish to modify it for a different number of towers or to use the "alternate sample" scheme of above.

```
DRC190 J3                                                  AM19 J1

__________                                                ______________________
          | 2                                          29 |
+SAMPLE10 |----------.------------------------------------|LOOP SAMPLE
          | 21       |                                 17 |
-SAMPLE10 |--------------.--------------------------------|GROUND
          | 23       |   |                              1 |
CHANOUT10*|-----------------------------------------------|TOWER 1 SELECT
          | 5        |   |                                |
+SAMPLE11 |----------|   |                                |
          | 24       |   |                                |
-SAMPLE11 |--------------|                                |
          | 26       |   |                              2 |
CHANOUT11*|-----------------------------------------------|TOWER 2 SELECT
          | 8        |   |                                |
+SAMPLE12 |----------|   |                                |
          | 27       |   |                                |
-SAMPLE12 |----------|---|                                |
          | 29       |   |                              3 |
CHANOUT12*|-----------------------------------------------|TOWER 3 SELECT
          | 11       |   |                                |
+SAMPLE13 |----------|   |                                |
          | 30       |   |                                |
-SAMPLE13 |----------|---|                                |
          | 32       |   |                              4 |
CHANOUT13*|-----------------------------------------------|TOWER 4 SELECT
          | 14       |   |                                |
+SAMPLE14 |----------|   |                                |
          | 33           |                                |
-SAMPLE14 |--------------|                                |
          | 35                                          5 |
CHANOUT14*|-----------------------------------------------|TOWER 5 SELECT
          | 20                                         14 |
GROUND    |-----------------------------------------------|GROUND
__________|                                               |
```

```
DRC19Ø J4                                              ¦
                                                       ¦
___________                                            ¦
          ¦ 2                                       32 ¦
+SAMPLE15 ¦----------.---------------------------------¦PHASE SAMPLE
          ¦ 21       ¦                              17 ¦
-SAMPLE15 ¦--------------.-----------------------------¦GROUND
          ¦ 23       ¦   ¦                           1 ¦
CHANOUT15*¦--------------------------------------------¦TOWER 1 SELECT
          ¦ 5        ¦   ¦                             ¦
+SAMPLE16 ¦----------¦   ¦                             ¦
          ¦ 24       ¦   ¦                             ¦
-SAMPLE16 ¦--------------¦                             ¦
          ¦ 26       ¦   ¦                           2 ¦
CHANOUT16*¦--------------------------------------------¦TOWER 2 SELECT
          ¦ 8        ¦   ¦                             ¦
+SAMPLE17 ¦----------¦   ¦                             ¦
          ¦ 27       ¦   ¦                             ¦
-SAMPLE17 ¦----------¦---¦                             ¦
          ¦ 29       ¦   ¦                           3 ¦
CHANOUT17*¦--------------------------------------------¦TOWER 3 SELECT
          ¦ 11       ¦   ¦                             ¦
+SAMPLE18 ¦----------¦   ¦                             ¦
          ¦ 3Ø       ¦   ¦                             ¦
-SAMPLE18 ¦----------¦---¦                             ¦
          ¦ 32       ¦   ¦                           4 ¦
CHANOUT18*¦--------------------------------------------¦TOWER 4 SELECT
          ¦ 14       ¦   ¦                             ¦
+SAMPLE19 ¦----------¦   ¦                             ¦
          ¦ 33           ¦                             ¦
-SAMPLE19 ¦--------------¦                             ¦
          ¦ 35                                       5 ¦
CHANOUT19*¦--------------------------------------------¦TOWER 5 SELECT
          ¦ 2Ø                                      14 ¦
GROUND    ¦--------------------------------------------¦GROUND
__________¦                                            ¦
                                                    23 ¦
Note:   For simplicity, AM19 pins are         _________¦+24 VDC
duplicated in this schematic.  Where          ¦     26 ¦
separate wires go to apparently separate      ---------¦RELAY HI SIDE
pins that are numbered the same, they               24 ¦
actually go to non-separate pins.             ---------¦-24 VDC
                                              ¦     14 ¦
                                              ---------¦GND
                                                    13 ¦
  PHASOR GROUNDS THIS LINE TO SELECT DAY PATTERN -------¦DAY SELECT
                                                    15 ¦
  PHASOR GROUNDS THIS LINE TO SELECT NIGHT PATTERN -----¦NIGHT SELECT
                                                    16 ¦
  PHASOR GROUNDS THIS LINE TO SELECT CRITICAL PATTERN---¦
                                                       ¦_____________________
                                                       AM19 J1
```

Terminal & Printer Interface

A variety of terminals and printers can be used with the DRC190. In addition, the use of RS232 to current loop converters or diode matrix circuits allow multiple terminals to be used on one port. Connections and device specific codes are listed below.

Qume QVT101 and QVT101+

The Qume QVT101 or QVT101+ terminal can be connected to the DRC190 J22 using a pin-1-to-pin-1 cable with the first seven pins of one connector connected to the first 7 pins of the other connector. Typically, the interconnection is completed with a six conductor shielded cable with the shield connecting to pin 1 of each connector. A table showing these connections is shown below.

DRC190 J22	Function	QVT101(+) Main Port
1	Shield, Frame Ground	1
2	Terminal data to DRC	2
3	DRC data to terminal	3
4	Request to Send to DRC	4
5	Clear to Send to terminal	5
6	Data Set Ready to terminal	6
7	Signal Ground	7

QVT101 and QVT101+ Setup

The setup menu of the QVT101(+) terminal should be set as shown below. To get into setup, press SETUP while holding the shift key. Use the cursor arrow keys to move the cursor to the field that needs changing. Use the space bar to scroll through the possible entries for each field. Once all changes have been made, press SHIFT-S to save the data in nonvolatile memory. Press SETUP to exit the setup menu.

Full Duplex (FDX), EIA 9600, AUX 9600, Line Feed Off, X-ON Only, Data Bits 8, Bit 8 = 0, Parity Off, Stop Bits 1, Emulation T101+ (or other emulation, if desired), AUX: X-ON Only.

QVT101+ Printer Control Strings

Printer enable and disable codes (DEV1$ and DEV0$) for the QVT101+ are shown below. Since the QVT101+ has a fully buffered printer port, the main port and the serial port may be run at different speeds, if desired, and no "fill" characters are required. It is suggested, however, that both the main and printer ports be run at 9600 bits per second to allow full speed operation of the DRC190 and the terminal.

```
DEV0$ = CHR$(20)+CHR$(27)+CHR$(65)        :REM Printer disable
```

```
DEV1$ = CHR$(18)                              :REM Printer enable
```

QVT1Ø1 Printer Control Strings

The QVT1Ø1 printer port is not buffered. Due to delays in the enabling of the printer port after reception of the printer port enable code, it is necessary to include "fill" characters after the printer enable code. The DEL code is suggested. The below program lines can be used to initialize DEV1$ (printer enable string) and DEVØ$ (printer disable string).

```
FOR N=1 TO 9: A$=A$+CHR$(255): NEXT  :REM Build string of 9 DELs
DEV1$=CHR$(18)+A$                    :REM Build DEV1$
DEVØ$=CHR$(2Ø)+CHR$(27)+CHR$(65)     :REM Build DEVØ$
```

Informer 2Ø3-1ØØ

The Informer 2Ø3-1ØØ can be connected to the DRC19Ø J22 with a six conductor plus shield cable. The table below shows the cable wiring.

DRC19Ø J22	Function	Informer 2Ø3-1ØØ Main Port
1	Shield, Frame Ground	1
2	Terminal data to DRC	2
3	DRC data to terminal	3
4	Request to Send to DRC	4
5	Clear to Send to terminal	5
6	Data Set Ready to terminal	6
7	Signal Ground	7

The Informer 2Ø3-1ØØ has a buffered printer port with full software (XON/XOFF) handshaking. The lines of code below can be used to initialize the printer enable and disable strings.

```
DEV1$=CHR$(27)+"[5i": DEVØ$=CHR$(27)+"[4i"
```

Freedom 1ØØ

The CRT terminal main port is wired to J22 on the DRC19Ø using a cable described below:

DRC19Ø J22	Function	Freedom 1ØØ Main Port
1	Shield, Frame Ground	1
2	Terminal data to DRC	2
3	DRC data to terminal	3
4	Request to Send to DRC	4
5	Clear to Send to terminal	5
6	Data Set Ready to terminal	6
7	Signal Ground	7

In addition, the CRT terminal pins 8 (Carrier Detect Input) is jumpered to pin 2Ø (Data Terminal Ready Output).

If no printer is used with the Freedom 1ØØ, print statements are used in the DRC19Ø program, and the DRC19Ø has been programmed with the peripheral port on and off codes, a dummy plug must be plugged into the Freedom 1ØØ peripheral port to allow it to complete the required handshaking. Failure to install this plug will cause the Freedom 1ØØ to stop operating while waiting for the non-existant printer to be ready for data. This dummy plug has a jumper between pin 6 and 19, and another jumper between pins 8 and 2Ø.

A program line showing the printer port enable and disable codes for the Freedom 1ØØ is shown below:

```
DEV1$=CHR$(27)+CHR$(96):DEVØ$=CHR$(27)+CHR$(97)
```

Current Loop Converters

The DRC19Ø RS232 data lines may be converted to 2Ø mA current loop lines, if desired. The 2Ø mA current loop allows longer lines to be driven due to the higher current capabilities of the drivers and the optical isolation of the receivers, preventing ground loops. In addition, the use of current loops makes it simple to add multiple terminals to the single standard data output of the DRC19Ø. Multiple terminals are merely connected in series.

RS232 uses -12 volts as a MARK condition (idle and data 1) and +12 volts as the SPACE condtion (start bit and data Ø). The original telegraph circuits used current loops with the sounder and key in series. When no data was being sent, a shorting switch around the hand key closed the communications loop. When the loop was broken at any key, all sounders in the loop released. With the introduction of electromechanical teleprinters, the current loop continued to be used. The printers themselves used a 6Ø mA current loop to pull the armature to the selector magnets. Later printers used "holding" magnets where a cam would move the armature up to the magnet. The cam would release the armature at the data sampling time. If there was loop current (line in the MARK condition), the armature would hold. If there was no loop current (SPACE condition), the armature would release. Since the selector magnets did not have to pull in the armature, less current was required. The standard loop current dropped to 2Ø mA. This current was adopted when ASCII mechanical teleprinters became available (such as the Teletype model 33, which followed the model 28, a Baudot printer). The 2Ø mA non-polar current loop has remained a standard for sending data over relatively long twisted pair lines.

In half-duplex circuits, the keyboards and selector magnets of each terminal (printer) were wired in series. Somewhere in the loop a power supply (typically about 15Ø volts DC with a series current limiting resistor) powered the loop. A relatively high voltage supply with a large resistor in series turned the supply into a "constant current generator". In addition, the large resistance in series with the inductance of the selector magnets provided a short time constant. This allows the current in the selector magnets to build up quickly, allowing higher data speeds. All keyboards would idle in the closed or shorted position. Pressing a key would send a start bit (open the line for a bit's time), then follow that with the data bits (five for Baudot, six for typesetting equipment, seven for ASCII), follow that with a parity bit, and finally follow that with a stop bit. The stop bit is always a MARK (current loop closed). It is of variable length with a specified minimum. The minimum varies from one bit time (modern day's "one stop bit") to 1.5 stop bits (standard with Baudot) to two stop bits (standard at 11Ø baud when mechanical printers used).

As electromechanical printers disappeared, the 2Ø mA current loop has remained as a standard. When operating full duplex (as with the DRC19Ø), there is no connection between the send and receive loops. Instead, the central processor receives all data off one loop and echoes it to the other loop. All keyboards are connected in series in one loop. All display devices are connected in series in a second loop. When no data is being transmitted, a constant 2Ø mA flows through each loop. When a character is sent, the loop is interrupted by the start bit. Following the start bit, the loop is closed or open to represent one (MARK) or zero (SPACE).

The DRC19Ø RS232 port can be converted to 2Ø mA current loop using a Black Box CLØ5ØB. This device includes the RS232 to 2Ø mA current loop current converters and a power supply to power each loop (making this an active device instead of a passive device). The CLØ5ØB can be connected to J22 of the DRC19Ø using our standard six condutor plus shield cable with a 25 pin D connector on each end. The shield connects pin 1 to pin 1. The remaining conductors connect pins 2 through 7 of one plug to pins 2 through 7 of the other plug.

The DIP switches in the CLØ5ØB should be set as below. Note that switch B determines the loop current for each loop. With all switches off, the current in each loop is about 2Ø mA when driving a single load. With two terminals in series, it may be necessary to turn all of the B switches on. This would result in 6Ø mA with a single load, but results in 2Ø mA when driving two terminals.

Switch #	Switch A	Switch C
1	on	off
2	off	on
3	on	off
4	off	on
5	on	off
6	on	on
7	off	off
8	on	on
9	off	off
1Ø	on	on

At each terminal, a Black Box CL412B-M can adapt the terminal to 2Ø mA current loop. The DTE/DCE switch on the CL412B-M should be set to DTE (Data Termial Equipment), since it is driving a terminal. The other connections are shown below.

Terminal & Printer Interface

```
 ____________________                           ____________________
|                    |                         |                    |
|                  T+|-------------------------|TXB                 |
|   Black Box        |  Data to terminal>>>>   |      Black Box     |
|                  T-|-------------------------|TXA                 |
|   CLØ5ØB           |                         |      CL412B-M      |
|                  R+|-------------------------|RXB                 |
|                    | <<<Data to DRC19Ø       |                    |
|                  R-|-------------------------|RXA                 |
|____________________|                         |____________________|
```

Current Loop Driving Two Terminals

```
 ____________________                           ____________________
|                    |                         |                    |
|                  T+|-------------------------|TXB                 |
|   Black Box        |  Data to terminal>>>>   |      Black Box     |
|                  T-|--------------------   --|TXA                 |
|   CLØ5ØB           |                    |  | |      CL412B-M      |
|                  R+|-------------------------|RXB                 |
|                    | <<<Data to DRC19Ø  |  | |                    |
|                  R-|---------------   --------|RXA                |
|____________________|              |  |  |  | |____________________|
                                    |  |  |  |
                                    |  |  |  |
                                    |  |  |  |  ____________________
                                    |  |  |  | |                    |
                                    |  |  |  --|TXB                 |
                                    |  |  |    |      Black Box     |
                                    |  |  -----|TXA                 |
                                    |  |       |      CL412B-M      |
                                    |  --------|RXB                 |
                                    |          |                    |
                                    -----------|RXA                 |
                                               |____________________|
```

Brother M15Ø9 Printer

The Brother M15Ø9 printer is normally connected to the DRC19Ø through the printer port on a CRT, although it can be connected to "port #3" when a direct connect modem board is included. In either case, the printer is normally set for 96ØØ bits per second, 7 data bits, no parity, 1 stop bit. It is connected to the DRC19Ø or the terminal with a standard six conductor plus shield cable with a 25 pin D connector on each end. The shield connects to pin one of each connector. The remaining conductors connect to pins two through seven of each connector (two to two, three to three, etc.). Note that when used with the Freedom 1ØØ terminal, a jumper between pins 8 and 2Ø on the terminal end of the printer cable is required.

The normal setting of the M15Ø9 dip switches is shown below.

Switch #	Switch 1	Switch 2	Switch 3
1	on	off	off
2	on	on	off
3	on	on	off
4	off	off	off
5	on	off	off
6	off	off	off
7	on	off	on
8	on	off	off

Introduction

The STX191 is a status expander for the Hallikainen & Friends DRC190 remote control system. The STX191 expands a DRC190 that has internal status from 32 channels of status to 96 channels. It expands a DRC190 that contains no internal status to 64 channels of status.

The STX191 is a 5.25 inch high rack cabinet, very similar to the DRC190. It has 64 LEDs on the front panel. The vertical spacing of the LEDs is 0.4 inch, allowing use of 0.375 inch embossed tape for labels. When used with a DRC190 that has internal status, the top left LED corresponds to status channel 32. When used without status in the DRC190, the top left LED corresponds to status channel 00. The LEDs follow the rear panel status outputs. When an output is low, the corresponding LED is on.

The STX191 may be set up in several ways. These are outlined below.

Status Receiver Site Selection

The receive portion of a status transceiver board may be programmed so that the LEDs and the rear panel connector follow the status of a particular site. The desired site is programmed in binary using the programming jumpers on P05 on the status transceiver board. A jumper that shorts the two adjacent pins indicates the bit is a 0. An open between two adjacent pins indicates a 1. The least significant bit is towards the top of the board. If all jumpers are in place, the receive portion of the status transceiver responds to site 0. If the top jumper is missing, it responds to site 1. If the jumpers are set to site 100 (decimal), the status LEDs follow the site selected by the DRC190 front panel keyboard and display.

Status Transceiver Cascading

The STX191 expands the status capability of the DRC190 by cascading status transceivers. The input portion of the status transceiver always adds additional status inputs to the site as the transceivers are cascaded. The output portion, however, may be connected in two different manners.

Independent Receiver Programming

If the serial data input of a status transceiver board (P02) is connected to P03 of the previous board, the receiver portion of the second board can be independently programmed to respond to a particular site. In this manner, the left board in an STX191 might display the status of site 0, while the right board displays the status of site 1. Note, however, that only the first 32 channels (00 to 31) of the site are displayed. When status transceivers are cascaded in this manner, the number of input channels at this site is increased, but the number of channels displayed for a particular site is not increased. These additional input channels are, however, accessable throughout the system using the Basic STATUS function.

Note that when a status transceiver is used in this manner, the following hardware changes are made:

U1ØA is replaced with a header. A 13Ø ohm resistor is placed between pins 3 and 14 of the header. This resistor provides a pull-up and line terminating resistor for the status clock (SCLK*). In addition, a jumper is added between pins 2 and 6 on this header. This directly cascades the input shift registers on the status transceiver boards. The status transceiver board that actually drives the disk/status interface in the DRC19Ø needs U1Ø to provide a 1 bit delay in the shifted input data (due to a difference in the operation of a 6522 shift register and a 74HC165 shift register). Finally, an additional pair of wires brings ground and +5 volts to pins 7 and 14 of the U1Ø socket, to lower the voltage drop in the interconnecting wiring.

Non-Independent Status Receiver Programming

If the serial data input of a status transceiver board (PØ2) is connected to PØ4 of the previous board, the receiver portion of the second board will indicate status channels 32 through 63 of the site selected by the previous board. A third board cascaded in this manner (for example, 1 in the DRC19Ø and 2 in the STX191) shows status channels 64 through 95. When status transceivers are cascaded in this manner, the address selection hardware on all boards other than the first one must be disabled. This is accomplished by replacing UØ6 with a jumper between pins 1 and 19 (passing through the shift register load pulse), replacing UØ5C with a jumper between pins 5 and 6 (these pins are flared and a jumper placed beneath them), and replacing UØ7 with a jumper between pins 2 and 13. In addition, the input shift register modifications described above are made.

Status Inputs and Outputs

The status inputs to the STX191 accept TTL levels, contact closures to ground, or open collector closures to ground. Each input has a single high speed CMOS (HC) input with a 1ØK pullup resistor to +5 volts. All inputs and outputs are filtered by a ferrite bead and a .Ø56 uF capacitor. The status outputs can sink a maximum of 1ØØ mA. The maximum open circuit voltage is +4Ø volts DC. If the outputs are used to drive relays (or other inductive loads), transient suppression diodes are required.

Installation

The STX191 should be rack mounted immediately below the DRC190 (leaving the cover vents of the DRC190 unobstructed; the STX191 has no special cooling requirements). The supplied cable should be installed between the 8 pin DIN connectors on each unit.

The status inputs to the STX191 accept TTL levels, contact closures to ground, or open collector closures to ground. Each input has a single HC input with a 10K pullup resistor to +5 volts. All inputs and outputs are filtered by a ferrite bead and a .056 uF capacitor. The status outputs can sink a maximum of 100 mA. The maximum open circuit voltage is +40 volts DC. If the outputs are used to drive relays (or other inductive loads), transient suppression diodes are required.

J1 and J2 are the status inputs. J3 and J4 are the status outputs. The pin-outs vary depending upon whether the DRC190 contains a status transceiver. These variations are listed starting on the next page. Recall that status channels start at 00. Status 96 (accessible through Basic) is true if control from other sites is locked out and false if control from other sites is enabled.

STX191 Pin-Outs When Used Witn DRC19Ø With no Status

All Outputs follow site selected by board 1.

J1 (input) & J3 (output)

1 - Shield
2 - Status Ø
3 - Status 1
4 - Status 2
5 - Status 3
6 - Status 4
7 - Status 5
8 - Status 6
9 - Status 7
1Ø - Status 8
11 - Status 9
12 - Status 1Ø
13 - Status 11
14 - Status 12
15 - Status 13
16 - Status 14
17 - Status 15
18 - Status 16
19 - Status 17
2Ø - Status 18
21 - Status 19
22 - Status 2Ø
23 - Status 21
24 - Status 22
25 - Status 23
26 - Status 24
27 - Status 25
28 - Status 26
29 - Status 27
3Ø - Status 28
31 - Status 29
32 - Status 3Ø
33 - Status 31
34 - Ground
35 - Ground
36 - Ground
37 - Ground

J2 (input) & J4 (output)

1 - Shield
2 - Status 32
3 - Status 33
4 - Status 34
5 - Status 35
6 - Status 36
7 - Status 37
8 - Status 38
9 - Status 39
1Ø - Status 4Ø
11 - Status 41
12 - Status 42
13 - Status 43
14 - Status 44
15 - Status 45
16 - Status 46
17 - Status 47
18 - Status 48
19 - Status 49
2Ø - Status 5Ø
21 - Status 51
22 - Status 52
23 - Status 53
24 - Status 54
25 - Status 55
26 - Status 56
27 - Status 57
28 - Status 58
29 - Status 59
3Ø - Status 6Ø
31 - Status 61
32 - Status 62
33 - Status 63
34 - Ground
35 - Ground
36 - Ground
37 - Ground

STX191 Pin-Outs When Used With DRC19Ø With no Status

Status Receivers Independently Programmed

J1 (input) & J3 (output) & J4 (output)	J2 (input)
1 - Shield	1 - Shield
2 - Status Ø	2 - Status 32
3 - Status 1	3 - Status 33
4 - Status 2	4 - Status 34
5 - Status 3	5 - Status 35
6 - Status 4	6 - Status 36
7 - Status 5	7 - Status 37
8 - Status 6	8 - Status 38
9 - Status 7	9 - Status 39
1Ø - Status 8	1Ø - Status 4Ø
11 - Status 9	11 - Status 41
12 - Status 1Ø	12 - Status 42
13 - Status 11	13 - Status 43
14 - Status 12	14 - Status 44
15 - Status 13	15 - Status 45
16 - Status 14	16 - Status 46
17 - Status 15	17 - Status 47
18 - Status 16	18 - Status 48
19 - Status 17	19 - Status 49
2Ø - Status 18	2Ø - Status 5Ø
21 - Status 19	21 - Status 51
22 - Status 2Ø	22 - Status 52
23 - Status 21	23 - Status 53
24 - Status 22	24 - Status 54
25 - Status 23	25 - Status 55
26 - Status 24	26 - Status 56
27 - Status 25	27 - Status 57
28 - Status 26	28 - Status 58
29 - Status 27	29 - Status 59
3Ø - Status 28	3Ø - Status 6Ø
31 - Status 29	31 - Status 61
32 - Status 3Ø	32 - Status 62
33 - Status 31	33 - Status 63
34 - Ground	34 - Ground
35 - Ground	35 - Ground
36 - Ground	36 - Ground
37 - Ground	37 - Ground

J1 & J2 provide 64 status inputs that are identified with this site. J3 provides 32 status outputs that follow the status inputs of one site. J4 provides 32 status outputs that follow the status inputs of another site.

STX191 Pin-Outs When Used With DRC190 With Status
(Inputs and outputs cascaded)

J1 (input) & J3 (output)

1 - Shield
2 - Status 32
3 - Status 33
4 - Status 34
5 - Status 35
6 - Status 36
7 - Status 37
8 - Status 38
9 - Status 39
10 - Status 40
11 - Status 41
12 - Status 42
13 - Status 43
14 - Status 44
15 - Status 45
16 - Status 46
17 - Status 47
18 - Status 48
19 - Status 49
20 - Status 50
21 - Status 51
22 - Status 52
23 - Status 53
24 - Status 54
25 - Status 55
26 - Status 56
27 - Status 57
28 - Status 58
29 - Status 59
30 - Status 60
31 - Status 61
32 - Status 62
33 - Status 63
34 - Ground
35 - Ground
36 - Ground
37 - Ground

J2 (input) & J4 (output)

1 - Shield
2 - Status 64
3 - Status 65
4 - Status 66
5 - Status 67
6 - Status 68
7 - Status 69
8 - Status 70
9 - Status 71
10 - Status 72
11 - Status 73
12 - Status 74
13 - Status 75
14 - Status 76
15 - Status 77
16 - Status 78
17 - Status 79
18 - Status 80
19 - Status 81
20 - Status 82
21 - Status 83
22 - Status 84
23 - Status 85
24 - Status 86
25 - Status 87
26 - Status 88
27 - Status 89
28 - Status 90
29 - Status 91
30 - Status 92
31 - Status 93
32 - Status 94
33 - Status 95
34 - Ground
35 - Ground
36 - Ground
37 - Ground

Calibration and Setup

The DRC190 permits metering channels to be calibrated by establishing a scaling factor that the A/D sample is multiplied by. This scaling factor is determined by the DRC firmware dividing the desired indication by the A/D sample. This floating point scaling factor is stored in Battery Backed RAM (a CSF Thompson Mostek TimeKeeper RAM). In addition, the displayed reading can be either proportional to the A/D sample or proportional to the square of the A/D sample (square law), allowing direct reading of power output from a reflectometer sample. Finally, a three character label and a two character units designator are stored for each channel.

The Setup mode allows changing the site number of this particular unit, the number of the highest site number in the system, the terminal and communications baud rate, the "site delay", fail safe enable, control enable, and the Morse CW identifier code.

The below procedure describes the calibration and setup of the DRC190.

Set Up Mode Select

1. Key in the number sequence "1 2 3 4" and press the decimal point on the front panel keyboard. The "1 2 3 4" sequence makes it difficult to get in the calibrate mode accidentally. The display will indicate you have entered the setup mode and instruct you to use up-arrow to answer yes, and down arrow to answer no. In addition, the front panel speaker is enabled. Any audio received on the communications link will be heard through the speaker, allowing a quick analysis of signal quality. The speaker is disabled on receipt of a speaker mute command (at the end of an intercom or CW ID message) or the pressing of any key.

Note that pressing the COM key at any time when a YES (Up arrow) or NO (down arrow) answer would be expected will immediately drop you out of the calibrate or setup mode. This allows a "quick escape" after desired changes have been made.

Lock out control

1. This is the equivalent of the "local" button on other remote control systems. The display will say "Lock out control from elsewhere?" if control is not currently locked out. It will say "Cancel control lockout?" if control is currently locked out. You may answer yes (up arrow) or no (down arrow) to either of these. Control lock out causes this site to ignore raise or lower commands from <u>other</u> sites. This site can always raise or lower a channel at this site. If you lock out control, the FS2* (local) output of each analog to digital converter board will be pulled low, activating a user supplied alarm. If control lockout is cancelled, the FS2* (local) line is released.

2. If you answer yes to "Lock out control from elsewhere?", the display will say "Lock out control for ### minutes". You can key in a two digit number between 000 and 599. These will replace the "###" as they are keyed in. The control lock out time out keeps you from having to make another trip to the site to release the "local" button. Note that the control lock out time out is based on the system real time clock minutes counter. If you select a lock out

time of one minute, the time out will expire when the next minute of the current time expires (less than a minute from now). Setting a lock out time out of 599 minutes results in a time out of between 598 and 599 minutes. Do not rely on the control lock out to protect you from dangers due to high voltage or high radiation. Always pull the line breakers and discharge all high voltage circuitry before working on it.

3. Note that if you answer yes to "Lock out control" or "Cancel Control Lockout", the system will return to the normal operating mode with site 12, channel 34 selected. You can immediately get back to the control lockout function by just hitting the decimal point key.

Calibration

1. The display asks if you wish to continue calibration. If so, press up arrow. If not, press down arrow and skip to Change Status Logic.

2. The display will say "Calibrate channel # ??". Enter a 2 digit channel number of the channel you wish to calibrate.

3. The display will say "Channel ØØ Label Units OK? ?????". The "ØØ" will be replaced by the channel number you selected in step 2. The "?????" represent a three character label and a two character units designator associated with this channel. The default label and units are "?????". The underlined character is the character to be changed. If no change is required, press up arrow (yes, the character is ok). The underline (cursor) will move to the next character in the label/units string of characters. If the character is not ok, press R (Raise) to increment to the next character (A to B to C etc.), or press L (Lower) to decrement to the last character (C to B to A etc.). Upper and lower case letters, numbers, punctuation and various other symbols are available. Set the first three characters to be the label (a blank space is available if a one or two character label is more appropriate), and set the last two characters to represent the units. Examples are: "FIL V", "EP KV", "IP A", "IA A", "ICP A", "LP1 %", "PH1DG", "LP2 %", "PH2DG", "FG1mA", etc. When the readings are displayed in the normal mode or are displayed or printed from Basic using METER$, the reading would appear as "EP =5.Ø23KV".

4. Once the label and units have been programmed, you will be asked if the current sample delay is ok. The sample delay is the number of analog to digital conversions that are thrown out before the sample is taken. The CHANOUT* output of the A/D board is enabled at the start of the first conversion. If this output is used to drive the tower select on an antenna monitor, the sample is not immediately available. A/D conversions can be thrown out until the reading should be stable. Each conversion takes about a third of a second. Setting sample delay to Ø3 allows the antenna monitor a settling time of 1 second. In addition, the Raise and Lower outputs are pulsed for a period of time corresponding to the sample delay. If the sample delay is Ø3 and a Raise is sent, the appropriate Raise* output of the DRC19Ø will be held low for 1 second. The sample delay must be between ØØ (no delay) and 15 (5 seconds). Setting the sample delay of the first channel of an A/D board (ie, channels ØØ, 1Ø, 2Ø, 3Ø etc.) can result in a slight improvement in the accuracy of readings returned to Basic. If the current delay is ok, hit yes. If not, hit no and then enter the 2 digit delay desired.

5. Once the sample delay has been programmed, you will be asked if the

linear or square law curve is ok. If not, press down-arrow until the appropriate curve is displayed. Then press up-arrow.

6. You will then be asked if a RAISE or LOWER is required during the sample. This will cause a RAISE or LOWER to be generated during the time we are measuring the sample voltage in the calibration routine. It does not cause a RAISE or LOWER to be generated each time the meter is read. Press the up arrow to generate a raise or lower during the time the calibration routine is getting a sample.

7. The reading with the current scaling factor will then be displayed. Note that for the best accuracy, the sample voltage should be at its highest expected voltage. A good way of calibrating the system with an antenna monitor is by pressing the 180 degree phase calibrate button on the antenna monitor prior to answering the RAISE/LOWER question above. If this is correct, press up-arrow. If it is not, press down-arrow. The correct reading can then be entered as a five character number, including the decimal point. The sample voltage being calibrated against must be greater than 10 mV (required to get a resolution of 1% on the A/D) and less than 1.900 volts (required to prevent A/D overrange, which occurs at 2.000 volts). In addition, the calibrated indication cannot be 0. There are some readings that are ideally very close to 0, yet require calibration. Examples include reference tower phase and reflected power. These can be calibrated by presenting the DRC with a non-typical, but proportional sample. The reference tower phase can be calibrated by pressing the 180 degree phase calibration button on the antenna monitor and calibrating the DRC for 180.0 degrees. Reflected power can be calibrated by substituting a sample of forward power and calibrating the DRC for 100%. Depending upon the installation, the substitution of forward power for reflected can be accomplished by "rotating the slug" in the reflectometer, swapping the RF sample outputs on the reflectometer, or swapping the DC samples from the reflectometer.

8. The DRC190 will then ask if you wish to continue calibration. This is the same as step 1. If you answer yes, you will be back there 3. If you answer no, you will advance to "Highest Status #".

Highest Status

This section of system setup allows you to tell the DRC how many status inputs are to be used. This serves two purposes.

The DRC refers to this number when reading status from the status transceivers. If this number is higher than necessary (perhaps 95 when the highest status input used is 31), the updating of status is slightly slower.

The DRC also allows "status" to be set through software as opposed to the standard "hardware". The DRC must know where the end of hardware status inputs is, and where to begin software status. If, for example, we set the highest status number to 15, the status inputs above 15 are ignored. Further, Basic will then allow statements such as STATUS(16)=0 or STATUS(16)=-1. When Basic executes such a statement, the change in status (if any) is transmitted to all sites in the system. These status changes will show up on the status transceiver LEDs and be available to Basic at all sites in the system. This software status is often used to provide parallel binary data to video routing switchers at remote sites.

If the "highest status" is not as desired, reply no (down arrow) until the correct number is displayed, then reply yes (up arrow).

Change Status Logic

1. The display will say "Change status logic?" If you answer no, you will advance to system setup. You should answer no if this site does not have a status transceiver (all those LEDs on the front panel).

2. If you answer yes, the display will say "Status number (ØØ - 95)?" You should key in a two digit number (ØØ to 95) corresponding to the status channel you wish to change.

3. The display will say "Status number ØØ Active High?" or "Status number ØØ Active Low" (assuming you keyed in status channel ØØ in step 2). Whichever is displayed is how that status channel is currently programmed. If it is not the way you want, answer no and the other message will appear. You can keep toggling back and forth between these two messages until you find the one you like. Then, press the yes (up arrow) key to save the selection. If a status line is programmed as active low, a low input will cause the LED corresponding to that channel to light, cause the status output corresponding to that channel to be low, and will cause STATUS(S,C) to return a -1 when this site and channel are selected. If the status line is programmed active low, a high input will cause the LED corresponding to that channel to not light, cause the status output corresponding to that channel to go high (open collector output), and cause STATUS(S,C) to return a Ø when this site and channel are selected. A status line programmed active hign will do the reverse of this.

4. After you answer yes to active high or active low, you'll end up back at step one of this section, allowing you to program other status inputs or to proceed to system set up.

System Set Up

1. The DRC19Ø will then ask if you want to do setup. If you answer no, you will be dumped back into normal operation. If you answer yes, you will proceed to the next step.

2. The DRC19Ø will ask if the programmed maximum site number is correct. If not, answer no and enter the highest site number in the system as a 2 digit number. All sites in the system must be programmed for the same highest site number (called MAXSITE). This should indeed be the highest site number in the system. If you try to make it lower, a higher site number will be disallowed. If you make MAXSITE too high, operation of the system will be needlessly slowed.

3. The DRC19Ø will ask if the programmed site number is correct. If not, answer no and enter the correct site number as a 2 digit number.

4. The DRC19Ø will ask if the programmed site delay is correct. A typical wire line system, or a system with full time subcarrier channels will use a site delay of Ø.Ø5 seconds. A system using a half duplex radio link (such as the Neulink DCL-SX-U provided by H&F) will use a site delay of Ø.2Ø seconds to allow for transmit to receive turnaround delays.

5. The DRC19Ø will ask if the programmed communications baud rate is correct. If not, answer no and the next choice will be displayed. Answer no to see the next choice or answer yes to accept the currently displayed baud

rate. The DRC19Ø normally operates at 1.2 Kbps (1,2ØØ bits per second), but can be slowed down for marginal communications links.

6. The DRC19Ø will ask if the programmed terminal baud rate is correct. Answer in a manner similar to question 5. The DRC19Ø is normally shipped with the terminal baud rate set to 9.6 Kbps.

7. The DRC19Ø will ask if the programmed port 3 baud rate is correct. Port 3 is the RS232 port on the direct connect modem card. This port is accessed using PRINT#3, INPUT#3, INKEY$(3), LINE(3) and MAXLINE(3). It is typically used to drive a printer. The speed of this port (if supplied) is changed in the same manner as the other speeds.

8. The DRC19Ø will then ask if you wish to change the failsafe or control enable programming. If so, answer yes, if not, answer no.

Fail Safe Programming

1. The DRC19Ø will ask you if the failsafe timeout is ok. If, for example, the timeout is set to 5 minutes, the FAILSAFE1* open collector output will be released if this site does not hear from all required sites within 5 minutes. If the displayed failsafe timeout is ok, press up arrow (yes). If not, press down arrow (no), then key in three digits representing the number of minutes the failsafe timer should be set for.

2. The DRC19Ø will ask if you wish to have failsafe enabled from each of the sites in the system. Answer yes for each site that the failsafe is to be enabled on, and no if the failsafe is not to be enabled. The DRC19Ø will release its failsafe output if it does not hear from an enabled site within the failsafe timout period.

Specific Site Control Lockout

1. After the highest site number in the system has had failsafe enabled or disabled, the DRC will ask if Raise or Lower should be enabled from each site. Answer yes or no as apporpriate. This programing feature prevents an "AM rock jock" from shutting down a TV station in the same DRC system.

Morse Code Identifier

1. The DRC19Ø will then ask if you wish to change the CW ID. The CW ID is used to identify UHF telemetry/control radio equipment. If you do wish to change the CW ID, answer yes. Otherwise, answer no to advance to Set Time.
1Ø. The DRC19Ø will first ask how many minutes should be between IDs. Answer ØØ if no IDs are to be done. The typical ID frequency is 6Ø minutes. You can enter anything between Ø1 minutes and 99 minutes.

2. The DRC19Ø will then ask if the ID character over the cursor is that desired. This is similar to setting the labels and units in the calibrate section. Pressing the R key increments to the next available character. Pressing the L key decrements to the previous character. Pressing the Yes key moves the cursor to the next character position. There is room for up to a 16 character ID. The [character is used to indicate the end of the message. If the ID is 16 characters long, no [is needed. A typical ID would be DE K1234[.

Time, Day, Date Set

1. The DRC19Ø will then ask if you wish to set the time, day and date. This allows setting the time, day and date in the Basic program without having to use an ASCII keyboard (some stations have the DRC19Ø drive a receive only printer only for logging). If you answer yes, you'll be asked to enter the time as HHMMSS (hours, minutes, seconds) in 24 hour format. You'll then be asked to enter the day as a number between Ø and 6 where Ø is Sunday, 1 is Monday, etc. You'll then be asked to enter the date in YYMMDD (year, month, date) format.

This complete the DRC19Ø calibration procedure. All calibration and set up information is stored in write protected battery backed RAM and should not require changes once it has been set. The battery backed RAM is non-volatile memory that will survive power failures. The write protect feature protects the information from processor crashes.

DRC19Ø Operation

Previous sections have covered the installation and calibration of the DRC19Ø. This section provides a "simple" set of instructions that can be given to operators to allow them to run the DRC19Ø through the front panel (manual control as opposed to automatic control through a Basic program).

The front panel of the DRC19Ø contains a sixteen key keyboard, a 32 character display, a speaker, and optionally, 32 status indicator LEDs.

Routine meter readings are accomplished by keying in a two digit site number followed by a two digit channel number. For example, if we have the AM transmitter at site 1 and the FM at site 2, the AM plate voltage might be read by keying in "Ø 1 Ø 1" for site Ø1, channel Ø1. A short time later, the second line of the display would show something similar to "EP = 3.1524 KV". To read the FM plate voltage, you might key in "Ø 2 Ø 1" for site Ø2, channel Ø1. A short time later, a similar display would appear (ie, "EP = 5.123 KV"). To avoid having to key in a four digit number each time another channel is to be selected, the up-arrow key can be pressed to increment to the next channel number. The down-arrow key can be pressed to decrement to the next lower channel number.

The R key is used to send a Raise command to the selected site and channel number. The function of the Raise command will vary depending upon what site and channel is selected. For example, most systems will use a Raise while the plate voltage is selected to turn on the transmitter plate voltage. A Raise command sent while the plate current is selected typically puts an AM transmitter on high power, while a Raise plate current command on an FM typically trims the power up.

The L key is used to send a Lower command to the selected site and channel number. A Lower command sent while plate voltage is displayed will typically turn off the plate voltage. A Lower command sent while plate current is displayed will typically change an AM transmitter to low power, or trim an FM transmitter power down.

The numeric keys, the up-arrow, the down arrow, the R key and the L key are all that are required for typical transmitter site control.

The COM key is used as a "press to talk" button by the DRC19Ø intercom circuitry. Shortly after holding down the COM key, the display will say "Talk now, 3Ø seconds left". At this time, should you talk into the speaker on the DRC19Ø front panel, your voice will be heard at all sites in the system. Anyone at those sites can talk back to you by pressing the COM key at that site. Be sure to hold the COM key for as long as you are talking; do not start talking until the display says "Talk now"; Release the COM key when you are finished talking. To avoid system fail safe timeout, intercom transmissions are limited to 3Ø seconds. If it times out, release the COM key and press it again. Start talking when the "Talk now" message appears.

The decimal point key has several functions in system set up and calibration. In routine operation, pressing the decimal point key ("DP") will cause the selected site to send back intercom audio for thirty seconds. For example, if you are currently reading the AM plate voltage at site Ø1, channel Ø1, and you press the DP key, audio from that site will appear shortly. This allows the operator to hear what is going on at a site, hearing bad blower bearings, smoke alarms, etc.

If a front panel status transceiver was provided with the DRC19Ø (32 LEDs on the front panel), on/off status of a transmitter site is available.

Depending upon programming of the system, the front panel LEDs may provide a continuous monitoring of the status of one site or may display the status of one of several selected sites. If a particular site has been programmed into the status transceiver, the front panel LEDs continuously display the status of that site. This typically displays the condition of various relays at the transmitter site, indicating the condition of filament and plate relays, antenna relays and overload relays.

If the status transceiver has been programmed to display the status of one of several sites (site 1ØØ selected), the LEDs will reflect the status of the site selected by the front panel keyboard and LCD. If, for example, the front panel LCD shows site Ø1, channel Ø2, IP = 2.374 A, the status LEDs will show the status of site Ø1. In large multisite systems, the same status lines are assigned to the same status channels at all sites, allowing common labeling of the LEDs.

Operation of the front panel of the DRC19Ø is similar to the operation of older non-programmable remote controls. It is suggested, however, that stations take advantage of the programmability of the DRC19Ø to provide a "friendly" user interface using either a standard CRT terminal with menues, or the Fluke touch screen CRT available from H&F.

Sample Basic Programs

Each unit of the DRC190 system includes a Basic Interpreter and an RS-232 port to allow connection of a CRT or Printer. If the CRT has a peripheral port that can be enabled and disabled under software control, a printer can be connected to this port. When the peripheral port is used in this manner, the peripheral port on and off codes must be programmed using DEV0$ and DEV1$ as outlined in the terminal and printer interface portion of the installation section of this manual.

Each DRC190 unit has 32 Kbytes of RAM at a minimum. This is enough memory to write sophisticated control and logging programs. Additional memory is available as an option to allow for larger programs.

If we wish to continuously update a CRT display with the readings at a remote site, we can use the below listed program. This program assumes we are interested in displaying ten channels of metering (channels 0..9) from site 1 of our system.

```
10 DISPLAY "Channel". "Reading"
20 SCAN 1,0,9
30 FOR C=0 TO 9
40 DISPLAY C, METER$(1,C)
50 NEXT
60 DISPLAY CHR$(27);"*";
70 GOTO 10
```

Note that no spaces are required between words. Although this makes the program a little difficult to read, it can be used to save memory, if desired.

The program works like this:

Line 10 puts up headings labelling the columns.

Line 20 sends a scan command to site 1. It tells site 1 to send back the readings for channels 0 through 9.

Lines 30 and 50 cause line 40 to be executed several times, first with C = 0, then with C = 1, then with C = 2. . . until, on the last time through the loop, C = 9. The program then proceeds to line 60.

Line 40 is executed 10 times by the FOR/NEXT loop formed by lines 30 and 50. This causes the channel (C) and the meter reading (in string form, with labels and units) to be displayed. The comma between C and METER$ causes the reading to be printed over one "comma field" (similar to a tab stop). Each comma field represents 13 character positions.

Line 60 sends the clear screen command (for the Qume QVT101) to the terminal. This command consists of the escape character (CHR$(27)) followed by an "*". The semicolons indicate that nothing appears between these characters (remember a comma moved us over a comma field. . . semicolon doesn't). The last semicolon on the line keeps a carriage return and line feed from being appended to the displayed character string. The lack of a semicolon or comma at the end of line 40 caused a carriage return and line feed to be sent to the terminal after the reading was displayed, causing the next channel and reading to appear on the next line.

Line 70 sends the program back to line 10. This program will run for ever, or at least until a control-C is typed to interrupt the program.

Should this site not be able to get data from site 1, the readings will all appear as "ERR= 1!!". This indicates a loss of communications error with that site. It will not try to communicate with any site until ERR is accessed, such as with the line DISPLAY ERR. This displays the ERRor code and resets ERR.

A simple program to print a log appears below. Text headings, dates, and other niceties have been left off this for simplicity. The program can, of course, be expanded to include headings, limit checking, alarms, etc., as desired.

```
1ØIFTIME<RTTHENRT=-9E9
2ØIFTIME<RT+3ØØØTHEN1Ø
3ØRT=TIME
4ØPRINTTIME$,
5ØFORC=ØTO9
6ØPRINTMETER$(1,C),
7ØNEXT
8ØPRINT
9ØGOTO1Ø
```

On startup, RT and all other variables are zero. Since the time is a positive integer, it is never less than Ø and line 1Ø is not executed. If the time is less than RT+3ØØØ (reading time + 3Ø minutes), then line 2Ø loops the program back to line 1Ø. If 3Ø minutes have elapsed, line 2Ø allows the program to drop through to line 3Ø where RT is updated to the current time. Line 4Ø prints the current time in string form (12 hour am/pm) and the comma leaves the printhead at the next tab stop. Lines 5Ø through 7Ø form a FOR-NEXT loop with C (the metering channel) varying between Ø and 9. Line 6Ø prints the appropriate channel of site 1, tabbing over after printing each. Line 8Ø forces a carriage return and line feed after the last channel has been printed. Line 9Ø loops the program back to the beginning, where it waits for another 3Ø minutes to elapse, or for time to be less than RT.

Line 1Ø waits for time to be less than RT. If this is the case, RT is set to -9E9 (-9 x 1Ø to the 9). Since -9E9 is substantially less than any valid time, it will foce line 2Ø to allow execution of the program to drop through. Since RT is updated to time each time the readings are printed, time will not be less than RT until the beginning of a new day, when time goes to Ø and RT remains at 233ØØØ or so. Line 1Ø forces the program to print readings at the beginning of a new day.

Note that time is stored in HHMMSS format. If time is 12Ø5ØØ (12:Ø5:ØØ PM), TIME+3ØØØ is 1235ØØ (12:35:ØØ PM). This is indeed one half hour from when the readings were last printed. However, what happens if we now add another 3Ø minutes? 1235ØØ+3ØØØ=1265ØØ (12:65:ØØ PM). TIME, however, goes from 125959 to 13ØØØØ, which is greater than 1265ØØ, causing a log print at the top of the hour. This is generally no problem. If you wish, however, you can write a routine in Basic to handle this base 6Ø/base 1Ø problem.

If you are using the DRC19Ø without a disc drive, you may wish to save the logging program in EEPROM. There is about 1 Kbyte of free EEPROM for basic storage. To save your program, type

```
SAVE EEPROM
```

To load a program from EEPROM, type

LOAD EEPROM

The program in EEPROM is automatically loaded and run on system reset.
The SAVE and LOAD EEPROM statements given above use the "boot" area of the EEPROM on the processor board.
If you have a RAM board that is partially loaded with EEPROM, the instructions can be modified to use this EEPROM. For example,

SAVE EEPROM 24576 will save the program in EEPROM residing at 24576 (6ØØØ Hex).

LOAD EEPROM 24576 will load that program into RAM.

If using a program in EEPROM on the RAM board, the boot program could be:

```
1Ø LOAD EEPROM 24576
```

This would cause the DRC19Ø to load and execute the boot program on power up. The boot program would load and execute the actual program (residing in EEPROM at 24576).

If you are using a 64K RAM board (2 banks of 32K with bank 2 being battery backed), and bank 2 is not used for another purpose (variable storage, etc.), bank 2 can be used for non-volatile storage of a program. Since the bottom 8K of RAM is common to all banks, we can use each bank individually for addresses of 8192 and above (decimal). To store a program currently in the main RAM area (bank Ø) to the battery backed bank 2, use the following statement:

SAVE EEPROM 2,8192

When the word EEPROM is not followed by a number, it is assumed that we are referring to the "boot" EEPROM on the processor board. When the word EEPROM is followed by a single number, it is assumed to be the address we wish to start storage at in bank Ø. If the word EEPROM is followed by two numbers separated by a comma, the first is assumed to be the bank of memory to store the program in. The second is assumed to be the starting address.
If you are using a DRC19Ø without a disc drive, you may wish to download Basic programs to the DRC19Ø from another computer. Since the DRC19Ø Basic interpreter compacts each line of code after it is entered, programs cannot be downloaded at high speed. The download program (located in the host computer) should check for the echo of the first character of each line sent to the DRC19Ø. This character is typically the line number. Once this character has been echoed, the DRC19Ø has finished "crunching" the last line entered and is ready for the current line. A suggested program (written in Turbo Pascal) is listed below:

```
PROGRAM ToPCC;
{$U+}

CONST
  StatPort = 32;
  DataPort = 33;

TYPE
  SType = STRING[132];

VAR
  InFile : TEXT;
  NLine,
  FileName.
  Line : SType;
  Question : SType;


  PROCEDURE WCompLin(VAR Line : SType);

   { Write compiled line;  This routine will write a line to the PCC if that
     is what the user selected.  It send the first character, wait for the PCC
     to echo it, and send the rest of the line.  Otherwise, it will just do
     a WRITELN to Compiled.  }

    VAR
      I      : INTEGER;

    FUNCTION InChar : integer;

     { Input a character from the PUNCH: port.   }

      BEGIN
        repeat
        until (Port[StatPort] and 64) = 64;
        InChar := Port[DataPort] AND 127
      END;

    BEGIN {WCompLin}
      for I := 1 to length(Line) do begin
        write(Line[i]);
        repeat
        until (Port[StatPort] and 128) = 128;
        Port[DataPort] := ord(Line[i]) and 127;
        repeat
        until (InChar = (ord(Line[i]) and 127)) or (Line[i] = '@')
      end;
      writeln;
      repeat
      until (Port[StatPort] and 128) = 128;
      Port[DataPort] := 13;
      repeat
```

```
      until InChar = 10;
      if (Line = 'NEW') or (Copy(Line, 1, 5) = 'CLEAR') then
        for I := 1 to 5000 do
    END;

  PROCEDURE WTitle;

    BEGIN
      WRITE(CHR(27), 'E');
      WRITELN('                    ****************************************');
      WRITELN('                    *                                      *');
      WRITELN('                    *         PCC TRANSFER PROGRAM         *');
      WRITELN('                    *                                      *');
      WRITELN('                    ****************************************');
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN('       A William F. Foote/Hallikainen & Friends in house producti
on.');
      WRITELN;
      WRITELN

    END;

BEGIN
  WTitle:
  WRITE('What file do you want to transfer to the PCC?  ');
  READLN(FileName);
  if pos('.', FileName) = 0
    THEN
      FileName := CONCAT(FileName, '.OBJ');
  WRITELN;
  assign(InFile, FileName);
  RESET(InFile);
  WHILE NOT EOF(InFile) DO
    BEGIN
      READLN(InFile, Line);
      IF Line <> ''
        THEN
          WCompLin(Line)
    END;
  WRITELN('Done!'^G);
  CLOSE(InFile)
END.
```

Finally, it may be desired to use an external computer to access DRC190 data through the RS-232 port. This can be accomplished by using a "null modem" cable and having the external computer appear to the DRC190 as a terminal. This "terminal" is then making immediate data requests of the DRC190. To make the DRC190 actually appear to be a terminal to the external computer, the DRC190 RS232 echo should be disabled. This is done using the statement ECHO=0.

When ECHO is non-zero, the DRC190 echoes each character input through the RS232 port. In addition, it adds line feeds to carriage returns and says "OK" after executing an immediate statement. When ECHO is zero, character echo is suppressed, line feeds are not added to carriage returns, and the "OK" message is suppressed.

With a non-zero ECHO, the following would appear on the RS232 port output in response to a meter request:

```
?METER$(1,3)
[CR][LF]
ABC=1.2345DE
[CR][LF]
[CR][LF]
OK
[CR][LF]
```

where [CR] is carriage return and [LF] is line feed, ABC is programmed label, DE is programmed units, and 1.2345 is the reading. Note that ? is the same as typing PRINT.

If ECHO is zero, a similar request would be handled as below:

```
ABC=1.2345DE[CR]
```

Note that this is the way an operator would request to a prompted input request from a typical high level language program. The operator puts in the requested data followed by a carriage return. The DRC190 with ECHO=0 does the same.

A program that demonstrates acquiring data from the DRC190 is shown below.

```
5  DIM M(4) :REM Dimension an array to hold readings
10 S=1 :REM Set site to 1
20 FOR C=0 TO 4 :REM Scan channels 0 thru 4
30 GOSUB 200 :REM Go get the reading
40 M(C)=M :REM Store the reading in an array
50 NEXT :REM Get the next channel
60 FOR N=1 TO 24: PRINT: NEXT :REM Scroll stuff off screen
70 FOR C=0 TO 4: PRINT C,M(C) :REM Display channel and reading
80 NEXT : Display next channel and reading
90 GOTO 10 :REM Go do it again

198 REM Subroutine to get reading.  Call with Site & Channel.
199 REM return with Meter and ERror code.
200 PRINT CHR$(4);"PR#4" :REM Select SSC in slot 4 for output
210 PRINT CHR$(1);"E D" :REM Prevent outgoing data from echoing
220 PRINT "?METER("; S; ","; C; ");CHR$(44);ERR"
222 REM Above line sends request to DRC. DRC gets
```

```
223 REM ?METER(1.3):CHR$(44);ERR  assuming S=1, C=3
23Ø PRINT CHR$(4);"PR#Ø": REM Send further output to screen
231 REM preventing input prompt ("?") from going to DRC
24Ø PRINT CHR$(4);"IN#4": REM Get input from SSC in slot 4
25Ø INPUT M, ER :REM Get meter reading and error code.
26Ø PRINT CHR$(4);"IN#Ø": REM Take further input from keyboard
27Ø RETURN
```

This program assumes an Apple Super Serial Card in Slot 4 of an Apple 2. The card should be set up as below:

Switch 1

1 - Off	Baud Rate = 48ØØ Bits per second
2 - Off	
3 - On	
4 - On	
5 - On	Communications Mode
6 - On	
7 - On	Standard RS-232

Switch 2

1 - On	Data Format: 7 Data, 1 Stop
2 - Off	
3 - Off	Even Parity
4 - Off	
5 - Off	Do not generate LF after CR
6 - Off	Interrupts Off
7 - Off	Standard RS-232

The MODEM-TERMINAL jumper should be pointing UP towards MODEM.

Note that the SSC is set for 48ØØ bits per second. The DRC19Ø should, of course, be set for the same speed using the Calibration and Set Up mode. The choice of RS-232 speed is a trade off. Too high a speed may introduce glitches in long cables between the computer and the DRC19Ø. Too low a speed will result in slow update times for the computer.

Note that program line 25Ø uses an INPUT statement to collect data from the DRC19Ø. The INPUT statement sends a ? prompt to the output device. We keep this from getting to the DRC19Ø by returning output to the screen in line 23Ø. The ? does less harm on the screen than it does to the DRC19Ø. In addition, the INPUT statement echoes received characters to the screen. This results in a bit more garbage on the screen. It is possible to use GET statements to get the DRC19Ø data character by character without echoing to the screen. We found that the time taken by Basic to add the latest character to the string, check for a carriage return and to GOTO back to the GET statement severely limited the speed of communications from the DRC19Ø to the Apple. It may be worthwhile playing with assemmbly routines, other Basic methods, or a compiled langauage to get a clean screen and high speed. Finally, the INPUT statement will hang if it does not get the expected input. A GET statement would also hang when the DRC19Ø stops sending data. Perhaps something similar to INKEY$ with a loop counter could be used to make the program more fail-safe.

Direct Connect Modem Interface Software

The manual on the Cermetek CH177Ø is reprinted in the rear of this manual. Refer to this documentation for further information on using the modem. This section will briefly cover the typical Basic statements used to interface to the modem.

The modem appears as device number 2 to Basic. It can be accessed using the below listed words:

PRINT#2,	Sends data to the modem
INPUT#2,	Receives data from the modem
INKEY$(2)	Gets a single character from the modem
LINE(2)	Returns the line number of the modem device
MAXLINE(2)	Sets the maximum line number used by modem
MDMSPD	Sets the modem speed (3=3ØØ BPS, 12=12ØØ BPS)

All commands sent to the modem must be preceeded by a control-N (CHR$(14)). A few of the typical commands are listed below:

PRINT#2, CHR$(14);"D 'TB(8Ø5)541-Ø2ØØ'"	Originate a call to H&F
PRINT#2, CHR$(14);"A"	Force modem to answer
MDMSPD=3	Change to 3ØØ BPS
MDMSPD=12	Change to 12ØØ BPS

The modem is initiallized to give unsolicited status messages. This is done so that speed change messages are properly received.

DC Basic Precompiler

H&F has written a pre-compiler that allows large programs to be written on a computer using a "structured Basic". The pre-compiler and download program are available from H&F in CP/M and MS-DOS versions. Each pre-compiler package is available for $100.00

The structured Basic does not require line numbers, allows GOTO and GOSUB calls by name, allows long variable names, and supports the IF-THEN-ELSE-ENDIF, WHILE-ENDWHILE, REPEAT-UNTIL, and similar structure constructs. The pre-compiler takes a ".DRC" file (.PCC in CP/M) as the source, and generates a .OBJ object file and a .LST listing file.

The .OBJ file can be downloaded into the DRC190 using the TERM program (TOPCCT in CP/M). The .LST file adds the assigned line numbers to each line of source code and adds a symbol and cross-reference table to the end of the listing. The cross-reference table indicates the 2 character "DRC Basic" variable that the long variable name was assigned, and a list of line numbers that each variable is referenced in.

Type DC (PCCCOMF in CP/M) to start execution. The source code should not have any line numbers. It should have only one statement per line. To refer to a subroutine, the beginning of the subroutine should have the statement "SUBROUTINE @SubName" alone on the first line. @SubName is the name of the subroutine. The name of all labels and subroutines must begin with the "@" character. With the structured constructs described below, you shouldn't have to use a goto. But if you insist, the ability to use one is supported. To define the label to which the goto will go, use the statement "LABEL @LabelName" just before the line you want to go to. Actually, there is no difference between the "@LabelName" just before the line you want to go to. Act ually, there is no
difference between the "LABEL" and "SUBROUTINE" keywords, except that the SUBROUTINE keyword will cause there to be 4 blank lines before it in the listing.

The MS-DOS version (DC) supports command line entry of program parameters. For example: DC MYPROG 10 L+ X+ C+ results in the compilation of MYPROG.DRC using a line increment of 10, generating a Listing file with a cross-reference and using DATA statement compression. Note that this version allows DATA statement compression where the data presented in sequential DATA statements is combined to the maximum allowed line length, causing the program to be smaller. However, the DATA statements must be the last lines in the program. No code can follow the DATA statements if data compression is used. To turn off the listing, cross-reference or compression, subsitute - for the +.

When you want to do a GOTO or GOSUB to a labelled line, just type it out. For example, "GOSUB @GetHisName" is a valid statement.

For clear looping and IF-THEN statements, there are three "structured constructs" provided. They are the IF-ELSE-THEN construct, the REPEAT-UNTIL construct, and the WHILE-ENDWHILE construct. Each statement must appear alone on its own line, i.e., the statement:

```
IF A=B THEN PRINT X
```

is <u>illegal</u>. The proper syntax of an IF-THEN statement is:

```
IF <Condition> {THEN}
    Statement(s)
ENDIF
```

The keyword THEN is optional. For the <Condition>, you must be careful. In the compiled code, this is actually converted into IF NOT(<Condition>) THEN GOTO (The line the ENDIF is on). This means that if you have a variable that is not equal to -1 it will be evaluated "FALSE". Therefore, the statement "IF A" is not equivalent to "IF A<>Ø", it is instead equivalent to "IF A=-1". This is true of the conditions evaluated for the WHILE loop and the REPEAT loop.

The IF statement also has an ELSE option. The syntax of this is as follows:

```
IF <Condition>
     Statement(s)
ELSE
     Statement(s)
ENDIF
```

For example, the source code:

```
IF Name$ = "JOE"
     PRINT "Hello, Joe"
     JoeCounter = JoeCounter + 1
ELSE
     PRINT "Too bad you're not JOE"
ENDIF
```

might be compiled down to the following:

```
1ØØ IF NOT(CD$="JOE") THEN 12Ø
11Ø PRINT "Hello, Joe":CE=CE+1:GOTO 13Ø
12Ø PRINT "Too bad you're not JOE"
```

Note that, in actual compiled code, all of the spaces not within quotes would be removed.

The REPEAT-UNTIL construct is used for a loop that you want executed at least once. The syntax of this loop is as follows:

```
REPEAT
     Statement(s)
UNTIL <Condition>
```

Notice that the <Condition> is evaluated only after the loop has been gone through once. This guarantees that the statement(s) will be executed at least once. For example, the following example:

```
REPEAT
     INPUT "What is your name"; Name$
UNTIL Name$ <> ""
```

might be compiled as follows:

```
1ØØ INPUT "What is your name"; EA$:IF NOT(EA$<>"") THEN 1ØØ
```

The WHILE-ENDWHILE loop has the following syntax:

```
WHILE <Condition>
     Statement(s)
ENDWHILE
```

Notice that here the <Condition> is evaluated before the loop is run through once. If <Condition> is not true, the statement(s) will be bypassed entirely. The following example:

```
WHILE INKEY$ = ""
     DISPLAY "Hit any key to continue...";
     DISPLAY TIME$ CHR$(13);
ENDWHILE
```

might be compiled as follows:

```
1ØØ IF NOT(INKEY$="") THEN 12Ø
11Ø DISPLAY"Hit any key to continue...";:DISPLAY TIME$ CHR$(13);:GOTO 1ØØ
```

All of these structured constructs can be nested (to a depth of 5Ø).

The compiler allows you to use long variable names, up to 2Ø characters. Just be sure that no variable name is the same as a BASIC reserved word, and that it doesn't start with the characters IF, ELSE, ENDIF, REPEAT, UNTIL, WHILE, ENDWHILE, LET, DATA, or REM. You can freely mix upper and lower case letters in a variable name. Internally, the program capitalizes all variable names, i.e., ThisCounter is the same variable as THISCOUNTER. On the variable table, the program prints the variable using the capitalization that you used the first time the variable is encountered in the program. This is true of labels, also. Note that the compiler will capitalize everything not within quotes before it is output to the DRC, therefore "let a$=chr$(13)" is an acceptable line of source code. Also, the word "let" will not appear in the compiled code.

There are two kinds of remarks in our BASIC. The standard REM is supported. This will cause the remark to be included in the object code. But, if you want the text of the remark to contain any lower case letters or spaces, you should enclose the text of the remark in quotes. For example, 'REM William F. Foote' would compile as '1ØØREMWILLIAMF.FOOTE', but 'REM" William F. Foote' would compile as '1ØØREM" William F. Foote"'. Notice that the trailing quote was left off, and that the compiler supplied it. This will work for displays, prints, and assignments to a string literal. For example, 'a$="This string' might compile to '1ØØ DC$="This string"'.

The other kind of remark is the source file remark. This remark will cause no compiled code to be generated. On any line, anything after the character sequence ";@ " will be ignored. NOTE the trailing space! For example:

```
FOR Counter = 1 TO 1Ø                ;@ Do it 1Ø times
```

is a valid source line, and could generate the code:

```
100 FOR CF = 1 TO 10
```

There is an ORG compiler directive. This will allow you to set the current value of the program counter. If you attempt to set the program counter to a value less than it is already, an error will result. This is useful for setting aside places to put extra code, and having standard entry/exit locations that don't change with subsequent revisions. The proper syntax of an ORG directive is:

```
ORG 22000
```

The compiler starts numbering all programs with line 0 + the line increment, and increments the program counter by the increment specified on program startup for each line.

There is a mechanism for continuing one logical line on more than one physical lines. Before each <CR>, have at least one space, and the "&" character. For example:

```
FOR Counter =   &
    1 to 10
```

is a valid source line, and could generate the code:

```
100 FOR CF = 1 TO 10
```

Programming in H&F BASIC

The DRC uses a slightly extended BASIC interpretter written by Microsoft. It is popularly called the 68ØØ 8K BASIC. This section provides an introduction to H&F BASIC. It is not intended to be a detailed course in BASIC programming. It will, however, serve as an excellent introduction for those unfamiliar with the language.

The text here will introduce the primary concepts and uses of H&F BASIC to get you started writing programs.

We recommend that you try each example in this section as it is presented. This will enhance your "feel" for H&F BASIC and how it is used.

After powering up the DRC, it should print a message followed by OK. If not, press and release the RESET button on the rear panel.

> NOTE: All commands to H&F BASIC should end with a carriage return. The carriage return tells H&F BASIC that you have finished typing the command. If you make a typing error, type a back-space to eliminate the last character. Repeated use of back-space will eliminate previous characters. A control-U (U typed with control key pressed) will eliminate the entire line that you are typing.

Now, try typing the following:

```
PRINT 1Ø-4   (end with carriage return)
```

The DRC will immediately print:

```
6

OK
```

The print statement you typed in was executed as soon as you hit the carriage return key. The DRC evaluated the formula after the "PRINT" and then typed out its value, in this case 6.

Now try typing in this:

```
PRINT 1/2,3*1Ø           ("*" means multiply, "/" divide)
```

The DRC will print:

```
.5   3Ø
```

As you can see, the DRC can do division and multiplication as well as subtraction. Note how a " , " (comma) was used in the print command to print two values instead of just one. The comma divides the 132 character line into 1Ø columns, each 13 characters wide. The result is a " , " causes the DRC to skip to the next character field on the printer, where the value 3Ø was printed.

Commands such as the "PRINT" statements you have just typed in are called Direct Commands. There is another type of command called an Indirect Command. Every Indirect Command begins with a Line Number. A Line Number is any integer from Ø to 64ØØØ.

Try typing in the following lines:

```
10 PRINT 2+3
20 PRINT 2-3
```

A sequence of Indirect Commands is called a "Program". Instead of executing indirect statements immediately, H&F BASIC saves Indirect Commands in the DRC memory (RAM). When you type RUN , H&F BASIC will execute the lowest numbered indirect statement that has been typed in, the tne next highest, etc. for as many as were typed in.

Suppose we type RUN now:

```
RUN
```

H&F BASIC will type:

```
 5
-1

OK
```

In the example above, we typed in line 1Ø first and line 2Ø second. However, it makes no difference in what order you type in indirect statements. H&F BASIC always puts them into correct numerical order according to the line number.

If we want a listing of the complete program currently in memory, we type in LIST . Type this in:

```
LIST
```

H&F BASIC will reply with:

```
10 PRINT 2+3
20 PRINT 2-3
OK
```

Sometimes it is desirable to delete a line number of a program altogether. This is accomplished by typing the Line Number of the line we wish to delete, followed only by a carriage return.

Type in the following:

```
10
LIST
```

H&F BASIC will reply with:

```
20 PRINT 2-3
OK
```

We have now deleted line 1Ø from the program. There is no way to get it back. To insert a new line 1Ø, just type in 1Ø followed by the statement we want H&F BASIC to execute.

Type in the following:

```
1Ø PRINT 2*3
LIST
```

H&F BASIC will reply with:

```
1Ø PRINT 2*3
2Ø PRINT 2-3
OK
```

There is an easier way to replace line 1Ø than deleting it and then inserting a new line. You can do this by just typing the new line 1Ø and hitting the carriage return. H&F BASIC throws away the old line 1Ø and replaces it with the new one.

Type in the following:

```
1Ø PRINT 3-3
LIST
```

H&F BASIC will reply with:

```
1Ø PRINT 3-3
2Ø PRINT 2-3
OK
```

It is not recommended that lines be numbered consecutively. It may become necessary to insert a new line betwen two existing lines. An increment of 1Ø between lines is generally sufficient.

If you want to erase the complete program currently stored in memory, type " NEW ". If you are finished running one program and are about to type in a new one, be sure to type " NEW " first. This should be done to prevent mixture of the old and new programs.

Type in the following:

```
NEW
```

H&F BASIC will reply with

```
OK
```

Now type in:

```
LIST
```

H&F BASIC will respond with:

```
OK
```

Often it is desirable to include text along with answers that are printed out, in order to explain the meaning of the numbers.

Type in the following:

```
PRINT "ONE THIRD IS EQUAL TO",1/3
```

H&F BASIC will reply with:

```
ONE THIRD IS EQUAL TO         .333333

OK
```

As explained earlier, including a " , " in a print statement causes it to space over to the next print field before the value following the comma is printed.

If we use a " ; " instead of a comma, the value next will be printed immediately following the previous value.

> NOTE: Numbers are always printed with at least one trailing space. Any text to be printed is always to be enclosed in double quotes (").

Try the following examples:

```
A -  PRINT "ONE THIRD IS EQUAL TO";1/3
     ONE THIRD IS EQUAL TO .333333

     OK

B -  PRINT 1,2,3
      1            2            3

     OK

C -  PRINT 1;2;3
      1  2  3

     OK

D -  PRINT -1;2;-3
     -1  2 -3

     OK
```

Number Format

We will digress for a moment to explain the format of numbers in H&F BASIC. Numbers are stored internally to over six digits of accuracy. When a number is printed, only six digits are shown. Every number may also have an exponent (a power of ten scaling factor).

The largest number that may be represented in H&F BASIC is 1.7Ø141E38, while the smallest positive number is 2.93874E-39.

When a number is printed, the following rules are used to determine the exact format:

1. If the number is negative, a minus sign (-) is printed. If the number is positive, a space is printed.

2. If the absolute value of the number is an integer in the range of Ø to 999999, it is printed as an integer.

3. If the absolute value of the number is greater than or equal to .1 and less than or equal to 999999, it is printed in fixed point notation, with no exponent.

4. If the number does not fall under categories 2 or 3, scientific notation is used.

Scientific notation is formatted as follows: SX.XXXXXESTT . Each X is an integer between Ø and 9. The leading "S" is the sign of the number, a space for a positive one, and a " - " for a negative one. One non-zero digit is printed before the decimal point. This is followed by the decimal point and then the other five digits of the mantissa. An "E" is then printed (for exponent), followed by the sign of the exponent; then the two digits (TT) of the exponent itself. Leading zeroes are never printed; i.e. the digit before the decimal point is never zero. Also, trailing zeroes are never printed. If there is only one digit to print after all trailing zeroes are suppressed, no decimal point is printed. The exponent sign will be "+" for positive and "-" for negative. Two digits of the exponent are always printed; that is zeroes are not suppressed in the exponent field. The value of any number expressed thus is the number to the left of the "E" times 1Ø raised to the power of the number to the right of the "E".

No matter what format is used, a space is always printed following a number. H&F BASIC checks to see if the entire number will fit on the current line. If not, a carriage return/line feed is executed before printing the number.

The following are examples of various numbers and the output format H&F BASIC will place them into:

Number	Output Format
+1	1
-1	-1
6523	6523
-23.46Ø	-23.46
1E2Ø	1E+2Ø
-12.3456E-7	-1.23456E-Ø6
1.234567E-1Ø	1.234567E-1Ø
1ØØØØØØ	1E+Ø6
999999	999999

```
.1              .1
.ØØ1            1E-Ø2
.ØØØ123         1.23E-Ø4
```

A number input from the terminal or a numeric constant used in a BASIC program may have as many digits as desired, up to the maximum length of a program line (111 characters). However, only the first 7 digits are significant, and the seventh digit is rounded up.

```
PRINT 1.234567890123456789Ø
 1.234567

OK
```

PRINT USING

Often the default number formats listed above are not desirable. To limit the number of digits that are printed, or to align decimal points in a table of numbers, the PRINT USING command is available.

The format for a PRINT USING statement is

```
PRINT USING US$ N; M, P
```

where US$ is a string describing the print format (or picture for COBOL fans), and N, M and P are numeric variables that are to be printed using this format.

US$ takes the form of "#####.#####". Each "#" represents a digit before or after the decimal point. In this example, there would be five digits before the decimal point and five after. The sum of the number of digits before and after must not exceed ten, or a syntax error will result. US$ can be a predefined string variable, or can be a literal string included in the statement.

PRINT USING will fit the numeric variables in the print statement to the number format, if possible. One extra space will be allocated to the number prior to the leading digit for the sign. Leading zeroes will be converted to spaces. Trailing zeroes will be printed to fill out the format.

String variables or numerics that cannot be fit into the format will be printed without reformatting. Strings will be unchanged and numbers will be printed using the above listed number formats, if they cannot fit the PRINT USING format.

To demonstrate the use of PRINT USING, try the below listed programs. The first creates a table without using PRINT USING.

```
1Ø FOR N = 1 TO 1Ø
2Ø PRINT 1ØØ*(RND(1)-RND(1))
3Ø NEXT
```

Line 2Ø prints random numbers between -1ØØ and +1ØØ. A typical run might appear

```
RUN
-46.7245
```

```
 38.2826
-5.Ø1143
-1.75Ø31
-24.694
 73.1252
 58.9242
 4.Ø9352
-19.Ø732
-67.6954
```

Notice the lack of decimal point alignment and the lack of "right fill". Try the below program.

```
1Ø FOR N=1 TO 1Ø
2Ø PRINT USING "###.####" 1ØØ*(RND(1)-RND(1))
3Ø NEXT

RUN
 -11.6151
 -24.8873
  13.4291
 -43.3Ø89
 -3Ø.5268
  -3.4926
 -34.Ø3Ø3
 -5Ø.7231
  81.6775
  14.96ØØ
```

Finally, multiple USINGs can be used in the same line. The above program might be modified as below:

```
1Ø FOR N=1 TO 1Ø
2Ø PRINT N; USING "###.#####", N; USING "###.##", N
3Ø NEXT

RUN
 1             1.ØØØØØ   1.ØØ
 2             2.ØØØØØ   2.ØØ
 3             3.ØØØØØ   3.ØØ
 4             4.ØØØØØ   4.ØØ
 5             5.ØØØØØ   5.ØØ
 6             6.ØØØØØ   6.ØØ
 7             7.ØØØØØ   7.ØØ
 8             8.ØØØØØ   8.ØØ
 9             9.ØØØØØ   9.ØØ
 1Ø           1Ø.ØØØØØ  1Ø.ØØ
```

This is a simple implimentation of PRINT USING. It allows the simple formatting of logs and display screens. It does not allow for floating dollar signs and other functions that are available with more advanced PRINT USING, FORMAT or PICTURE statements in other languages.

PRINT, DISPLAY & PRINT#

There are several text I/O devices available on the DRC19Ø. These are listed below:

Ø - Standard Console (terminal plugged into J22)
1 - Standard Printer (plugged into terminal peripheral port)
2 - Direct Connect Modem (optional)
3 - RS232 port on direct connect modem (optional J23)

Use of the word DISPLAY sends output to the console. Use of the word PRINT sends output to the printer plugged into the console peripheral port. Use of the words PRINT # (# is a key word, so spaces are not required) allows direction of output to depend upon a specified number or a numeric variable. The number or numeric variable must be followed by a comma or semicolon to separate it from the remainder of the statement. Sample statements are given below:

PRINT#Ø,"HELLO"	Puts "HELLO" on console
PRINT#1,"HELLO"	Puts "HELLO" on console printer
PRINT#2,"HELLO"	Puts "HELLO" out on modem
PRINT#3,"HELLO"	Puts "HELLO" out on modem RS232 port
N=3:PRINT#N,"HELLO"	Sends "HELLO" to device 3
N=2:PRINT#N	No comma required. Sends CRLF to modem

INPUT

The following is an example of a program that reads a value from the terminal and uses that value to calculate and print a result:

```
1Ø INPUT R
2Ø PRINT 3.14159*R*R
RUN
1Ø
 314.159

OK
```

Here's what's happening. When BASIC encounters the input statement, it waits for you to type in a number. When you do (in the example above, 1Ø was typed), execution continues with the next statement in the program after the variable R has been set (in this case to 1Ø). In the above example, line 2Ø would now be executed. When the formula after the PRINT statement is evaluated, the value 1Ø is substituted for the variable R each time R appears in the formula. Therefore, the formula becomes 3.14159*1Ø*1Ø. or 314.159 .

If you haven't already guessed, what the program above acutally does is to calculate the area of a circle with radius R.

If we wanted to calculate the area of various circles, we could keep re-running the program over each time for each successive circle. But, there's an easier way to do it simply by adding another line to the program as follows:

```
30 GOTO 10
RUN
10
 314.159
3
 28.2743
4.7
 69.3977
CTRL-C

BREAK IN LINE 10
```

Note that when a program is "control-C'd" or stops on an error, the DRC190 will beep the terminal once per second until a key is pressed. This insures that a program crash does not go unnoticed.

INPUT#

Just as with PRINT, input can be specified to come from a specific device. The format for this is INPUT#N, where N is the device number. The number must be followed by a comma or semicolon to separate it from the rest of the statement. Device 0 (the console) is the only device allowed to interrupt a program using control-C. Other devices will have a control-C changed to a space.

```
10 INPUT#2, A$
20 DISPLAY A$
```

The above program gets A$ from the direct connect modem and displays it on the console terminal. Note that line 10 will wait for a carriage return from the specified device. If this never arrives, the program will continue to wait. For this reason, it is suggested that broadcast control programs use INKEY$ instead of INPUT. This allows the device to be checked for a keystroke without hanging at that point in the program.

GOTO

By putting a " GOTO " statement on the end of our program, we have caused it to go back to line 10 after it prints each answer for the successive circles. This could have gone on indefinitely, but we decided to stop after calculating the area of three circles. This was accomplished by typing CTRL-C (control-C) by holding the key marked CONTROL while typing C. This will always stop program execution (unless NOBREAK has been executed).

Numeric Variables

The letter "R" in the program we just used was termed a "variable". A variable name can be any alphabetic character and may be followed by any alphanumeric character. Any alphanumeric character after the first two are ignored. An alphanumeric character is any letter (A-Z) or any number (Ø-9).

Below are some examples of legal and illegal variable names:

LEGAL	ILLEGAL	
A	%	First character must be alphabetic
Z1	Z1A	Variable name too long
TP	TO	Names cannot be reserved words
COUNT	RGOTO	Names cannot contain reserved words

The words used as BASIC statements are reserved for this specific purpose. You cannot use these words as variable names or inside of any variable name. For instance, "FEND" would be illegal because "END" is a reserved word.

The following is a list of the reserved words in H&F BASIC:

ABS, ASC, AND, ATN, BREAKOK, CHR$, CLEAR, CLRSTK, CONT, COS, DATA, DATE, DATE$, DAY, DAY$, DCD, DEBUG, DEF, DEV, DIM, DIR, DISPLAY, DTR, ECHO, EEPROM, END, ERR, EXP, FN, FOR, FRE, GOSUB, GOTO, IF, INKEY$, INPUT, INT, LEFT$, LEN, LET, LINE, LISTEN, LIST, LOAD, LOG, LOWER, LOWER$, MAXLINE, MDMSPD, MDMSTAT, MESSAGE$, METER, METER$, MID$, MODEMTST, MONITOR, NEW, NEXT, NOBREAK, NOT, ON, OR, PEEK, POKE, POS, PRESET, PRINT, RAISE, RAISE$, READ, REM, RESET, RESTORE, RETURN, RIGHT$, RND, RUN, SAVE, SBCMD, SCAN, SGN, SET, SIN, SPC(, SQR, STATUS, STATCLR, STEP, STOP, STR$, SWAP, SYSTAT, TAB(, TAN, THEN, TO, TIME, TIME$, TROFF, TRON, USING, USR, VAL, WAIT, #, +, -, *, /, ^, >, <, =

Besides having values assigned to variables with an input statement, you can also set the value of a variable with a LET or assignment statement.

Try the following examples:

```
A=5

OK

PRINT A,A*2
5    1Ø

OK

LET Z=7

OK

PRINTZ, Z-A

7    2

OK
```

As can be seen from the examples, the "LET" is optional in an assignment statement.

H&F BASIC remembers the values that have been assigned to variables using this type of statement. This "remembering" process uses space in the DRC memory to store the data.

The values of the variables are thrown away and the space in memory used to store them is released when one of four things occur:

1. A new line is typed into the program or an old line is deleted.

2. A CLEAR command is typed in.

3. A RUN command is typed.

4. NEW is typed.

Another important fact is that if a variable is encountered in a formula before it is assigned a value, it is automatically assigned the value zero. Zero is then substituted as the value of the variable in the particular formula. Try the example below:

```
PRINT Q, Q+2, Q*2
 Ø   2   Ø

OK
```

Another statement is the REM statement. REM is short for remark. This statement is used to insert comments or notes into a program. When H&F BASIC encounters a REM statement the rest of the line is ignored.

This serves mainly as an aid for the programmer, and serves no useful function as far as the operation of the program in solving a particular problem. Good program design makes liberal use of remarks.

IF-THEN

Suppose we want to write a program to check if a number is zero or not. With the statements we've gone over so far, this could not be done. What is needed is a statement which can be used to conditionally branch to another statement. The "IF-THEN" statement does just that.

Try typing in the following program: (remember, type NEW first).

```
1Ø INPUT B
2Ø IF B=Ø THEN 5Ø
3Ø PRINT "NON-ZERO"
4Ø GOTO 1Ø
5Ø PRINT "ZERO"
6Ø GOTO 1Ø
```

When this program is typed into the DRC and run, it will ask for a value of B. Type any value you wish. The DRC will come to the "IF" statement. Between the "IF" and the "THEN" portion of the statement there are two expressions separated by a relation.

A relation is one of the following six symbols:

RELATION	MEANING
=	Equal to
>	Greater than
<	Less than
<>	Not equal to
<=	Less than or equal to
=>	Greater than or equal to

The IF statement is either true or false, depending upon whether the two expressions satisfy the relation or not. For example, in the program we just did, if Ø was typed in for B, the IF statement would be true because Ø=Ø. In this case, since the number after the THEN is 5Ø, execution of the program would continue at line 5Ø. Therefore, "ZERO" would be printed and then the program would jump back to line 1Ø (because of the GOTO statement in line 6Ø).

Suppose a 1 was typed in for B. Since 1=Ø is false, the IF statement would be false and the program would continue execution with the next line. Therefore, "NON-ZERO" would be printed and the GOTO in line 4Ø would send the program back to line 1Ø.

H&F BASIC uses the number -1 to represent TRUE and Ø to represent FALSE. Actually, any non-zero number will be interpretted as TRUE. In the above program, the expression B=Ø is replaced by a -1 if B does indeed equal zero. Otherwise it is replaced by a zero. The -1 again represents TRUE and the Ø represents FALSE.

This idea can be used in other than IF-THEN statements. For example, the statement PRINT Ø=1 will print Ø since Ø=1 is FALSE. The statement PRINT 1E3=1E3 will print -1, since 1E3=1E3 is TRUE.

Now try the following program for comparing two numbers:

```
1Ø INPUT A, B
2Ø IF A<=B THEN 5Ø
3Ø PRINT "A IS BIGGER"
4Ø GOTO 1Ø
5Ø IF A<B THEN 8Ø
6Ø PRINT "THEY ARE THE SAME"
7Ø GOTO 1Ø
8Ø PRINT "B IS BIGGER"
9Ø GOTO 1Ø
```

When the program is run, line 1Ø will input two numbers from the terminal. At line 2Ø, if A is greater than B, A<=B will be false. This will cause the next statement to be executed, printing " A IS BIGGER" and then line 4Ø sends the computer back to line 1Ø to begin again.

At line 2Ø, if A has the same value as B, A<=B is true so we go to line 5Ø. At line 5Ø, since A has the same value as B, A<B is false; therefore, we go to the following statement and print "THEY ARE THE SAME". Then line 7Ø sends us back to the beginning again.

At line 2Ø, if A is smaller than B, A<=B is true so we go to line 5Ø. At line 5Ø, A<B will be true so we then go to line 8Ø. "B IS BIGGER" is then printed and again we go back to the beginning.

Try running the program several times. It may make it easier to understand if you try writing your own programs at this time using the IF-THEN statement. Actually trying programs of your own is the quickest and easiest way to understand how H&F BASIC works. Remember, to stop these programs, just type CTRL-C (Control-C).

One advantage of computers is their ability to perform repetitive tasks. Let's take a closer look and see how this works.

Suppose we want a table of square roots from 1 to 1Ø. The H&F BASIC function for square root is "SQR"; the form being SQR(X), X being the number you wish the square root calculated from (the function "argument"). We could write the program as follows:

```
10 PRINT 1, SQR(1)
20 PRINT 2, SQR(2)
30 PRINT 3, SQR(3)
40 PRINT 4, SQR(4)
50 PRINT 5, SQR(5)
60 PRINT 6, SQR(6)
70 PRINT 7, SQR(7)
80 PRINT 8, SQR(8)
90 PRINT 9, SQR(9)
100 PRINT 10, SQR(10)
```

This program will do the job; however, it is terribly inefficient. We can improve the program tremendously by using the IF statement just introduced as follows:

```
10 N=1
20 PRINT N, SQR(N)
30 N=N+1
40 IF N<=10 THEN 20
```

When this program is run, its output will look exactly like that of the 1Ø statement program above. Let's look at how it works.

At line 1Ø we have a LET statement which sets the value of the variable N at 1. At line 2Ø we print N and the square root of N using its current value. It this becomes 2Ø PRINT 1, SQR(1), and this calculation is printed out.

At line 3Ø we use what will appear to be a rather unusual LET statement. Mathematically, the statement N=N+1 is nonsense. However, the important thing to remember is that in a LET statement, the symbol " = " does not signify equality. In this case " = " means "to be replaced with". All the statement does is to take the current value of N and add 1 to it. Thus, after the first time through line 3Ø, N becomes 2.

At line 4Ø, since N now equals 2, N<=1Ø is true so the THEN portion branches us back to line 2Ø, with N now at a value of 2.

The overall result is that lines 2Ø through 4Ø are repeated, each time adding 1 to the value of N. When N finally equals 1Ø at line 2Ø, the next line will increment it to 11. This results in a false statement at line 4Ø, and since there are no further statements to the program, it stops.

<u>FOR-NEXT</u>

This technique is referred to as "looping" or "iteration". Since it is used quite extensively in programming, there are special H&F BASIC statements for using it. We can show these with the following program.

```
10 FOR N=1 TO 10
20 PRINT N, SQR(N)
30 NEXT N
```

The output of the program listed above will be exactly the same as the previous two programs.

At line 10, N is set equal to 1. Line 20 causes the value of N and the square root of N to be printed. At line 30, we see a new type of statement. The "NEXT N" statement causes one to be added to N, and then if N<=10 we go back to the statement following the "FOR" statement. Note that in H&F BASIC, the N in "NEXT N" is optional. By simply using NEXT for line 30, the DRC looks for the previous FOR statement and indexes the variable it specifies. The overall operation then is the same as with the previous program.

Suppose we want to print a table of square roots from 10 to 20, only counting by two's. The following program would perform this task:

```
10 N=10
20 PRINT N, SQR(N)
30 N=N+2
40 IF N<=20 THEN 20
```

Note the similar structure between this program and the previous one for printing square roots for numbers 1 to 10. This program can also be written using the FOR-NEXT loop just introduced.

```
10 FOR N=10 TO 20 STEP 2
20 PRINT N, SQR(N)
30 NEXT
```

Notice that the only major difference between this program and the previous one using the FOR-NEXT loop is the addition of "STEP 2".

This tells H&F BASIC to add 2 to N each time, instead of 1 as in the previous program. If no "STEP" is given in a "FOR" statement, H&F BASIC assumes that one is to be added each time. The "STEP" can be followed by any expression.

Suppose we want to count backwards from 10 to 1. A program for doing this would be as follows:

```
10 I=10
20 PRINT I
30 I=I-1
40 IF I>=1 THEN 20
```

Notice that we are now checking to see that I is greater than or equal to the final value. The reason is that we are now counting by a negative number. In the previous examples, it was the opposite, so we were checking for a variable less than or equal to the final value.

The "STEP" statement previously shown can also be used with negative numbers to accomplish the same purpose. This can be done using the same format as in the other program, as follows:

```
10 FOR I=10 TO 1 STEP -1
20 PRINT I
30 NEXT
```

"FOR" loops can also be "nested". An example of this procedure follows:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 3
30 PRINT I, J
40 NEXT: REM THIS NEXT LOOPS BACK TO J
50 NEXT: REM THIS NEXT LOOPS BACK TO I
```

Notice that the "NEXT J" comes before the "NEXT I". This is because the J-loop is inside of the I-loop. By the way, the colon (:) allows us to put more than one statement on a line. In this case, the second statement on the line was a REMark, which the DRC ignores. REMarks aid the programmer in remembering how the program is supposed to work. The following program is incorrect; run it and see what happens.

```
10 FOR I=1 TO 5
20 FOR J=1 TO 3
30 PRINT I, J
40 NEXT I
50 NEXT J
```

It does not work because when the "NEXT I" is encountered, all knowledge of the J-loop is lost. This happens because the J-loop is "inside" of the I-loop.

Matrices

It is often convenient to be able to select any element in a table of numbers. H&F BASIC allows this to be done through the use of matrices.

A matrix is a table of numbers. The name of this table, called the matrix name, is any legal variable name, "A" for example. The matrix name "A" is distinct and separate from the simple variable "A", and you can use both in the same program.

To select an element of the table, we subscript "A": that is to select the I'th element, we enclose I in parenthesis "(I)" and then follow "A" by this subscript. Therefore, "A(I)" is the I'th element in the matrix "A".

In this section of the manual we will be concerned with one-dimensional matrices only.

"A(I)" is only one element of matrix A, and H&F BASIC must be told how much space to allocate for the entire matrix.

This is done with a "DIM" statement, using the format "DIM A(15)". In this case, we have reserved spaces for the matrix index "I" to go from Ø to 15. Matrix subscripts always start at Ø; therefore, in the above example, we have allowed 16 numbers in matrix A.

If "A(I)" is used in a program before it has been dimensioned, H&F BASIC reserves space for 11 elements (Ø through 1Ø).

As an example of how matrices are used, try the following program to sort a list of 8 numbers with you picking the numbers to be sorted.

```
10 DIM A(8)
20 FOR I=1 TO 8
30 INPUT A(I)
50 NEXT
70 F=0
80 FOR I=1 TO 7
90 IF A(I)<=A(I+1) THEN 140
100 T=A(I)
110 A(I)=A(I+1)
120 A(I+1)=T
130 F=1
140 NEXT
150 IF F=1 THEN 70
160 FOR I=1 TO 8
170 PRINT A(I),
180 NEXT
```

When line 1Ø is executed, H&F BASIC sets aside space for 9 numerical values, A(Ø) through A(8). Lines 2Ø through 5Ø get the unsorted list from the user. The sorting itself is done by going through the list of numbers and upon finding any two that are not in order, we switch them. "F" is used to indicate if any switches were done. If any were done, line 15Ø tells H&F BASIC to go back and check some more.

If we did not switch any numbers, or after they are all in order, lines 16Ø through 18Ø will print the sorted list. Note that a subscript can be any expression.

Matrices are also called arrays. Arrays may hold either numeric or string variables. Use of numeric and string arrays proves very valuable in typical DRC programs. Numeric arrays may hold lower and upper limits for each metering channel, while string arrays may hold labels, units or alarm messages associated with each metering channel.

GOSUB & RETURN

Another useful pair of statements are "GOSUB" and "RETURN". If you have a program that performs the same action in several different places, you can duplicate the same statements for the action in each place within the program.

The "GOSUB-RETURN" statements can be used to avoid this duplication. When a "GOSUB" is encountered, H&F BASIC branches to the line whose number follows the "GOSUB". However, H&F BASIC remembers where it was in the program before it branched. When the "RETURN" statement is encountered, H&F BASIC goes back to the first statement following the last "GOSUB" that was executed. Observe the following program.

```
10 PRINT "WHAT IS THE NUMBER";
30 GOSUB 100
40 T=N
```

```
50 PRINT "WHAT IS THE SECOND NUMBER";
70 GOSUB 100
80 PRINT "THE SUM OF THE TWO NUMBERS IS", T+N
90 STOP
100 INPUT N
110 IF N = INT(N) THEN 140
120 PRINT "SORRY, NUMBER MUST BE AN INTEGER. TRY AGAIN"
130 GOTO 100
140 RETURN
```

What this program does is ask for two numbers which must be integers, and then prints the sum of the two. The subroutine in this program is lines 100 to 130. The subroutine asks for a number, and if it is not an integer, asks for a number again. It will continue to ask until and integer value is typed in.

The main program prints "WHAT IS THE NUMBER". and then calls the subroutine to get the value of the number into N. When the subroutine returns (to line 40), the value input is saved in the variable T. This is done so that when the subroutine is called a second time, the value of the first number will not be lost.

"WHAT IS THE SECOND NUMBER" is then printed, and the second value is entered when the subroutine is called again.

When the subroutine returns the second time, "THE SUM OF THE TWO NUMBERS IS" is printed, followed by the value of their sum. T contains the value of the first number that was entered and N contains the value of the second number.

The next statement is a "STOP" statement. This causes the program to stop execution at line 90. If the "STOP" statement was not included in the program, we would "fall into" the subroutine at line 100. This is undesirable because we would be asked to input another number. If we did, the subroutine would try to return; and since there was no "GOSUB" which called the subroutine, an RG error (Return without Gosub) would occur. Each "GOSUB" executed in a program should have a matching "RETURN" executed later, and the opposite applies, i.e. a "RETURN" should be encountered only if it is part of a subroutine which has been called by a "GOSUB".

Either "STOP" or "END" can be used to separate a program from its subroutines. "STOP" will print a message saying at what line "STOP" was encountered.

In broadcast or other process control applications, we don't really want the program to stop. Instead, we want it to forever repeat its control program. In the above program, this could be accomplished by substituting GOTO 10 for line 130. The program would continue to run (asking for numbers to be added) until a Control-C was typed in. So, we can use "STOP", "END", or "GOTO" to separate the main program from subroutines.

<u>DATA & RESTORE</u>

Suppose you had to enter numbers to your program that didn't change each time the program was run, but you would like it to be easy to change them if necessary. H&F BASIC contains special statements for this purpose, called the "READ" and "DATA" statements.

Consider the following program:

```
10 PRINT "GUESS A NUMBER";
20 INPUT G
30 READ D
40 IF D=-999999 THEN 90
50 IF D<>G THEN 30
60 PRINT "YOU ARE CORRECT"
70 END
90 PRINT "BAD GUESS, TRY AGAIN."
95 RESTORE
100 GOTO 10
110 DATA 1,393,-39,28,391,-8,0,3.14,90
120 DATA 89,5,10,15,-34,-999999
```

This is what happens when this program is run. When the "READ" statement is encountered, the effect is the same as an INPUT statement. But, instead of getting a number from the terminal, a number is read from the "DATA" statements.

The first time a number is needed for a READ, the first number in the first DATA statement is returned. The second time one is needed, the second number in the first DATA statement is returned. When the entire contents of the first DATA statement have been read in this manner, the second DATA statement will be used. DATA is always read sequentially in this manner, and there may be any number of DATA statements in your program.

The purpose of this program is to play a little game in which you try to guess one of the numbers contained in the DATA statements. For each guess that is typed in, we read through all of the numbers in the DATA statements until we find one that matches the guess.

If more values are read than there are numbers in the DATA statements, an out of data (OD) error occurs. That is why in line 40 we check to see if -999999 was read. This is not one of the numbers to be matched, but is used as a flag to indicate that all of the data (possible correct guesses) has been read. Therefore, if -999999 was read, we know that the guess given was incorrect.

The above mentioned "flag" (-999999) is one way to detect the end of data. A more commonly used method is to have the first piece of data read indicate the number of data items to follow (or, often, the number to follow -1, allowing all elements of an array to be used). For example,

```
READ MC             :READ MaxChannel, highest channel used
FOR N=0 TO MC
READ LB$(N), UN$(N) :REM Get label and unit for each
NEXT

DATA 5              :REM Highest channel
DATA "Filament Voltage", "Volts"   :REM Channel 0 label and units
DATA "Plate Voltage", "Kilovolts"  :REM Channel 1
DATA "Plate Current", "Amperes"    :REM Channel 2
DATA "Forward Power", "%"          :REM Channel 3
DATA "Reflected Power", "%"        :REM Channel 4
DATA "Line Voltage", "Volts"       :REM Channel 5
```

Before going back to line 1Ø for another guess, we need to make the READ's begin with the first piece of data again. This is the function of the "RESTORE". After the RESTORE is encountered, the next piece of data read will be the first piece in the first DATA statement again.

DATA statements may be placed anywhere within the program. Only READ statements make use of the DATA statements in a program, and any other time they are encountered during program execution they will be ignored.

Strings

A list of characters is referred to as a "String". H&F, DRC, and THIS IS A TEST are all strings. Like numeric variables, string variables can be assigned specific values. String variables are distinguished from numeric variables by a "$" after the variable name.

For example, try the following:

```
A$="Hallikainen & Friends "

OK
PRINT A$
Hallikainen & Friends

OK
```

In this example, we set the string variable A$ to the string value "Hallikainen & Friends". Note that we also enclosed the character string to be assigned to A$ in quotes.

LEN

Now that we have set A$ to a string value, we can find out what the length of this value is (the number of characters it contains). We do this as follows:

```
PRINT LEN(A$), LEN("HELLO")
 21  5

OK
```

The "LEN" function returns a integer equal to the number of characters in a string.

The number of characters in a string expression may range from Ø to 255. A string which contains Ø characters is called the "NULL" string. Before a string variable is set to a value in the program, it is initialized to the null string. Printing a null string on the terminal will cause no characters to be printed, and the print head (or cursor) will not be advanced to the next column. Try the following:

```
PRINT LEN(Q$);Q$;3
 Ø  3
```

```
OK
```

Another way to create the null string is: Q$="" .
Setting a string variable to the null string can be used to free up the string space used by non-null string variables.

LEFT$

Often it is desirable to access parts of a string and manipulate them. Now that we have set A$ to "Hallikainen & Friends", we might want to print out only the first six characters of A$. We would do so like this:

```
PRINT LEFT$(A$,6)
Hallik

OK
```

"LEFT$" (pronounced "LEFT-STRING") is a string function which returns a string composed of the leftmost N characters of its string argument. Here's another example:

```
FOR N=1 TO LEN(A$):PRINT LEFT$(A$,N):NEXT
H
Ha
Hal
Hall
Halli
Hallik
Hallika
Hallikai
Hallikain
Hallikaine
Hallikainen
Hallikainen
Hallikainen &
Hallikainen &
Hallikainen & F
Hallikainen & Fr
Hallikainen & Fri
Hallikainen & Frie
Hallikainen & Frien
Hallikainen & Friend
Hallikainen & Friends

OK
```

Since A$ has 21 characters this loop will be executed with N=1, 2, 3, ... 18, 19, 29, 21. The first time through only the first character will be printed. The second time through the first two characters will be printed.

RIGHT$

There is another string function called "RIGHT$" which returns the right N characters from a string expression. Try substituting RIGHT$ for LEFT$ in the previous example and see what happens.

MID$

There is also a string function which allows us to take characters from the middle of a string. Try the following:

```
FOR N=1 TO LEN(A$):PRINT MID$(A$,N):NEXT
Hallikainen & Friends
allikainen & Friends
llikainen & Friends
likainen & Friends
ikainen & Friends
kainen & Friends
ainen & Friends
inen & Friends
nen & Friends
en & Friends
n & Friends
 & Friends
& Friends
 Friends
Friends
riends
iends
ends
nds
ds
s

OK
```

MID$ returns a string starting at the Nth posistion of A$ to the end (last character) of A$. The first position of the string is position 1 and the last possible postion of a string is position 255.

Very often it is desirable to extract only the Nth character from a string. This can be done by calling MID$ with three arguments. The third argument specifies the number of characters to return.

For example:

```
FOR N=1 TO LEN A$:PRINT MID$(A$,N,1),MID$(A$,N,2):NEXT
H    Ha
a    al
l    ll
l    li
i    ik
```

```
k    ka
a    ai
i    in
n    ne
e    en
n    n
      &
&    &
      F
F    Fr
r    ri
i    ie
e    en
n    nd
d    ds
s    s

OK
```

Concatenation

Strings may also be concatenated (put or joined together) through the use of the "+" operator. Try the following:

```
B$="WONDERFUL"+" "+A$

OK
PRINT B$
WONDERFUL Hallikainen & Friends

OK
```

Concatenation is especially useful if you wish to take a string apart and then put it back together with slight modifications. For instance:

```
C$=LEFT$(B$,4)+"-"+MID$(B$,6,6)+"-"+RIGHT$(B$,4)

OK
PRINT C$
WOND-RFUL H-ends

OK
```

VAL & STR$

Sometimes it is desirable to convert a number to its string representation and vice-versa. "VAL" and "STR$" perform these functions. Try the following:

```
STRING$="567.8"

OK
PRINT VAL(STRING$)
 567.8

OK
STRING$=STR$(3.1415)

OK
PRINT STRING$,LEFT$(STRING$,5)
3.1415    3.14

OK
```

"STR$" can be used to peform formatted I/O on numbers. You can convert a number to a string and then use LEFT$, RIGHT$, MID$, and concatenation to reformat the number as desired. Also see "USING".

"STR$" can also be used to conveniently find out how many print columns a number will take. For example:

```
PRINT LEN(STR$(3.157))
6

OK
```

If you have an application where a user is typing in a question such as "WHAT IS THE VOLUME OF A CYLINDER OF RADIUS 5.36 FEET, HEIGHT 5.1 FEET?" you can use "VAL" to extract the numeric values 5.36 and 5.1 from the question.

The following program sorts a list of string data and prints out a sorted list. This program is very similar to one given earlier for sorting a numeric list.

```
100 DIM A$(15):REM ALLOCATE SPACE FOR STRING MATRIX
110 FOR I=1 TO 15:READ A$(I):NEXT:REM READ IN STRINGS
120  F=0:I=1:REM EXCHANGE FLAG = 0 & SUBSRIPT FLAG = 1
130 IF A$(I)<=A$(I+1) THEN 180:REM IN ORDER, NO CHANGE
140  T$=A$(I+1):REM SAVE A$(I+1)
150  A$(I+1)=A$(I): REM EXCHANGE
160 A$(I)=T$
170 F=1:REM FLAG THAT WE EXCHANGED ELEMENTS
180 I=I+1:IF I<15 GOTO 130
185 REM ONCE WE HAVE MADE A PASS THRU ALL ELEMENTS,
186 REM CHECK TO SEE IF WE EXCHANGED ANY.  IF NOT,
187 REM WE ARE DONE SORTING
190 IF F THEN 120:REM EQUIVALENT TO IF F<>0 THEN 120
200 FOR I=1 TO 15:PRINT A$(I):NEXT:REM PRINT LIST
210 REM STRING DATA FOLLOWS
220 DATA APPLE, DOG, CAT, BITS, BYTES, RANDOM
230 DATA MONDAY, "***ANSWER***", " FOO"
240 DATA COMPUTER,  FOO, ELP, MILWAUKEE, SEATTLE
250 DATA ALBUQUERQUE
```

METER

All of the previous material has dealt with Basic as might be used in a business or scientific application. To log the operation of your broadcast station, you'd have to type the readings in using INPUT statements. This is not very efficient.

METER is a function that can be called with one or two arguments. It is generally called with two arguments in the form

```
METER(S,C)
```

where S is the site number and C is the channel number that we wish to get a reading from. For example,

```
PRINT METER(Ø,Ø)
```

will print the meter reading at site Ø, channel Ø. Site and channel must be between Ø and 99 or a function call error will occur.

There are several single argument methods of calling METER. These are useful in that a single array (matrix) element can be used to specify the metering data to be acquired. One number can specify both site and channel. In addition, a few negative numbers can be used to specify other common logging data such as time, day, and date.

METER(-1) returns the time as a 6 digit integer in HHMMSS format (see the section on time). METER(-2) returns the day of week as a number between Ø and 6 where Ø is Sunday. METER(-3) returns the date as a 6 digit integer in YYMMDD format.

If a single positive integer is used as the argument to METER, the argument assumes the following meaning:

```
(512*SITE) + (256*RAISE) + (128*LOWER) + CHANNEL
```

For example, METER(513) is equivalent to METER(1,1).

RAISE or LOWER take the value of Ø normally. They take the value of 1 if a RAISE or LOWER during the sample is required (sometimes required to drive an antenna monitor).

SCAN

When Basic encounters the word METER, it sends a request to the specified site and channel for that meter reading. When that reading is available, it is sent back to the originating site and passed back to Basic. A program often contains a FOR/NEXT loop to gather these readings. Such a loop that fills an array with readings from site 1, channels Ø through 9 is shown below:

```
FOR N=Ø TO 9: M(N)=METER(1,N): NEXT
```

Each time around the loop, a separate request is sent to site 1 for the required reading. This results in about one sample per second being stored in the array M.

By using the word SCAN, a single request can be sent to site 1 for all the required readings. The readings must be gathered in channel order. The form of the SCAN command is:

```
SCAN Site, MinChan, MaxChan
```

Adding the SCAN command to the above program line results in the following:

```
SCAN 1,Ø,9: FOR N=Ø TO 9: M(N)=METER(1,N): NEXT
```

This will sample about three channels per second.

It is suggested that scans of remote sites scan 1Ø channels or less. Longer scans result in slower operation if a data transmission error occurs.

Meter$

METER$ (meter-string) returns the meter reading in string form with the label and units programmed during system calibration. This allows simple programs to put the station parameters with labels and units on a CRT. A sample program using METER$ is shown in the Sample Basic Programs section of this manual.

METER$ can be called with the same arguments as METER (described above). If, for example, we call METER$(-3), we'll get a string with the date in English (September 11, 1985).

RAISE

RAISE can be used as a statement to generate a RAISE control pulse at a specified site and channel. The arguments passed to RAISE are of the same form as those used in METER. This allows the two argument form, or the single argument form.

```
RAISE(1,Ø)
```

This statement might be used to turn on the transmitter filaments.

RAISE can also be used as a function. In this case, it will return the meter reading that occurs if the selected site and channel are read while a RAISE pulse is applied. A sample program statement might be:

```
PRINT RAISE(1,Ø)
```

The RAISE function can also be used in string form, returning the METER$ that results when a RAISE is sent to the designated site and channel. A sample program statement might appear:

```
PRINT RAISE$(1,Ø)
```

LOWER

LOWER can be used as a statement or function in the same manner as RAISE.

STATUS

If any site in a system has a status transceiver, the status of each of the input lines can be read with the status function. Status returns a -1 if the status line is true (low if programmed for active low, high if programmed for active high) and a Ø if false. The status function takes the following form:

STATUS(Site,Channel)

A program that displays the first 2Ø lines of status from site 1 as strings is shown below. Each status line has two strings assocated with it, the true string and the false string. These are stored in the 19x1 array ST$. A subroutine dimensions this array and initializes it using READ statements. The false strings are stored in ST$(x,Ø), while the true strings are stored in ST$(x,1).

```
10 GOSUB 1000: REM Go initialze status$ array
20 GOSUB 2000: REM Display status
30 END

1000 DIM ST$(19,1): REM Dimension the string array
1010 FOR N=0 TO 19
1020 READ ST$(N,1), ST$(N,0): REM Read true$ and false$ for each line
1030 NEXT
1040 RETURN
1050 DATA "Main filaments on", "Main filaments off": REM True/false line 0
1060 DATA "Main plates ready", "Main plates not ready"
1070 DATA "Main plates on", "Main plates off"
1080 DATA "Main TX high power", "Main TX low power"
1090 DATA "Main TX driving antenna","Aux TX driving antenna"
1100 DATA "Day pattern", "Night pattern"
1110 DATA "Aux filaments on", "Aux filaments off"
1120 DATA "Aux plates ready", "Aux plates not ready"
1130 DATA "Aux plates on", "Aux plates off"
1140 DATA "Aux TX high power", "Aux TX low power"
1150 DATA "Generator on", "Generator off"
1160 DATA "Fire alarm tripped", "Fire alarm not tripped"
1170 DATA "Burgular alarm tripped", "Burgular alarm not tripped"
1180 DATA "Stereo","Mono"
1190 DATA "STL RX 1", "STL RX 2"
1200 DATA "Audio processor 1", "Audio processor 2"
1210 DATA "",""
1220 DATA "","": REM Additional strings can go here. . .
1230 DATA "",""
1240 DATA "",""
```

```
2000 DISPLAY CHR$(26);: REM Clear screen
2010 FOR N=0 TO 19
2020 DISPLAY N, ST$(N,-STATUS(1,N)); CHR$(27);"T"
2030 NEXT
2040 RETURN
```

Note that line 2020 displays the loop counter (N) followed by the ST$ corresponding to the value of the loop counter and opposite the value of the status function. This "opposite the value of the status function" returns a zero is the status is false, and a one (opposite -1) if the status is true.

Also in line 2020, CHR$(27);"T" tells the terminal (assuming a Qume QVT101) to clear from the cursor to the end of the line. This allows a new string to overwrite an old one without leaving a portion of the old one.

Status$

All 96 channels of status can be acquired at once using the word STATUS$. This function returns a 13 character (byte) string, each byte containing the state of 8 status lines. STATUS$ is passed a single argument, the site number from which status is to be obtained. It is often convenient to keep the state of a site (all the status) in a string, using much less space than keeping it in numeric variables (each numeric takes a minimum of four bytes while an entire status string is 13 bytes plus the string descriptor). In addition, string comparisons make it very easy and quick to determine whether any status at a site has changed.

To determine the state of any individual status line in STATUS$, use the ASC function to isolate an individual byte, and the AND operator to isolate an individual bit of the byte. Some sample code (written in the precompiler source code form) follows.

```
OldStat$(1)=OldStat$(1)  ;@ Store old status
stat$(1)=status$(1)      ;@ Get new status of site 1 in stat$(1)
site=1
chan=31
gosub @GetStat           ;@ Get status of site 1, channel 31 from stat$ array
if stat                  ;@ if it's true
  display "Day Pattern"
else                     ;@ if it's not true
  display "Night Pattern"
endif
if OldStat$(1)<>Stat$(1) ;@ If any status has changed
  gosub @StatChange      ;@ Go handle the status change
endif
end

subroutine @GetStat

;@      Set site and chan before calling this routine.  Result is returned as
;@      0 or -1 in Stat.  Stat is determined by evaluating corresponding bit of
;@      stat$(site).
```

```
byte=int(chan/8)+1              ;@ Chans Ø..7 in byte 1 of stat$, chan 8..15
                                ;@ in byte 2, etc.
bit=chan-(8*(byte-1))           ;@ The +/- 1 stuff cuz asc uses 1 as first
                                ;@ byte instead of Ø
if (asc(stat$(site),byte) and (2 ^ bit))<>Ø     ;@ If selected bit set
  stat=-1
else
  stat=Ø
endif
;@ Also, set up OldStat based on OldStat$
if (asc(OldStat$(site),byte) and (2 ^ bit))<>Ø          :@ If selected bit set
  OldStat=-1
else
  OldStat=Ø
endif
return
```

<u>TIME</u>

TIME can be used as a statement or a function. It almost appears to be a system variable, but the setting of time <u>must</u> be done through what looks like a LET statement. TIME cannot be set with an INPUT statement or a DATA statement. A typical routine to set time would be:

```
1Ø DISPLAY "Is it now "; TIME$
2Ø INPUT YN$
3Ø IF YN$="Y" OR YN$="y" THEN RETURN
4Ø DISPLAY "Type the current time in HHMMSS format.  "
5Ø INPUT A
6Ø TIME=A
7Ø GOTO 1Ø
```

Note that the time was set using a LET statement in line 6Ø. INPUT TIME will not work!

TIME is to be a 6 digit integer representation of the time in HHMMSS format. 6:Ø1:Ø2 pm would be 18Ø1Ø2.

TIME can be printed or used in calculations. As such, it appears as a system variable, but is actually implimented as a function call with no arguments required.

TIME is most often used to determine if something needs to be done based on a time schedule. If you need to switch pattern at sunset and SS(9) (sunset array entry number 9, for the ninth month) holds the sunset time for this month, the following statement might be used:

```
IF (TIME=>SS(9)) AND (METER(1,9)>1ØØØ) THEN LOWER(1,9)
```

This program line checks to see if it's past sunset and if we are still in day pattern (indicated by METER(1,9) being greater than 1ØØØ). If so, a LOWER statement is sent to the pattern select channel. If this line is included in a program loop, the pattern will only be changed once, since the line checks to see if it's already been done. Similar statements could be used to check the pattern between midnight and sunrise, and between sunrise and sunset.

When the time is printed or displayed, TIME$ is normally used. The below program line demonstrates the difference between TIME and TIME$.

```
DISPLAY TIME, TIME$
 140524    2:05:24 PM
```

Note that TIME$ cannot be set in a LET statement. TIME$ is determined based on TIME, which can be set using a LET statement.

DAY, DAY$, DATE, DATE$

These statements and functions act the same way as TIME and TIME$.

DAY is an integer between 0 and 6, inclusive, that indicates the current day of the week. It can be set in a LET statement or evaluated in a numeric formula. It is often used when program decisions are based on the day of the week. A typical application of DAY would be in a routine that tries to determine if we should currently be on advanced (daylight savings) or non-advanced (standard) time.

DAY$ returns the day of the week in English. The below program line demonstrates this:

```
DISPLAY DAY, DAY$
 2         Tuesday
```

DATE is a 6 digit integer representing the date in YYMMDD format. It can be set using a LET statement and read using DATE or DATE$. A sample program line is shown belos:

```
DISPLAY DATE, DATE$
 850911    September 11, 1985
```

The DATE can be taken apart to get the day of the month, the month, and the year using the below statements.

```
10 YR=INT(DATE/1E4)
20 MO=INT((DATE-(YR*1E4))/100)
30 DT=DATE-(YR*1E4)-(MO*100)
40 DISPLAY "Year = "; YR
50 DISPLAY "Month= "; MO
60 DISPLAY "Day   = "; DT
```

This is handy to index into an array holding the sunrise and sunset times for the year.

Program Storage

Programs reside in system RAM once they are typed in or are executing.

These programs will be lost during a power failure, or during an extended power failure if battery backup is included. Short programs (about 1 Kbyte) can be stored in space left in the EEPROM (Electrically Erasable Programmable Read Only Memory). Once a program has been typed into the DRC190, executing the following statement will save the program in EEPROM, if there is sufficient space available.

```
SAVE EEPROM
```

The program in EEPROM can be loaded using the following statement:

```
LOAD EEPROM
```

In addition, a program in EEPROM will be loaded and run on a system reset. This program might be a simple logging program, or could be a program to load another program from disc. A sample of this is shown below:

```
10 DISPLAY "Loading Logging Program"
20 LOAD "LOGGER PROGRAM",8
```

In addition, if there is additional EEPROM in the system (typically loaded into high memory on the RAM board), larger programs can be saved in EEPROM by specifying the address. A program residing in RAM can be saved in EEPROM on the RAM board using the statement:

```
SAVE EEPROM 24576
```

The program can be loaded in a similar manner:

```
LOAD EEPROM 24576
```

If a large program is saved in EEPROM on the RAM board, the boot program in the processor board EEPROM can load the other program. The boot program might be:

```
10 LOAD EEPROM 24576
```

This program would be loaded and executed on power up. The execution of this program loads and executes the other program from EEPROM on the RAM board.

Finally, a program can be stored and loaded to/from a second bank of RAM, typically battery backed. The DRC190 normally has the bottom 8 K of RAM on the processor board in common to all banks. For this reason, programs are typically stored at address 8192 and above. The form of a save to a different bank is:

```
SAVE EEPROM Bank, Address
```

You'd typically use the line:

```
SAVE EEPROM 2,8192
```

To load from another bank, use a similar statement:

```
LOAD EEPROM 2,8192
```

If this statement is preceeded with a line number and stored in the "boot" EEPROM, the DRC will, on reset or power up, load and run the program in the boot EEPROM. This will the cause the program at 2,8192 to be loaded and run. Since the RAM bank is considerably larger than the boot EEPROM, this allows an auto-boot of a large program from non-volatile RAM.

To allow for the simple chaining of programs, variables are saved during a LOAD. This allows a run-time program to be broken into two parts, an initialization program and an operating program. Special precautions are required, however, when using this feature.

The saving of variables during a load does not affect the deletion of variables when RUN is typed. Generally when chaining programs, the load of the operating program is done from within the initialization program, allowing the operating program to be loaded and "go to'd" without operator intervention and without deleting variables.

To save space, string variables that are defined in LET statements utilizing literals or DATA statements are not copied into "string space". Instead, H&F Basic "points" to the string in the program area of memory and recovers the string from there whenever required. Since these pointers will not be valid when a different program is loaded, it is necessary to force the copying of strings into string space during the initialization program. This can be done by adding a null string to each string as it is defined. A sample code segment follows:

```
FOR N=Ø TO 3
READ LB$(N)
LB$(N)=LB$(N)+""     :REM Concatenation forces copy to string space
NEXT
```

Summarizing, if LOAD/SAVE EEPROM is not followed by a number, the boot EEPROM is assumed. If it is followed by a single number, bank Ø (the main bank) of memory is assumed. If it is followed by two numbers separated by a comma, the first number is the bank, the second is the address.

Disc Program Storage

Programs may be stored on a Commodore 1541-II disc drive plugged in to the optional disc interface on the DRC19Ø. Program saving and loading procedures are similar to that used on Commodore computers, since the Commodore operating system (residing in the disc drive) is used.

In each of the following statements, the ",8" can be deleted if drive 8 is being used. That is, drive 8 is the default drive. If another drive is used, the statement should be followed by a comma and the drive number (which may be a numeric expression, if desired).

The following statement loads a program from drive 8 (usually the first drive in a system):

```
LOAD"PROGRAM NAME",8
```

The following statement saves a program residing in the DRC19Ø to disc drive number 8.

```
SAVE"PROGRAM NAME",8
```

Note that the above statement will not replace an existing file with the same name. To replace a file with the same name, use the below statement:

```
SAVE"@Ø:PROGRAM NAME",8
```

Note, however, that the "save and replace" function of Commodore disk drives is not very reliable. A better approach is to first delete the old program, then save the new program. This can be done using the following statements.

```
SBCMD"SØ:Program Name"
SAVE "Program Name"
```

The first line sends a "serial bus command" (SBCMD) of "scratch" (S) to delete Program Name. Note that the "Ø" following the "S" specifies drive Ø of the default drive (we left off the trailing ",8"). Commodore made dual drives, and the operating system still requires you to specify which drive of a dual drive is to be used, even though this is a single drive.

Serial Bus Commands

The word "SBCMD" allows the DRC19Ø to send several commands to a disk drive. The manual on the disk drive outlines the commands that are available. With the DRC19Ø, only the format and erase commands are typically used.

Formatting a Disk

The Commodore NEW command is used to format a disk. As part of the formatting, you can specify a name for the disk and a drive identifier. The following line of code will format the disk in drive 8 and name the disk.

```
SBCMD "NØ:DRC19Ø 11/15/89,HF",8
```

The following line of code does the same thing, taking advantage of the default drive specification.

```
SBMCD "NØ:DRC19Ø 11/15/89,HF"
```

The "NØ:" tells the drive to format the disk. The "DRC19Ø 11/15/89" names the disk. You can name it anything you want. You may want to include the date the disk was formatted in the disk name. Finally, "HF" is the disk identifier. The Commodore disk operating system (in the drive) uses this identifier to determine if a disk has been replaced.

Erasing a Program From Disk

The Commodore SCRATCH command is used to erase a program from a disk. The command allows you to erase a specific file (program) or a group of files, using wildcard characters. The following line of code will erase a program named "ProgramName".

```
SBCMD "SØ:ProgramName",8
```

As before, the ",8" is optional. Using wildcard characters, you can erase all files that start with "Pr" as below.

```
SBCMD "SØ:"Pr*"
```

Directory

Finally, you may get a directory of what is on a disc by typing the following statement:

```
DIR N
```

where N is the drive number. If N is omitted, drive 8 is assumed.

See the paragraph in the previous section regarding saving of variables during a program load.

There are various other functions and statements available in H&F Basic. These are covered in the following section.

KDDB6.OBJ

Programming Reference Material

Descriptions and examples of key word usage are listed below. These are listed in alphabetical order.

Words are broken into statements and functions.

A statement instructs the processor to take some action. Examples are NEW, LIST, RAISE, LOWER, CLEAR, etc. Most statements can be used within a program (a sequence of statements with line numbers), or as a command. A command is usually given after H&F BASIC has typed OK. This is called the "Command Level". Commands may be used as program statements. Certain commands, such as LIST and NEW will terminate program execution when they finish.

Functions act upon zero or more arguments and return a result to be used in an expression. For example, PRINT N is a statement that prints the numeric expression (a somewhat simple one) N. PRINT SQR(N) is a statement that prints the numeric expression SQR(N). SQR is a function that finds the square root of its argument (N) and returns it to the expression that called it.

NOTE: In the following descriptions, an argument of V or W denotes a numeric variable, X denotes a numeric expression, X$ denotes a string expression and an I or J denotes an expression that is truncated to an integer before the statement is executed. Truncation means that any fractional part of the number is lost, e.g. 3.9 becomes 3, 4.Ø1 becomes 4.

An expression is a series of variables, operators, function calls and constants which after the operations and function calls are performed using the precedence rules, evaluates to a numeric or string value.

A constant is either a number (3.14) or a string literal ("FOO").

A string may be from Ø to 255 characters in length. All string variables end in a dollar sign ($); for example, A$, B9$, K$, HELLO$.

String matrices may be dimensioned exactly like numeric matrices. For instance, DIM A$(1Ø,1Ø) creates a string of 121 elements, eleven rows by eleven columns (rows Ø to 1Ø and columns Ø to 1Ø). Each string matrix element is a complete string, which can be up to 255 characters in length.

Statements and functions available on the DRC19Ø are listed below.

NAME	EXAMPLE	PURPOSE/USE
ABS(X)	12Ø PRINT ABS(X)	Gives the absolute value of the expression X. ABS returns X if X>=Ø, -X otherwise.
AND	1ØØ IF (A<B) AND (B<C) THEN 5ØØ	AND can be used to indicate a logical or bitwise AND function. In the 1ØØ example, it is indicating a logical AND. The two relational expressions return a -1 if they are true, and a Ø if they are false. The -1 is represented as a 16 bit integer (1111111111111111), which when ANDed with another -1, yields another -1, which indicates TRUE. TRUE AND TRUE is TRUE.
	2ØØ PRINT 16 AND 31	In line 2ØØ, the two integers are changed to 16 bit binary form, ANDed, and the result returned. In this case, 16 AND 31 is ØØØØ ØØØØ ØØØ1 ØØØØ

		AND 0000 0000 0001 1111, giving a result of 0000 0000 0001 0000 or 16.
ASC(X$)	300 PRINT ASC(X$)	Returns the ASCII numeric value of the first character of the string expression X$. An FC error will occur if X$ is the null string.
ASC(X$,N)		Returns the ASCII numeric value of the Nth character of string X$.
ATN(X)	210 PRINT ATN(X)	Gives the arctangent of the argument X. The result is returned in radians and ranges from -PI/2 to PI/2.
BREAKOK	113 BREAKOK	This statement allows the DRC Basic to accept a control-C from the console (device 0). This is the default condition (control-C is allowed unless a NOBREAK has been encountered. This is most often used where a modem is driving the console RS232 port (allowing unknown users access to the system, and we don't want them to halt the program), or where a terminal sends back binary codes that might include a control-C (such as a request for cursor address). In this case, a NOBREAK would be executed immediately prior to doing the cursor address request, and a BREAKOK would be executed immediately after the terminal responds with the cursor address.
CHR$(I)	275 PRINT CHR$(I);	Returns a one character string whose single character is the ASCII equivalent of the value of the argument (I) which must be >=0 and <=255.
CLEAR	CLEAR 500	Clears all variables, resets FOR-NEXT, GOSUB, and DATA states. Sets aside 500 characters of string storage.
	CLEAR 500,1E3	Clears all variables, resets FOR-NEXT, GOSUB, and DATA states. Sets aside 500 characters of string storage. Tells BASIC to use only 1000 (1E3) bytes of RAM, if available. This command allows the user to set aside high RAM for use by other than the BASIC program. This command may be executed from within a program, typically in an initialization routine. Remember, however, that CLEAR clears the GOSUB stack, so a RETURN cannot be used at the end of a routine that includes a CLEAR.
CLRSTK	20 CLRSTK	Clears the subroutine and FOR-NEXT stack. Used to abort a subroutine. Typically followed by a GOTO statement returning program control to the main level of the program.

CONT	CONT	Continues execution of program after it was interrupted by CTRL-C. Variables can be examined and changed in command mode, but no program changes can be made or command level errors encountered, or continue cannot be done (CN error).
COS(X)	2ØØ PRINT COS(X)	Gives the cosine of the expression X. X is interpreted as being in radians.
DATA	1Ø DATA 1,3,-1E3,.Ø4	Specifies data, read from left to right. Information appears in data statements in the same order it will be read in the program.
	2Ø DATA " FOO",ZOO	Strings may be read from DATA statements. If you want the string to contain leading spaces (blanks), colons (:) or commas (,), you must enclose the string in double quotes ("). It is impossible to have double quotes within string data or a string literal. (""HELP"") is not legal. See READ.
DATE	5ØØ PRINT DATE	DATE is an integer variable holding the date in YYMMDD format. January 23, 1982 is stored as 82Ø123. By comparing DATE with a previously stored value, date dependent functions can be performed. For example, INT(DATE/1ØØ)-1ØØ*INT(DATE/1ØØØØ) represents the month. This can be used to select pattern change times. DATE is updated by the internal clock.
	6ØØ DATE=82Ø123	Set DATE using a LET statement. DATE cannot be set using an INPUT or READ statement. If this is desired, INPUT or READ to a temporary variable and then set DATE using a LET statement. Setting DATE also sets DATE$.
DATE$	8ØØ PRINT DATE$	DATE$ is a string variable that can be read, but not written to (DATE$=Tuesday is NOT legal). As such (and it is in fact coded this way in the firmware), it may be thought of as a zero argument function. The example prints date (set up above) in English. Date$ is set up using a LET DATE= statement. DATE$ cannot be set other than through DATE.
DAY	1ØØ IF DAY=Ø THEN 3ØØ 2ØØ DAY=3	DAY is an integer variable holding the day of week. Ø represents Sunday, 6 represents Saturday. DAY is normally used in IF statements. DAY can be initialized with a LET statement. It cannot be initialized with an INPUT or DATA statement. If this is desired, use INPUT or DATA to set a temporary variable, then use a LET statement to set DAY to the temporary

		variable. DAY is updated by the internal clock.
DAY$	4ØØ PRINT DAY$	Prints day (set up above) in English. Since DAY$ follows DAY, it cannot be set using a string assignment statement. Set the numeric DAY instead.
DCD	123 IF DCD(Ø)	Data Carrier Detect samples the logic state on pin 2Ø of J22. If the voltage on J22-2Ø is greater than 3 volts, DCD(Ø)=-1. If the voltage is less than -3 volts, DCD(Ø)=Ø. If J22 of the DRC19Ø is connected to an external modem using a null modem cable (cross wire 2 and 3, 4 and 5, 8 and 2Ø), then DCD will be true if the modem has dectected carrier. This is a reliable method of determining whether a modem connection has been made. Note that although the software for DCD and DTR support the sampling of multiple ports on the DRC, currently only port Ø of the hardware supports these functions.
DEBUG	1Ø2 DEBUG=1	DEBUG is normally zero. Setting it to one causes data received by the Bell 2Ø2 modem on the processor board to be displayed in hexadecimal. It is suggested that the terminal be run at 9.6 Kbits/second so that it can keep up with the incoming data. If a received message has a valid checksum, the message is followed by an exclamation point. If DEBUG is set to two, the data being transmitted by the 2Ø2 modem is displayed. Since this data is prior to the modem, the checksum is always assumed correct and no exclamation point follows the messages. See the firmware theory of operation section for further information on the format of messages.
DEF	1ØØ DEF FNA(V)=V/B+C	The user can define functions like the built in functions (SQR, SGN, ABS, etc.) through the use of the DEF statement. The name of the function is "FN" followed by any legal variable name, for example: FNX, FNJ7, FNKØ, FNR2. User defined functions are restricted to one line. A function may be defined to be any expression, but may have only one argument. In the example, B and C are variables that are used in the program. Executing the DEF statement defines the function. User defined functions can be redefined by executing another DEF statement for the same function. User defined string functions are not allowed. "V" is called the dummy variable.
	11Ø Z=FNA(3)	Execution of this statement following the above would cause Z to be set to 3/B+C, but the value

of V would be unchanged.

DEV

```
DEVØ$=CHR$(2Ø)+CHR$(27)+"A"
DEV1$=CHR$(18)
```

DEV1$ is the string of characters that Basic sends to the CRT terminal to enable the printer port after executing a print statement. The CRT receives this string and sends the following data to the printer instead of the screen. DEVØ$ disables the printer port and sends the following data to the screen. The samples shown apply to the Qume QVT1Ø1+ terminal. DEVØ$ and DEV1$ can each be up to ten characters long. They can only be on the left side of an equal sign in an assignment statement. They cannot be loaded using an INPUT or READ statement. They cannot be used as functions (on the right side of an equal sign or in a print or display statement). They are not cleared by CLEAR, RUN or LOAD.

DIM

`113 DIM A(3),B(1Ø)` Allocates space for matrices. All matrix elements are set to zero by the DIM statement.

`114 DIM R3(5,5),D$(2,2,2)` Matrices can have more than one dimension. Up to 255 dimensions are allowed, but due to the restriction of 113 characters per program line (as opposed to 132 characters allowed per print line) the practical maximum is about 34 dimensions.

`115 DIM Q1(N),Z(2*I)` Matrices can be dimensioned dynamically during program execution. If a matrix is not explicitly dimensioned with a DIM statement, it is assumed to be a single dimensioned matrix whose single subscript may range from Ø to 1Ø (11 elements).

`117 A(8)=4` If this statement was encountered before a DIM statement for A was found in the program, it would be as if a DIM A(1Ø) had been executed previous to the execution of line 117. All subscripts start at zero (Ø), which means that DIM X(1ØØ) really allocates 1Ø1 matrix elements.

DIM

`25 DIM A$(1Ø,1Ø)` Allocates space for a pointer and length for each element of a string matrix. No string space is allocated.

DIR

`DIR 8` Displays a directory of the specified disc drive (8 in the example) on the console. If the drive number is not specified, drive 8 is assumed.

DISPLAY

`43Ø DISPLAY A, A$` DISPLAY operates the in the same manner as PRINT, except that DISPLAY drives the CRT terminal while PRINT drives the printer. On the DRC DISPLAY and PRINT both drive the same RS232 port. When a change between the CRT or printer is encountered, the CRTSTR or PRTSTR string of characters is sent to disable or enable the printer port on the CRT ternimal. Note that the same character counter is used by both DISPLAY and PRINT. It is suggested that DISPLAY and PRINT statements ending with semicolons (;) or commas (,) not be intermixed, since the shared character counter will cause strange print formatting. On printers, sending a CHR$(13) will resynchronize the character counter and the print head, allowing tabs to work properly. On a CRT, it is suggested that direct cursor addressing be used (requiring the writing of a subroutine typically called GOTOXY).

DISPLAY USING — Formats text according to a string expression. See PRINT USING.

DTR — IF DTR(Ø) THEN... — Data Terminal Ready performs exactly the same function as DCD (Data Carrier Detect). If the DRC19Ø is driving a terminal instead of a modem, DTR may be used to determine if the terminal is ready. See DCD for further information.

ECHO — ECHO=Ø — ECHO determines whether Basic will echo the characters typed into the RS232 port. Setting ECHO=1 ECHO to zero causes the echo to be suppressed. In addition, the line feed normally appended to a carriage return is deleted. The OK message when the DRC is ready to accept a command is also deleted. When ECHO = Ø, an immediate statement sent to the DRC is executed without the statement being echoed, or there being any output, unless output is requested. An immediate statement that makes a print or display request will cause the requested data to be output, followed by only a carriage return. This simplifies the interface between the DRC and an external computer. The external computer can use a PRINT statement to send a "prompt" to the DRC. The prompt is actually the immediate request for data. The external computer would then use an INPUT, GET or INKEY$ to gather the response from the DRC, which is terminated by a carriage return. Since this is quite similar to the way an operator responds to a request for data, the interface can be easily handled from a variety of languages and computers. On power-up, ECHO is set to 1, enabling the echo. ECHO can be set or cleared at any point in the program. It is a "write only" variable that must be set in an assignment statement, not in an INPUT or READ

		statement. ECHO is not affected by CLEAR, RUN or LOAD.
EEPROM	SAVE EEPROM SAVE EEPROM 24576 SAVE EEPROM 2,8192 LOAD EEPROM LOAD EEPROM 24576 LOAD EEPROM 2,8192	EEPROM redirects a SAVE or LOAD statement from disk to system non-volatile memory. If the word EEPROM is not followed by a number, the LOAD or SAVE refers to approximately 1 K of free EEPROM on the processor board. This area is called the "boot" EEPROM because any program saved there is automatically executed on system reset. The boot program could be a small display/logging program, or it could load a larger program from other non-volatile memory or disk. If the word EEPROM is followed by one number, it is assumed that the save or load applies to non-volatile memory in bank Ø of system memory. Some systems may have a protion of bank Ø memory loaded with EEPROM chips, allowing non-volatile storage in this area. Logging/control programs are typically stored in this area. Finally, if the word EEPROM is followed by two numbers separated by a comma, the first number is assumed to be the bank number of memory to be referenced, and the second number is assumed to be the address. Systems with multi-bank RAM typically have 8K of RAM on the processor board in common to all banks. Bank Ø is used for Basic program and variable storage. Bank 2 may be used for non-volatile program storage on RAM boards that have a battery backed second bank (Systek 644Ø). Non-volatile storage usually starts at bank 2, address 8192. This is the first location of non-volatile storage (below 8192 is the common RAM on the processor board). For further information, see SAVE and LOAD.
END	999 END	Terminates program execution without printing a BREAK message. (see STOP). CONT after an END statement causes execution to resume at the statement after the END statement. END can be used anywhere in the program, and is optional.
ERR	55Ø PRINT ERR	ERR returns a numeric error code. If ERR=1, then metering data was not present during the last METER, RAISE, LOWER, or STATUS function. Reading ERR clears it (sets it to Ø). ERR cannot be used on the left side of a LET statement. As such, it may be thought of as a zero argument function.
EXP(X)	15Ø PRINT EXP(X)	Gives the constant "E" (2.71828) raised to the power of X. (E^X). The maximum argument that can be passed to EXP without overflow occuring is 87.3365.

FN	2ØØ DEF FNA(V)=V/B+C	FN means "function". Use of FN allows the programmer to define functions based on a single numeric "dummy variable" (in the example, it's V). For further information, see DEF.
FOR	3ØØ FOR V=1 TO 9.3 STEP .6	(see NEXT statement) V is set equal to the expression following the equal sign, in this case 1. This value is called the initial value Then the statements between FOR and NEXT are executed. The final value is the value of the expression following the TO. The step is the value of the expression following STEP. When the NEXT statement is encountered the step is added to the variable.
	31Ø FOR V=1 TO 9.3	If no STEP was specified, it is assumed to be one. If the step is positive and the new value of the variable is <= the final value (9.3), or the step value is negative and the new value of the variable is => the final value, then the first statement following the FOR statement is executed. Otherwise, the statement following the NEXT statement is executed. All FOR loops execute the statements between the FOR and the NEXT at least once, even in cases like FOR V=1 TO Ø.
	315 FOR V=1Ø*N TO 3.4/Q STEP SQR(R)	Note that expressions (formulas) may be used for the initial, final and step values in a FOR-NEXT loop. The values of the expressions are computed only once, before the body of the FOR-NEXT loop is executed.
	32Ø FOR V=9 TO 1 STEP -1	When the statement after the NEXT is executed, the loop variable is never equal to the final value, but is equal to whatever value caused the FOR-NEXT loop to terminate. The statements between the FOR and its corresponding NEXT in both examples above (31Ø & 32Ø) would be executed 9 times.
	33Ø FOR W=1 TO 1Ø: FOR W=1 TO 7: NEXT W: NEXT W	ERROR!!! Do not use nested FOR-NEXT loops with the same index variable. FOR-NEXT loop nesting is limited only by the available memory.
FRE(X)	27Ø PRINT FRE(Ø)	Gives the number of memory bytes currently unused by H&F BASIC if the argument is a numeric, such as example line 27Ø.
	275 PRINT FRE(A$)	Gives the number of bytes of memory allocated to string storage and currently unused. Can be changed using the CLEAR command. This function forces a "string garbage collection". Depending upon the amount of string space used, the function may take several seconds to execute.

		During string garbage collection, system interrupts are disabled.
GOTO	5Ø GOTO 1ØØ	Branches to the statement specified.
GOSUB	1Ø GOSUB 91Ø	Branches to the specified statement (91Ø) until a RETURN is encountered; when a branch is then made to the statement after the GOSUB. GOSUB nesting is limited only by the available memory.
IF...GOTO	32 IF X<=Y+23.4 GOTO 92	Equivalent to IF-THEN, except that IF-GOTO must be followed by a line number, while IF-THEN can be followed by either a line number or another statement.
IF...THEN	IF X<1Ø THEN 5	Branches to specified statment if the relation is true.
	2Ø IF X<Ø THEN PRINT "X LESS THAN Ø"	Executes all of the statements on the remainder of the line after the THEN if the relation is true.
	25 IF X=5 THEN 5Ø:Z=A	WARNING!!! The "Z=A" will never be executed because if the relation is true, H&F BASIC will branch to line 5Ø. If the relation is false, H&F BASIC will proceed to the line after line 25.
	26 IF X<Ø THEN PRINT "ERROR, X IS NEGATIVE":GOTO 35Ø	In this example, if X is less than Ø, the PRINT statement will be executed and then the GOTO statement will branch to line 35Ø. If the X was Ø or positive, H&F BASIC will proceed to execute the lines after line 26.
INKEY$	1Ø A$=INKEY$(N)	Inkey$ gets a single byte string from the specified I/O device (Ø - Console, 1 - Printer, 2 - Direct Connect Modem, 3 - RS232 port 3). If there has been no keystroke at the specified device, inkey$ returns with the null string ("").
INPUT	3 INPUT V,W,W2	Requests data from the terminal. Each value typed in must be separated by a comma (,). The last value should be followed by a carriage return. If more data was requested in an INPUT statement than was typed in, a "??" is printed and the rest of the data should be typed in. If more data was typed in than was requested, the extra data will be ignored. Strings must be input in the same form as they are specified in DATA statements (leading spaces require quotes surrounding string).
	5 INPUT "VALUE";V	Optionally types a prompt string ("VALUE") before requesting data from the terminal. The prompt string is typed on the console (device Ø). Typing a carriage return in response to an INPUT statement leaves the variable unchanged.

	4Ø INPUT X$	Reads a string from the CRT. String does not have to be quoted, but if not, leading blanks will be ignored and the string will be terminated on a "," or ":" character.
INPUT#	15Ø INPUT#2,A$ 16Ø N=2 17Ø INPUT#N,"HI";A$	INPUT# allows input from a selected input output device. The device is specified as a numeric expression following the # sign and is followed by a comma or semicolon. A prompt string can be included after the "#N,", and it will be sent to the selected device. Device numbers are: Ø - Console CRT, 1 - Printer plugged into CRT peripheral port, 2 - Direct Connect Modem, 3 - RS232 port 3, which appears on J23 on systems equipped with the direct connect modem.
INPUT()	12Ø INPUT(1Ø)#2,A$	An optional "time out" may be specified for the input statement. This time out (in parenthesis) is specified in seconds. It uses the real time clock, which has a resolution of 1 second. The sample line will result in a time out of between 9 and 1Ø seconds. If the requested data is input within the specified time period, the statement acts the same as a normal INPUT statement. If the terminating carriage return is not received in the specified time, one is automatically generated. If this happens before any data was input, the variables that were to be loaded (whether numeric or string) remain unchanged. If it is important to know whether this statement was terminated because of operator input or because of a time out, the variables can be preloaded with some flag information (perhaps loading a string with a null string). This statement is especially useful when more than a single character is requested of an operator, but you do not want the program to wait for ever for the operator to respond.
INT(X)	14Ø PRINT INT(X)	Returns the largest integer less than or equal to its argument. For example: INT(.23)=Ø, INT(7)=7, INT(-.1)=-1, INT(-2)=-2, INT(1.1)=1. The following would round X to D decimal places: INT(X*1Ø^D+.5)/1Ø^D
LEFT$(X$,I)	31Ø PRINT LEFT$(X$,I)	Gives the leftmost I characters of the string expression X$. If I<=Ø or >255 an FC error occurs.
LEN (X$)	22Ø PRINT LEN (X$)	Gives the length of the string expression X$ in characters (bytes). Non-printing characters and

		blanks are counted as part of the length.
LET	300 LET W=X	Assigns a value to a variable.
	310 V=5.1	LET is optional.
	27 LET A$="FOO"+V$	Assigns the value of a string expression to a string variable. LET is optional.
LINE	600 IF LINE(1)>45 THEN 10	LINE is an integer variable holding the current line number of the specified device (device 1 or the printer in this case). Separate line counters are maintained for each device. The line number is updated each time a line feed is sent to the device. LINE is incremented on each line feed until LINE(N) exceeds MAXLINE(N), whereupon LINE(N) is returned to zero.
	602 LINE(1)=0	LINE(N) may be set to any number in an assignment (LET) statement. Typically, it is set to 0 after the operator has set the printer paper to the top of a page.
LIST	LIST	Lists entire current program.
	LIST 100	Lists program starting at line 100.
	?:LIST	? is equivalent to the word PRINT, so it sets the I/O device number to 1. LIST does not change the I/O device number, so the listing is sent to the printer.
LISTEN	LISTEN 1	LISTEN forces the specified site into the intercom talk mode. In the example, site 1 is placed into the talk mode for thirty seconds. This allows hearing of sound at that site. This is typically used to hear noisey blowers, burgulars, etc. The word LISTEN must be followed by a numeric expression for the site that needs to be listened to. That site will send an intercom talk message, enabling the intercom speaker at all other sites. The addressed site will then send thirty seconds of ambient audio, followed by an intercom untalk message, muting all the system speakers. The LISTEN command can be dupliated by pressing the decimal point key on the DRC190 front panel. When the front panel decimal point is pressed, the site displayed on the front panel LCD is instructed to send intercom audio.
LOAD	LOAD FN$,8 LOAD FN$ 10 LOAD"HI",8 10 LOAD"HI"	Loads the specified file name (FN$) from the specified disc drive (8) into the DRC. Any previous program is deleted from the DRC. If LOAD is used within a running program, the running program is deleted, the new program is

		loaded and run. This allows for simple chaining of programs. If the comma and drive number is deleted, drive 8 is assumed. Note that LOAD does not delete variables. If variable deletion is required, the LOAD should be preceeded by a CLEAR. Since a LOAD does not delete variables, simple program chaining can be accomplished. For example, a run-time program may be broken down into an initialization program that initializes all variables and an operating program that actually does the work. When initializing string variables, however, they need to be copied into "string space". This can be accomplished by concatenating each string with a null string as it is defined. ie: READ LB$(N): LB$(N)=LB$(N)+"". Without this copying, H&F Basic sets up a pointer into the program to reference a string literal, saving RAM. Once a different program is loaded, these pointers would be invalid. Forcing the copying of strings into string space in the initialization program allows these strings to be referenced properly in the operating program. Utilizing an initialization program and a run-time program allows the DRC to perform more complex tasks. In addition, seldom accessed menu items in the operating program could load the code for the task from disk or non-volatile RAM, again saving RAM space.
LOAD EEPROM		Loads a program from non-volatile memory (EEPROM or Electrically Erasable Programmable Read Only Memory). On system reset, the EEPROM program is loaded and run. This may be used to auto-start a simple logging program, or may be used to auto-boot a more complex program from disc.
LOAD EEPROM N	LOAD EEPROM 24576	Loads a program from non-volatile memory (EEPROM) at the specified address (N) of bank 0. This allows loading larger programs than can be contained in the "boot" section of the processor board EEPROM. The RAM board in the system must be partially loaded with EEPROM. When loaded as H&F suggests, N=24576.
LOAD EEPROM B,A		Loads a program from non-volatile memory (EEPROM or battery backed RAM) at bank B, address A. This may be used for non-volatile storage of LOAD EEPROM 2,8192 programs in systems with multi-bank RAM. Note that addresses 0 though 8191 are in common to all banks. Therefore, address 8192 of bank 2 is suggested for non-volatile storage. See also EEPROM and SAVE.
LOG(X)	160 PRINT LOG(X)	Gives the natural (Base E) logarithm of its

argument X. To obtain the base Y log of X, use the formula LOG(X)/LOG(Y). For example, the base 1Ø (common) log of 7 = LOG(7)/LOG(1Ø).

LOWER 5ØØ LOWER (3,2) The LOWER statement selects the desired site and channel. It then sends a LOWER pulse to the appropriate remote control. In the example, the LOWER pulse was sent to site 3, channel 2.

25Ø PRINT LOWER (S,C) Selects site S, channel C, sends a series of LOWER pulses. Returns the resulting meter reading. The length of time that the LOWER output of the DRC19Ø is held while waiting for the reading to stabalize is determined by the "sample delay" programmed during calibration of the particular channel.

LOWER$ 255 PRINT LOWER$ (S,C) Does the same as LOWER except that a string is returned. The string consists of 3 label characters, an equal sign, the reading, and 2 label characters. ICP= 4.52AP

MAXLINE 6Ø3 IF MAXLINE(1)-LINE(1)<5 THEN GOSUB 1ØØØ

MAXLINE(N) is a system defined variable that represents the maximum line number of device N. This can be used with LINE for end of page detection. This is more valuable than an EOP (End Of Page) function available on some systems, since it allows you to determine how much space is left on a page instead of merely knowing that you just ran off the end. It allows log entries to include multiple lines without the entry going over a page break. This is similar to the .CP (conditional page break) command in Wordstar.

1Ø MAXLINE(1)=65 MAXLINE(N) can also be set with an assignment statement (LET). Since line numbers start at zero, MAXLINE should be the number of lines per page less one. MAXLINE cannot be set using an input or read.

MDMSPD 1Ø MDMSPD=3 MDMSPD allows setting of the direct connect modem board speed. This modem will operate at either 2Ø PRINT MDMSPD 3ØØ or 12ØØ bits per second. MDMSPD indicates the speed in hundreds of bits per second. MDMSPD can be read or written to (can be on either side of the equals sign in an assignment statement), but cannot be written to or set using an INPUT or READ statement.

MDMSTAT$ 1Ø IF MDMSTAT$="A" THEN 1ØØ

MDMSTAT$ contains the last received modem status character. These characters come from the direct connect modem. The direct connect modem sends a control-N followed by a status characater and a line feed and a carriage return. The DRC190 firmware captures all control-N - status character sequences, preventing the control-N or the status character from showing up in an INPUT#2 or INKEY$(2) result. The line feed, carriage return sequence following the control-N status character sequence is not trapped out, however. MDMSTAT$ holds the single character status message from the direct connect modem, as described here, and in the modem manual (reprinted in the rear of this manual). MDMSTAT$ characters are: R - Indicates the line the modem is connected to is ringing. A - Indicates the modem has answered an incoming call, or an out-going call has been answered by a remote modem. N - Indicates that there was no answer on an out-going call (and the modem has disconnected itself). D - Indicates the modem has disconnected. Note that the modem module sends a D only when it initiates the disconnect (due to a loss of carrier from the other modem). The DRC190 firmware sets MDMSTAT$ to "D" on receiving a disconnect status message from the modem module (the remote modem has disconnected), or on an END command (control-N E) being sent to the modem module by the DRC190 application program (ie, PRINT#2,CHR$(14);"E"). Finally, note that MDMSTAT$ changes only on receiving a status message from the modem, or on receiving an END command from the applications program. If MDMSTAT$ is N due to no answer on a previous call, and another call is placed, MDMSTAT$ will remain N until that call is answered. It is suggested that prior to placing a call, the modem be sent an END command, forcing MDMSTAT$ to D. Then, when the call is placed, the applications program can watch MDMSTAT$ for A or N indicating that the call was answered or not answered.

MESSAGE$ MESSAGE$(1)="Hi"
DISPLAY MESSAGE$

This statement passes a character string to another site in the DRC190 system. The string can be up to 100 bytes or characters long. If the message is received at a site with an error, the message is rejected. It is up to the user software to determine if any messages are missing, possibly through the use of message acknowledgments and message sequence numbers. The message receive buffer can hold only one message. This message can be retrieved using the MESSAGE$ function. Reading the message clears the buffer. The argument to a MESSAGE$ statement is the site number that the message is to be sent to. Intersite message passing allows a DRC19Ø system to do parallel processing. The message passing is similar that in the Occam language.

MID$(X$,I)	33Ø PRINT MID$(X$,I)	MID$ called with two arguments returns characters from the string expression X$ starting at character position I. If I>LEN(X$), then MID$ returns a null (zero length) string. If I<=Ø or >255, an FC error will occur.
MID$(X$,I,J)	34Ø PRINT MID$(X$,I,J)	MID$ called with three arguments returns a string expression composed of the characters of the string expression X$ starting at the Ith character for J characters. If I>LEN(X$), MID$ returns a null string. If J specifies more characters than are left in the string, all characters from the Ith are returned.
METER	26Ø PRINT METER (S,C)	Selects site S, channel C and returns the resulting meter reading. This can also be called with a single argument. That argument is (512*SiteNumber)+(1*Raise)+(2*Lower)+ChanNum where Raise is 1 if a RAISE pulse is required during the reading and Lower is 1 if a LOWER pulse is required during the reading.
METER$	265 PRINT METER$ (S,C)	Same as meter, but returns a string consisting of label, reading and units.
MODEMTST	MODEMTST	Puts the DRC19Ø in the modem test routine. This routine allows checking of transmit levels and frequencies, and checks modem demodulator tuning. See the processor board adjustments section for further information.
MONITOR		The word MONITOR exits Basic and enters the system monitor. This allows for the inspection and changing of processor registers and system memory locations. In addition, it allows for initialization of the system EEPROM. The monitor can also be entered by causing a Nonmaskable Interrupt (NMI) by grounding pin 6 of the processor chip. For further information on the monitor, see the monitor section of the adjustments section.
NEW	NEW	Deletes current program and all variables.
NEXT	34Ø NEXT V 345 NEXT	Marks the end of a FOR-NEXT loop. If no variable is given, it matches the most recent FOR.

	350 NEXT V,W	A single NEXT may be used to match multiple FOR statements. Equivalent to NEXT V:NEXT W.
NOBREAK	122 NOBREAK	Execution of the NOBREAK statement disallows the stopping of the program by pressing control-C on the console terminal (device 0). Program breaks would be allowed again if the statement BREAKOK is executed. NOBREAK is typically used to disallow program breaks from a direct connect modem that may be connected to the device 0 port. It may also be used to disallow program breaks that may be generated by cursor position reports from terminals (cursor position may be reported in binary).
NOT	100 IF NOT A=B THEN 200	A logical or bitwise NOT. NOT inverts the logic of the expression following it. In the example, if A=B was TRUE, the NOT A=B is FALSE. Note that TRUE is represented by a -1 in 16 bit integer form while FALSE is represented by a zero. NOT does a bitwise inversion, changing all the 1s to 0s and 0s to 1s. This corresponds to the described logical NOT.
ON. .GOSUB	110 ON I GOSUB 50,60	Identical to ON-GOTO except that a subroutine call (GOSUB) is executed instead of a GOTO. RETURN from the GOSUB branches to the statement after the ON-GOSUB.
ON...GOTO	100 ON I GOTO 10,20,30	Branches to the line indicated by the I'th number after the GOTO. That is: IF I=1 THEN GOTO 10 IF I=2 THEN GOTO 20 IF I=3 THEN GOTO 30 If I=0 or I attempts to select a nonexistent line (>=4 in this case), the statement after the ON statement is executed. However, if I is >255 or <0, an FC error will result. As many line numbers as will fit on a line can follow an ON...GOTO.
	105 ON SGN(X)+2 GOTO 40,50,60	This statement will branch to line 40 if the expression X is less than zero, to line 50 if it equals zero, and to line 60 if it is greater than zero.
OR	100 IF A=B OR A=C THEN 200	Logical or bitwise OR. If the expression on either side of the OR is true, then the entire expression is true. In a bitwise OR, if a particular bit position in either of the arguments is 1, then that bit position will be a

		1 in the result. For example, 1 OR 3 = 3.
PEEK(X)	356 PRINT PEEK(I)	The PEEK function returns the contents of memory address I. The value will be >=Ø and <=255. If I is >65535 or <Ø, an FC error will occur.
PEEK(B,A)	357 PRINT PEEK(2,8192)	If the PEEK function is given two arguments, the first argument is the bank of bank selected memory and the second is the address.
POKE	POKE I,J	The POKE statement stores the byte specified by its second argument (J) into the location of the memory map specified by its first argument (I). The byte to be stored must be =>Ø and <=255, or an FC error will occur. The address (I) must be =>Ø and <=65535 or an FC error will occur. Careless use of the POKE will probably cause you to "POKE" H&F BASIC to death; that is the machine will hang, and you will have to reset it, losing your program. Note that since the processor board has a write protected EEPROM, it is difficult (at best) to POKE into the EEPROM. All EEPROM accesses should be through the front panel set up and calibrate, and through Basic SAVE. The POKE statement could be used to load machine language routines or to give direct access to input output devices (such as the parallel port on the direct connect modem card).
	POKE I,J,K BANK ADDRESS DATA	~~If POKE is given three arguments, the first is the data to be POKEd, the second is the bank of bank selected memory where the data will be stored and the third is the address where the data will be stored.~~
POS(I)	26Ø PRINT POS(I)	Gives the current print head or cursor position on the last used I/O device. The leftmost position is position Ø. This function uses default tab spacing and does not know about cursor addressing of terminals, so it is not used very frequently.
PRINT	35Ø PRINT X,Y;Z 37Ø PRINT 38Ø PRINT X,Y; 39Ø PRINT"VALUE=";A 4ØØ PRINT A2,B,	Prints the value on the printer (plugged into peripheral port on the CRT terminal). If the list of values to be printed does not end with a comma or a semicolon, then a carriage return/line feed is executed after all the values have been printed. Strings enclosed in quotes (") may also be printed. If a semicolon separates two expressions in the list, their values are printed next to each other. If a comma appears after an expression in the list, then the print head is advanced to the next TAB

position, which may be on the next line. A comma moves the print head to the next "comma-field", each of which is 13 characters wide. If there is no expression to be printed, as in line 370 of the examples, then a carriage return/line feed is sent to the printer.

60 PRINT X$ — Prints the string expression on the printer.

70 PRINT "FOO"+A$

410 PRINT MID$(A$,2) — String expressions may be printed.

PRINT USING

10 PRINT USING "##.##" A

20 A$="###.##"

30 PRINT USING A$ 3.4

PRINT USING allows numbers that are not in exponential form to be formatted. They are formatted according to the string or string expression following the word USING. The number of #s prior to the decimal point corresponds to the number of digits to be printed prior to the decimal point. Leading zeroes will be suppressed by the printing of spaces instead of zeroes. Recall that an additional space is saved for the sign of the number. The number of #s following the decimal point corresponds to the number of digits that will be printed after the decimal point. Trailing zeroes are not replaced with spaces. Multiple USINGs may be used in one PRINT statement to change the print format without having to do another PRINT. PRINT USING makes it easy to align decimal points in a table.

RAISE

100 RAISE (S,C) — The RAISE statement selects site S, channel C, and then sends a RAISE command to the appropriate remote control.

279 PRINT RAISE(S,C) — Selects site S, channel C and sends a series of RAISE pulses. Returns the resulting reading.

RAISE$

280 PRINT RAISE$(S,C) — Does same as RAISE, except that a string consisting of the label, reading, and units is returned.

READ

490 READ V,W — Reads data into specified variables from a DATA statement. The first piece of data will be the first piece of data listed in the first DATA statement of the program. The second piece of data read will be the second piece listed in the first DATA statement, and so on. When all of the data have been read from the first DATA statement, the next piece of data to be read will listed in the second data statement of the program. Attempting to read more data than there is in all the DATA statements in a program will cause and OD (Out of Data) error.

50 READ X$ — Reads a string from DATA statements within the program. Strings do not have to be quoted, but

		if they are not, they are terminated by a "," or ":" character or end of line and leading spaces are ignored. See DATA for the format of string data. Note that strings read from DATA statements are <u>not</u> copied into string space. Instead, the string descriptor points to the string in the data statement (in the program). The same is the case when a string is defined using a string literal in a let statement (A$="Hello"). This saves string space for strings that are changing in the program. To force a copy into string space (which may be required in programs using bank switching), "add" a null string to a string after it is read. For example, READ A$: A$=A$+""
REM	5ØØ REM NOW SET V=Ø	Allows the programmer to put comments in her/his program. REM statements are not executed, but can be branched to. A REM statement is terminated by the end of a line, but not a ":".
	5Ø5 REM SET V=Ø: V=Ø	In this case, V=Ø will never be executed by H&F BASIC.
	5Ø6 V=Ø: REM SET V=Ø	In this case, V=Ø will be executed.
RESTORE	51Ø RESTORE	Allows the re-reading of DATA statements. After a RESTORE, the next piece of data read will be the first piece listed in the first DATA statement of the program. The second piece of data read will be the second piece listed in the first DATA statement, and so on, as in a normal READ operation. A suggested application of RESTORE is to read through the data statements once, checking for flags while counting the statements. Set up the limits for each array and dimension the arrays. Then, do a restore and read the data into the appropriate arrays. This approach (as opposed to the standard approach of putting a data count prior to the data) simplifies the changing of data, since the count need not be modified (it is determined at run time). Further, the data statements can be checked during the initial reading by seeing if the flag is in the proper "field", where each data statement can be thought of as a record of several fields that cause several variables to be loaded with data at the same record number (index of the arrays).
RETRIES	RETRIES=5 DISPLAY RETRIES	This statement sets the number of tries the DRC will make in an attempt to get data (metering or status) from a site before giving up and setting ERR. Tries are separated by 5 seconds. The

		default value is 10. This may be changed as necessary to allow for variations in communications link quality. The current value of RETRIES may also be read.
RETURN	50 RETURN	Causes a subroutine to return to the statement after the most recently executed GOSUB.
RIGHT$(X$,I)	320 PRINT RIGHT$(X$,I)	Gives the rightmost I characters of the string expression X$. When I<=0 or >255 an FC error will occur. If I>=LEN then RIGHT$ returnss all of X$.
RESET	RESET	The RESET statement simulates a power on RESET to the system, reinitializing all hardware and software. It is often used when POKE statements have been used to change system memory pointers.
RND(X)	170 PRINT RND(X)	Generates a random number between 0 and 1. The argument X controls the generation of random numbers as follows: X<0 starts a new sequence of random numbers using X. Calling RND with the same X starts the same random number sequence. X=0 gives the last random number generated. Repeated calls to RND(0) will always return the same random number. X>0 generates a new random number between 0 and 1. Note that (B-A)*RND(1)+A will generate a random number between A and B.
RUN	RUN	Starts execution of the program currently in memory at the lowest numbered statement. Run deletes all variables and restores DATA.
	RUN 100	Starts execution of program in memory at line 100. Run deletes all variables and restores DATA.
SAVE	SAVE "FN$",8	Saves program currently loaded in DRC RAM to the specified drive number (8) under the specified file name (FN$). If the comma and drive number are deleted, drive 8 is assumed.
	SAVE "@0:HI"	he same file name is to be replaced, FN$ must start with "@0:".
SAVE EEPROM		Saves the program currently in DRC RAM to non-volatile memory (EEPROM), in the boot program section of the processor board EEPROM, if the program will fit. About 1 Kbyte is available. This program can be reloaded by using the LOAD EEPROM command, or by doing a system reset. On system reset, the program is reloaded and run.
SAVE EEPROM 24576		If an address is specified after the SAVE EEPROM

		command, the program is saved in EEPROM starting at that address. This is allowed if a portion of the RAM board has been loaded with EEPROM. See the theory of operation section on the RAM board.
SAVE EEPROM 2,8192		If the system is equipped with a bank switched RAM board with the second bank battery backed, the battery backed bank may be used for non-volatile program storage. In the example, the 2 represents the bank, and the 8192 represents the address of that bank, that the program is to be stored at. Note that addresses Ø through 8191 are in common to all banks. A program stored in battery backed RAM can be loaded and run automatically on system reset using a boot program such as "1Ø LOAD EEPROM 2,8192".
SBCMD	SBCMD"NØ:DRC19Ø,HF",8	SBCMD sends the command in the string following SBCMD to the specified drive. The first example formats a disk using the NEW command (NØ:), names the disk DRC19Ø and puts an identifier (HF) on the disk. The ",8" specifies which drive the command is sent to. If the ",8" is deleted, the default drive 8 is assumed.
	SBCMD"SØ:DRC19Ø"	The Scratch command (SØ:) deletes the specified file (in this case, DRC19Ø). Wildcard characters (* and ?) are allowed, where * replaces any sequence of characters, and ? replaces any single character.
SCAN	SCAN A,B,C	SCAN sends a request to the A/D portion of the addressed site (A) telling it to send back all the readings from channel B to channel C, inclusive. This single request causes the readings to come back in channel number order where they may be captured using a FOR/NEXT loop. Such a loop might appear as: SCAN 1,Ø,9:FOR N=Ø TO 9: M(N)=METER(1,N):NEXT
SGN(X)	23Ø PRINT SGN(X)	Gives 1 if X>Ø, Ø if X=Ø, and -1 if x<Ø.
SIN(X)	19Ø PRINT SIN(X)	Gives the sine of the expression X. X is interpretted as being in radians. Note: COS(X)=SIN(X+3.14159/2) and that 1 Radian is 18Ø/PI degrees, or 1 Radian = 57.2958 degrees; so that the sine of X degrees is SIN(X/57.2958).
SPC(I)	25Ø PRINT SPC(I)	Prints I space (or blank) characters on the printer. May be used only in a PRINT or DISPLAY statement. X must be >=Ø and <=255 or an FC error will occur.
SQR(X)	18Ø PRINT SQR(X)	Gives the square root of the argument X. An FC

error will occur if X<Ø.

STATUS — 19Ø IF STATUS (S,C) THEN 3ØØ

STATUS returns a TRUE or FALSE 2ØØ ?STATUS(S,C) according to whether the status indicator at site S, channel C is ON or OFF. TRUE is represented by a -1. False is represented by a Ø.

STATUS(S,95)

STATUS(Site, 95) is true if, at that site, control has been locked out from elsewhere (the timed local function in the set up - calibration routines). STATUS(Site,95) is false if control is not locked out.

STATUS$(3)

This function returns a string representing the entire status of the specified site (3 in this case). Normally, this string has a length of 13 bytes. Status channel Ø is held in the least significant bit of the first byte of the string. Status channel 7 is held in the most significant bit of the first byte of the string. Status channel 95 is held in the most significant bit of byte 12 of the string. Status channel 96 (local) is held in the least significant bit of byte 13 of the string (and duplicated in all the other bits of this byte). A TRUE status is represented by a 1 in the byte. A FALSE status is represented by a Ø. A very quick check to determine if the status of a site has changed can be accomplished with the following code: IF STATUS$(3)<>OS$(3) THEN OS$(3)=STATUS$(3) GOTO NNNN, where OS represents the "old status", 3 is the site of interest, and NNNN is the line number of the code to execute on discovering a change in status. Further, status changes can easily be buffered for later reporting in a circular buffer constructed from a string array. Finally, NOTE that STATUS$ returns a null string and sets ERR if it is unable to get a response in RETRIES tries. You may find the ASC(A$,N) function helpful in taking apart status strings for further evaluation.

STATUS(32)=-1

This status statement allows "software" status to be set. The number (or expression) in parenthesis corresponds to the status channel that is to be set or cleared (the site is the site that the statement is executed at). This number must be higher than the "highest status number" set in system set up. Setting a status to Ø sets the status false (LED out on status transceivers that are monitoring this site, open collector output open on status transceivers that

		are monitoring this site, status function of this site and channel returns Ø). Setting a status to non-zero sets the status true. This statement is typically used to control binary coded devices at remote sites, such as video routing switchers.
STEP	FOR N=Ø TO 9 STEP 1	See FOR/NEXT.
STOP	9ØØØ STOP	Causes a program to stop execution and to enter the command mode. Prints BREAK IN LINE 9ØØØ (for this example). The CRT will beep once per second until a keystroke to indicate the program execution has stopped. CONT after a STOP branches to the statement following the STOP.
STR$	29Ø PRINT STR$(X)	Gives a string which is the character representation of the numeric expression X. For instance, STR$(3.1)=" 3.1".

SWAP	SWAP A,B
	SWAP A$,B$
	SWAP A(N),A(N+1)
	SWAP A$(N),A$(N+1)
		Swaps the contents of two variables. If A=4 and B=5 prior to executing the SWAP, A=5 and B=4 after executing the swap. Swaps can be between any two numeric variables or between any two string variables. Either or both variables can be elements of an array or matrix. An attempt to swap a string variable with a numeric variable results in a TM (Type Mismatch) error.

TAB(I)	24Ø PRINT TAB(I)	Spaces to the specified print position on the terminal. May be used only in PRINT or DISPLAY statements. Zero is the leftmost column of the printer, while 131 is the rightmost. The rightmost column on the CRT varies with the CRT used. Care should be used to insure that you do not TAB beyond the capability of the CRT. If the carriage (or cursor) is beyond position I, no change is made. Note that there is only one position counter shared between the printer and CRT. In addition, the use of CRT escape sequences (such as cursor position, reverse video, wide characters, underline, blink, etc.) will cause the TAB counter to be in error, since the ESC will count as a non-printing character, but other characters of the sequence are assumed to take space, even though they actually don't on the CRT. If these codes are used, the use of direct cursor addressing is suggested (instead of TAB). This'll put the cursor in the right location and be faster.
TAN(X)	2ØØ PRINT TAN(X)	Gives the tangent of the expression X where X is in radians.

THEN		See IF THEN.
TIME	8ØØ IF TIME>=RT+2ØØØØ THEN GOSUB 1ØØØ 1ØØØ TIME=RT	TIME is an integer variable representing the time of day in 24 hour format. 2359ØØ represents 11:59:ØØ pm. Ø represents midnight. Use of TIME in comparisons allows program operation to vary with time of day. Example line 8ØØ causes the DRC19Ø to go to subroutine 1ØØØ if it has been 2 hours or more since RT was last updated. This would typically cause a set of readings to be printed every two hours. Time is set using a LET statement. TIME is updated by the internal clock. TIME cannot be set using an input or read statement.
TIME$	PRINT TIME$	Prints time (set up above) in HH:MM:SS 12 hour format.
TIMER	TIMER(1Ø)=1ØØ IF TIMER(Ø)=Ø THEN...	The DRC19Ø includes 11 timers, callled TIMER(Ø) through TIMER(1Ø). Each timer may be set to an integer value between Ø and 65534 seconds. Each second, the value in each timer is decremented until that timer holds a value of Ø. The value in a timer may be checked at any time. In the first example, timer(1Ø) is initialized to 1ØØ seconds. The second example checks to see if timer(Ø) has "timed out" (decremented to Ø), and if so, will execute the statement following the THEN statement. These timers can be used to schedule filament warm up and after cooling times, log prints, etc.
TO	FOR N=Ø TO 9	See FOR-NEXT.
TROFF	1ØØ TROFF TROFF	Turns TRace OFF. May be used within a program, or may be used in immediate or command mode.
TRON	1ØØ TRON Ø TRON Ø	Turns on program trace. With program trace on, a CRLF followed by the line number of the line about to be executed is sent to the specified I/O device, and followed by a colon and a space. TRON can be used within a program or in the immediate or command mode.
USING	PRINT USING A$,	See PRINT USING or DISPLAY USING.
USR	A=USR(Ø)	This is the User function call, allowing the call of machine language routines (similar to SYS in other systems). Prior to calling USR, the starting address of the routine must be placed in

USRLOC and USRLOC+1, with the most significant byte of the address in USRLOC. Refer to the symbol table in the firmware theory of operation section for the actuall address. The argument to the function will be left in the floating point accumulator (FAC) in four byte floating point form. The routine is expected to leave its result in the FAC. Calling USR prior to initializing USRLOC will result in a syntax error.

VAL(X$)	28Ø PRINT VAL(X$)	Returns the string expression X$ converted to a number. For instance, VAL("3.1")=3.1 . If the first non-space character of the string is not plus (+) or minus (-) sign, a digit or a decimal point (.) then zero will be returned.

OPERATORS

SYMBOL	SAMPLE STATEMENT	PURPOSE/USE
=	A=1ØØ	Assigns a value to a variable.
	LET Z=2.5	The LET is optional.
-	B=-A	Negation. Note that Ø-A is subtraction, while -A is negation.
^	13Ø PRINT X^3	Exponentiation. Sample is X raised to the third power (X*X*X). Ø^Ø=1. Ø to any other power = Ø. A^B with A negative and B not an integer gives an FC error. Note that the ^ symbol is the "caret". The printer that prints this manual does not have the proper symbol available.
*	14Ø X=R*(B*D)	Multiplication.
/	15Ø PRINT X/1.3	Division.
+	16Ø Z=R+T+Q	Addition.
-	17Ø J=1ØØ-I	Subtraction.

RULES FOR EVALUATING EXPRESSIONS:

1. Operations of higher precedence are performed before operations of lower precedence. This means the multiplication and divisions are performed before additions and subtractions. As an example, 2+1Ø/5 equals 4, not 2.4 . When operations of equal precedence are found in a formula, the left hand one is executed first: 6-3+5=8, not -2.

2. The order in which operations are performed can always be specified explicitly through the use of parentheses. For instance, to add 5 to 3 and then divide by 4, we would use (5+3)/4, which equals 2. If instead we had used 5+3/4, we would get 5.75 as a result (5 plus 3/4).

The precedence of operators used in evaluating expressions is as follows, in order beginning with the highest precedence. Operators listed on the same line have the same precedence.

1. Formulas in parenthesis are always evaluated first.
2. ^ Exponentiation.
3. Negation. -X where X may be a formula.
4. * / Multiplication and division.
5. + - Addition and subtraction.
6. Relational Operators (Equal precedence for all 6):
   - = Equal
   - <> Not equal
   - < Less than
   - > Greater than
   - <= Less than or equal
   - >= Greater than or equal
7. NOT Logical and bitwise "NOT". Like negation, NOT takes only the formula to its right as an argument.
8. AND Logical and bitwise "AND"

9. OR Logical and bitwise "OR"

Relational Operator expressions will always have a value of TRUE (-1) or a value of FALSE (Ø). Therefore, (5=4)=Ø, (5=5)=-1, (4>5)=Ø, (4<5)=-1, etc.

The THEN clause of an IF statement is executed whenever the formula after the IF is not equal to Ø. That is to say, IF X THEN. . . is equivalent to IF X<>Ø THEN. . .

SYMBOL	SAMPLE STATEMENT	PURPOSE/USE
=	1Ø IF A=15 THEN 4Ø	Expression equals expression.
<>	7Ø IF A<>Ø THEN 5	Expression doesn't equal expression
>	3Ø IF B>1ØØ THEN 8	Expression greater than expression
<	16Ø IF B<2 THEN 1Ø	Expression less than expression.
<=,=<	18Ø IF A<=1 THEN 2	Expression less than or equal to expression.
>=,=>	19Ø IF Q>=R THEN 7	Expression greater than or equal to expression.
AND	2 IF A<5 AND B>2 THEN 7	If expression 1 (A<5) AND expression 2 (B>2) are BOTH true, then branch to line 7.
OR	IF A<1 OR B<2 THEN 2	If EITHER expression 1 (A<1) or expression 2 (B<2) is true, then branch to line 2.
NOT	IF NOT Q3 THEN 4	If expression "NOT Q3" is true (because Q3 is false), then branch to line 4. NOT(-1)=Ø, NOT(TRUE)=FALSE.

STRING OPERATORS

NAME	EXAMPLE	PURPOSE/USE
= < > <= >=		String comparison operators. Comparison is made on the basis of ASCII codes, a character at a time until a difference is found. If during the comparison of two strings, the end of one is reached, the shorter string is considered smaller. Note that "A " is greater than "A" since trailing spaces are significant.
+	3Ø LET Z$=R$+Q$	String concatentation. The resulting string must be less than 256 characters in length or an LS error will occur.

SPECIAL CHARACTERS

CHARACTER	USE
Control-U	Erases current line being typed and types a carriage return/line feed.
BACKSPACE	Erases last character typed. If no more characters are left on the line, types a carriage return/line feed.
CARRIAGE RETURN	A carriage return must end every line typed in. Returns printhead or CRT cursor to first position on next line.
CONTROL C	Interrupts execution of a program or list command. The DRC returns to command level. Prints "BREAK IN LINE XXXX" , where XXXX is line number of next statement to be executed. On a program break due to a control-C or an error, or a STOP statement, the CRT beeps once per second until a keystroke to indicate that the program has stopped.
: (colon)	A colon is used to separate statements on a line. Colons can be used in direct and indirect statements. The only limit on the number of statements per line is the program line length (113 characters). It is not possible to GOTO or GOSUB to the middle of a line, although it is possible to RETURN to the middle of a line.
; (semicolon)	When used as part of a print or display statement, causes the following expression to be printed or displayed with no spaces between the expressions (note that numerics allocate a leading space for sign). If a print or display statement ends with a semicolon, the carriage return/line feed sequence normally appended is deleted. This allows the next print or display statement to print/display on the same line as the last.
, (comma)	When used as part of a print or display statement, causes the following expression to be printed or displayed at the beginning of the next "comma field". A comma field is 13 characters wide. This allows simple printer and display formatting, although the use of TAB and PRINT USING and DISPLAY USING is encouraged.
?	Question marks are equivalent to PRINT. For instance, ?2+2 is equivalent to PRINT 2+2. Question marks can also be used in indirect statements. 1Ø ? X , when listed will list at 1Ø PRINT X .

ASCII CHARACTER CODES

0	NUL	32	SPACE	64	@	96	Grave Accent
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESCAPE	59	;	91	[	123	Left Brace
28	FS	60	<	92	Back Slash	124	Vertical Bar
29	GS	61	=	93	]	125	Right Brace
30	RS	62	>	94	Caret	126	Tilde
31	US	63	?	95	Under Score	127	DEL

NOTE that the first column of codes are referred to as "control-codes." On most keyboards, these can be generated by holding the control key while striking the corresponding key two columns to the right. For example, control-@ yields a NUL. Control-G yields a BEL, etc.

The meaning of most control codes varies between terminals. The "standard" meanings are listed below.

NUL	Null, or all zeroes	DC1	Device control 1 (X-ON)
SOH	Start of heading	DC2	Device control 2
STX	Start of text	DC3	Device control 3 (X-OFF)
ETX	End of text	DC4	Device control 4
EOT	End of transmission	NAK	Negative acknowledge
ENQ	Enquiry	SYN	Synchronous idle
ACK	Acknowledge	ETB	End of transmission block
BEL	Bell or alarm	CAN	Cancel
BS	Backspace	EM	End of medium

HT	Horizontal tabulation	SUB	Substitute
LF	Line feed	ESC	Escape
VT	Vertical tabulation	FS	File separator
FF	Form feed	GS	Group separator
CR	Carriage return	RS	Record separator
SO	Shift out	US	Unit separator
SI	Shift in	SP	Space
DLE	Data link escape	DEL	Delete

Note that DC1 and DC3 are commonly called X-ON and X-OFF. These are used for software data flow control. This the only method of flow control available between the DRC190 and a CRT terminal. Should the terminal buffer fill, it sends an X-OFF to the DRC, which stops sending data until it receives an X-ON. Note, however, the DRC190 includes an X-OFF time out so that data will be transmitted after a 255 second pause, whether an X-ON has been received or not. This feature is included so that an accidental X-OFF does not halt the DRC system indefinitely.

ERROR MESSAGES

After an error occurs, H&F BASIC generally returns to the command level and types OK. If the error occurred during the execution of a program (instead of during an immediate command), the error will be displayed and the console bell (or beep) will signal once a second until any key is pressed on the console. This prevents an error in the program from shutting the system down with no warning. The system operator is notifed of the error and is signalled until the error is acknowledged. After an error, H&F BASIC to command level, variable values and the program text remain intact, but the program cannot be continued, and all GOSUB and FOR context is lost. Variables can be inspected using PRINT or DISPLAY to see why the program crashed.

When an error occurs in a direct statement, no line number is printed. The error message format is:

Direct Statement	XX ERROR
Indirect Statement	XX ERROR IN YYYY

In each case, XX is the error code and YYYY is the line number where the error occured for the indirect statement.

The error codes and their meanings follow:

BS Bad Subscript. An attempt was made to reference a matrix element which is outside the dimensions of the matrix. This error can occur if the wrong number of dimensions are used in a matrix reference. For example, LET A(1,1,1)=Z when A has been dimensioned DIM A(2,2).

CN Continue Error. Attempt to continue a program when none exists. an error occured, or after a new line was typed into the program.

DD Double Dimension. After a matrix was dimensioned, another dimension statement for the same matrix was encountered. This error often occurs if a matrix has been given the default dimension of 1Ø because a statement like A(I)=3 is encountered and then later in the program a DIM A(1ØØ) is found.

FC Function Call error. The parameter passed to a math or string function was out of range. FC errors can occur due to:

- a- a negative matrix subscript (LET A(-1)=Ø)
- b- an unreasonably large matrix subscript (>32767)
- c- LOG negative or zero argument
- d- SQR negative argument
- e- A^B with A negative and B not an integer
- f- calls to MID$, LEFT$, RIGHT$, PEEK, POKE, TAB, SPC, or ON GOTO with an improper argument.
- g- A METER, RAISE, LOWER or STATUS call with improper channel or site.

ID Illegal Direct. You cannot use INPUT or DEFFN statement as a direct command.

LS Long String. Attempt was made by use of concatenation operator to create a string more than 255 characters long.

NF Next without For. The variable in a NEXT statement corresponds to no previously executed FOR statement.

OD Out of Data. A READ statement was executed but all of the DATA statements have already been read. The program tried to read too much data or insufficient data was included in the program.

OM Out of Memory. Program too large, too many variables, too many FOR loops, too many GOSUBS, too complicated an expression or any combination of the above.

OS Out of string space. Too many characters have been stored as strings. Can be fixed by reusing string variables, setting variables no longer in use to the null string (""), or use the CLEAR statement to allocate more string space.

OV Overflow. The result of a calculation was too large to be represented in H&F BASIC's number format. If an underflow occurs, zero is given as the result and execution continues without any error message being printed.

RG RETURN without GOSUB. A RETURN statement was encountered without a previous GOSUB statement being executed.

SB Serial Bus error. An attempt was made to access a device (such as a disc drive) on the serial bus, and that device is not responding, or that device is in an error condition (such as a request to load a non-existant file).

SN Syntax error. Missing parenthesis in an expression, illegal character in a line, incorrect punctuation, etc.

ST String Temporaries. A string expression was too complex. Break it into two or more shorter ones.

TM Type Mismatch. The left hand side of an assignment statement was a numeric variable and the right side was a string, or vice versa; or, a function which expected a string argument was given a numeric one or vice versa.

UF Undefined Function. Reference was made to a user defined function which had never been defined.

US Undefined Statement. An attempt was made to GOTO, GOSUB, or THEN to a statement which does not exist.

/Ø Division by Zero.

SPEED HINTS

The execution of a program is generally determined by the speed of I/O devices (the DRC modems, CRT terminal and printer), however, programs can be sped up by using the space hints in the next section plus the following:

1. Define frequently used variables early in the program. Variables are stored in the order they are encountered. Putting frequently used ones at the start of the list minimizes search time.

2. Deleting the index variable in NEXT statements slightly improves speed.

3. Organizing programs in the order listed below. Line number searches (used in Gosub and GoTo statements) can take a significant amount of time. If a searched line number is less than the line currently being executed, the search begins at the beginning of the program. If the searched line number is greater than the line currently being executed, the search begins at the current line. Putting globally called subroutines at the beginning of the program allows the entire program to find these routines quickly. Putting locally called routines immediately behind (nigher line number) the calling routine allows the calling routine to find this routine quickly while not slowing down the search for global routines. Since initialization code is run only once, the speed with which it is found has little impact on system speed.

Suggested Program Organization

1. Global Subroutines, called from several places in the program
2. Main Program
3. Local Subroutines, each followed immediately by any local routines they call.
4. Initialization Code.
5. Initialization DATA statements.

Note also that initialization code and data statements can be in a separate initialization program. See the information on the LOAD statement.

SPACE HINTS

In order to make your program smaller and save space, the following hints may be useful.

1. Use multiple statements per line. There is a small amount of overhead (5 bytes) associated with each line in the program. Two of the five bytes contain the line number in binary. This means that no matter how many digits in your line number (minimum line number is Ø, maximum is 64ØØØ), it takes the same number of bytes (although the use of short line numbers results in a slight reduction of space when that line number is GOTOd or GOSUBd, since the line number following a GOTO or GOSUB is stored in ASCII with one byte per charachter). Putting as many statements as possible on a line will cut down on the number of bytes used by your program (including the combination of data from several data statements into fewer statements - data compression).

2. Delete all unnecessary spaces from your program.

3. Delete all REM statements

4. Use variables instead of constants.

5. A program need not end with END, so an END statement can be deleted.

6. Reuse the same variables.

7. Use GOSUBs to execute sections of program statements that perform identical actions.

8. Use the zero elements of matrices.

9. Break the program into two parts, initialization and run-time. Load the initialization code and data, execute it (initializing all variables), load the runtime code (deleting the initialization code while retaining the variables).

Storage Allcoation Information

Simple (non-matrix) variables use 6 bytes; 2 for the name, 4 for the value. Simple non-matrix string variables also use 6 bytes; 2 for the variable name, 2 for the length, and 2 for the pointer.

Matrix variables use a minimum of 12 bytes. Two bytes are used for the name, two for the size of the matrix, two for the number of dimensions, and two for each dimension aong with 4 bytes for each of the matrix elements

String variables also use one byte of string space for each character in the string. This is true whether the string is a simple string variable or an element of a string matrix.

When a new function is defined by a DEF statement, 6 bytes are used to store the definition.

Reserved words like FOR, GOTO, NOT and the names of any intrinsic functions such as COS, INT, and STR$ take only one byte of program storage. All other characters in programs use one byte of program storage each.

When a program is being executed, space is dynamically allocated on the stack as follows:

22 bytes for each active FOR NEXT loop.
6 bytes for each active GOSUB
4 bytes for each parenthesis in an expression
12 bytes for each temporary result in an expression

DERIVED FUNCTIONS

The following functions can be calculated using existing H&F BASIC functions.

SECANT(X)	1/COS(X)
COSECANT(X)	1/SIN(X)
COTANGENT(X)	1/TAN(X)
ARCSINE(X)	ATN(X/SQR(-X*X+1))
ARCCOS(X)	-ATN(X/SQR(-X*X+1))+1.5708
ARCSEC(X)	ATN(SQR(X*X-1))+(SGN(X)-1)*1.5708
ARCCSC(X)	ATN(1/SQR(X*X-1))+(SGN(X)-1)*1.5708
ARCCOT(X)	-ATN(X)+1.5708
SINH(X)	(EXP(X)-EXP(-X))/2
COSH(X)	(EXP(X)+EXP(-X))/2
TANH(X)	-EXP(-X)/(EXP(X)+EXP(-X))*2+1
SECH(X)	2/(EXP(X)+EXP(-X))
CSCH(X)	2/(EXP(X)-EXP(-X))
COTH(X)	EXP(-X)/(EXP(X)-EXP(-X))*2+1
ARGSINH(X)	LOG(X+SQR(X*X+1))
ARGCOSH(X)	LOG(X+SQR(X*X-1))
ARGTANH(X)	LOG((1+X)/(1-X))/2
ARGSECH(X)	LOG((SQR(-X*X+1)+1)/X)
ARGCSCH(X)	LOG((SGN(X)*SQR(X*X+1)+1/X)
ARGCOTH(X)	LOG((X+1)/X-1))/2

Monitor & EEPROM Initialization

The DRC19Ø firmware includes a "monitor" program to aid in troubleshooting.

This program can be reached in a couple of different ways. If the Basic program is operating properly, typing MONITOR at the command level will drop you into the monitor. If the Basic program is not operating properly (which can happen if the EEPROM gets crashed), the monitor program can be reached by generating a Non-Maskable Interrupt (NMI) on the processor. This is accomplished by grounding pin 6 on the processor itself momentarily, or grounding momentarily the wire wrap in installed on the back-plane between the A/D boards (if any) and the remainder of the boards in the system. When the monitor is called from Basic, the terminal port is left at the existing speed. If the monitor is called with an NMI, the serial port is set to 96ØØ bits/second.

When the monitor is initiated, the processor status should appear on the screen.

The processor status display has the following meaning:

S=ØØ78	Contents of the stack pointer
C=22	Contents of the condition code register
B=FF	Contents of accumulator B
A=ØØ	Contents of accumulator A
X=FFØØ	Contents of the index register
P=ØØCØ	Contents of the program counter

Note that the monitor initialization sets the stack pointer to $ØØFF to insure the monitor stack is in RAM. The stack indication shown is actually 7 bytes below the stack pointer address prior to the NMI (which pushed everything on the stack).

Monitor commands include:

P	Prints the above processor status
B	Show the contents of accumulator B *
A	Show the contents of accumulator A *
X	Show the contents of index register X *
Mnnnn	Show contents of memory location nnnn where nnnn is in hex *
I	Initialize EEPROM
Gnnnn	Go start executing a program at address nnnn (hex).

B, A, X and M allow the current contents of the register or location to be changed by keying in the 2 digit or 4 digit hexadecimal replacement. If no change is required, type carriage return.

Typing a Line Feed after M has displayed a memory location will show the contents of the next memory location.

The EEPROM holds calibration data used by the A/D converter subroutines. A new EEPROM will typically have all one's programmed in every address. This will cause a errors in the A/D routines, preventing the DRC19Ø from operating properly.

To initialize the EEPROM, type I while in the monitor. When the terminal does a carriage return and line feed, the EEPROM has been initialized.

EEPROM initialization sets all scaling factors to 1 (causing the displayed reading to be the A/D conversion in hundreds of microvolts). All labels and units are set to question marks. The site number is set to zero, the maximum site number to 1. The site delay is set to 1ØØ mS. The CW ID frequency is set to Ø. The CW ID message is initialized to "H&F DRC19Ø[". The modem speed is set to 12ØØ bits per second. The terminal speed is set to 9.6 K bits per second.

Note that the monitor program does not reset the watchdog timer, so you can only stay in the program for about 3Ø seconds.

Adjustments PC1441 A/D Board

The closest thing to an adjustment on the A/D board are the address select jumpers. These jumpers (along with the memory map PROM on the processor board) determine where the A/D board resides in the system memory map. This determines which channels this A/D board covers.

The A/D boards reside at $ZZXØ where X is the board number, and ZZ is the base address of A/D board Ø (identified in the symbol table in the firmware theory of operation section as ADØ). Board Ø covers channels Ø-9; board 1 covers channels 1Ø-19. . . Board 9 covers channels 9Ø-99. The board number is determined by the bottom 4 programming jumpers on PØ1 on the A/D board. The jumper is in place to program a Ø and is removed to program a 1. Wtih this in mind, the below table can be used to determine the proper jumper positioning for the desired board number.

Channels	Board	Top-Jumpers-Bottom
ØØ..Ø9	Ø	Ø Ø Ø Ø Ø Ø Ø Ø
1Ø..19	1	Ø Ø Ø Ø Ø Ø Ø 1
2Ø..29	2	Ø Ø Ø Ø Ø Ø 1 Ø
3Ø..39	3	Ø Ø Ø Ø Ø Ø 1 1
4Ø..49	4	Ø Ø Ø Ø Ø 1 Ø Ø
5Ø..59	5	Ø Ø Ø Ø Ø 1 Ø 1
6Ø..69	6	Ø Ø Ø Ø Ø 1 1 Ø
7Ø..79	7	Ø Ø Ø Ø Ø 1 1 1
8Ø..89	8	Ø Ø Ø Ø 1 Ø Ø Ø
9Ø..99	9	Ø Ø Ø Ø 1 Ø Ø 1

If desired, a voltage divider may be added in front of the A/D converter. This allows for sample voltages higher than the normal limit of +/- 2 volts. Normally, a jumper is shipped in the socket for R2Ø (the series resistor of the voltage divider) and R21 (the shunt resistor) is left open. The user may plug resistors into these locations to allow for higher sample voltages. For example, if R2Ø is 6Ø K and R21 is 1ØK, a maximum sample voltage of 14 volts would be allowed. The A/D would then have a resolution of 1.4 mV. The resistors used in this voltage divider should be very stable to minimize the introduction of errors. It is suggested that resistors with a temperature coefficeint of 5 ppm/degree C be used.

Adjustment of PC1442 Processor Board

The processor board has several adjustments. Most are connected with the modem.

The one adjustment that is not connected with the modem is the LCD viewing angle. Multiplexed LCDs have a limited viewing angle. The viewing angle adjustment is used to insure that the optimum viewing angle of the LCD is the angle that will actually be used. Outside the viewing angle range the LCD loses contrast, making it difficult to read.

It is suggested that the DRC19Ø be installed with the display at "eye-level". When the DRC19Ø is on the test bench and the cover is removed, adjust R37 for optimum contrast at the desired viewing angle.

Modem Adjustments

For the modem adjustments, it is suggested that the DRC19Ø be set up for two-wire telephone line operation. This can be done by jumpering pins 3 to 4 and pins 16 to 17 on J21 on the rear panel. Connect an audio voltmeter and a frequncy counter to pins 3 and 16 of J21. Connect a CRT terminal to J22 on the rear panel.

On the CRT terminal, type MODEMTST followed by a carriage return. Instructions for the modem adjustment will appear on the CRT. For preview, these instructions are:

Adjust R1Ø for desired TX level. Normally, this is Ø dBm as measured with the audio voltmeter.

Mark frequency 2Ø9Ø - 231Ø Hz. There is no adjustment for Mark frequency. We are just checking to insure it is within tolerance. If it is not, R15, CØ4 or U1Ø should be checked.

Space frequency 114Ø - 126Ø Hz. There is no adjustment for Space frequency. We are just checking to insure it is within tolerance. If it is not, R14, CØ4 or U1Ø should be checked.

Modem demodulator tuning is checked by watching the output od the demodulator for bias distortion. Should the mark/space threshold be at the wrong frequency, mark pulses will be a differnt duration than space pusles. This test sends all the alpahnumeric characters through the modem and displays the received characters. The CRT should show this stream of characters in a recognizable alphabetical order. In addition, the modem tuning can be checked by checking the length of a received bit on pin 7 of the XR2211. This bit length should be 833 uS when the modem is running at 12ØØ bits per second. This test also does not include a watchdog timer reset. Running this test along with the last test for more than 3Ø seconds should result in a system reset. This is a good way to check the operation of the watchdog timer.

Adjust R41 for CW ID level. The DRC19Ø generates the CW ID tone using a timer in U16. The timer divides the system clock down to the FCC required CW ID frequency of 75Ø Hz (+/- 1Ø Hz). The tone is filtered from a square wave to a triangle wave. The FCC requires the CW ID level to be at 4Ø% modulation (+/- 1Ø%). Most transmitters used in TRL service (including those supplied by H&F) have a 6dB/octave preemphasis. Based on this, if the transmitter is adjusted to yield 1ØØ% modulation (1.5 KHz deviation) when driven with 2.2 KHz at Ø dBm (Ø.776 volts RMS), R41 should be adjusted for Ø.91Ø volts RMS at 75Ø Hz.

As the DRC sets up for each adjustment, it puts the instructions for the adjustment on the CRT. When the adjustment is complete, pressing any key on the CRT keyboard will advance you to the next instruction.

This completes the adjustment of the PC1442 processor board.

Adjustment PC1443 Power Supply Interface

The PC1443 board serves three purposes: Disc drive interface, Status panel interface, and Power supply interface.

If the disc drive and status interface is provided, programming jumpers on PØ1 and the memory mapping PROM on the processor board determine the address of the interface. This interface is normally placed at $9ØFØ (DISCSTAT in the symbol table in the firmware theory of operation section), requiring jumpers as shown below. A 1 indicates the lack of a jumper, while a Ø indicates the presence of a jumper.

Address Line	Jumper Position	Jumper*
11	Top	Ø
1Ø		Ø
Ø9		Ø
Ø8		Ø
Ø7		1
Ø6		1
Ø5		1
Ø4	Bottom	1

If the DRC was provided with an Uninterruptable Power Supply (UPS), then RØ2 is used to adjust the battery charger voltage. Disconnect one of the battery leads and connect a DVM to the two battery leads (reading the charger voltage). RØ2 should be adjusted to give 14.ØØ volts.

Connect the battery leads.

Power Supply Voltage Adjustment

When making battery charger voltage adjustments, it would be a good idea to check the 5 volt output of the main power supply. Due to wiring and connector voltage drops, the power supply voltage should be checked in the following manner.

Clip the negative lead of a digital voltmeter to the top motherboard mounting bracket (which provides a system ground when the cover is in place). Connect the positive lead of the DVM to the top horizontal trace on the component side of the processor board. This trace should be at +5.ØØ volts. The processor will not work properly should the voltage be outside the range of +4.75 volts to +5.25 volts. A screwdriver adjustment for the power supply voltage is on the power supply board.

Subcarrier Transceiver Board 1444 Adjustment

There are several adjustments on the subcarrier transceiver board. Some of these are outlined in the installation section. They will all be covered here. The adjustments are broken into two groups: those on the subcarrier generator, and those on the subcarrier demodulator. We'll start with the subcarrier generator.

Subcarrier Transmit Frequency

Connect a frequency counter to the subcarrier output on the rear panel of the DRC19Ø or on PØ1-1 on the subcarrier board. Adjust RØ6 to get the frequency within 1 KHz of the desired frequency. Adjust RØ7 to get the exact frequency.

Subcarrier Deviation

Using a jumper, apply the +5 volt logic supply to the audio input of the subcarrier generator (PØ3-1). Adjust RØ9 so that the subcarrier frequency is 5 KHz offset from the frequency with no applied voltage. This corresponds to the recommended deviation of 1 KHz/volt. Note, however, that with a subcarrier frequency of 26 KHz, the maximum obtainable deviation is 692 Hz. This will be fine for DRC19Ø control/metering operation.

Subcarrier Distortion

If the controls are far out of adjustment, connect an oscilloscope to the subcarrier output. Adjust RØ1 for a symmetrical waveform. Then connect a distortion analyzer or a spectrum analyzer to the subcarrier output. Alternately adjust RØ1 and RØ2 for minimum harmonic distortion.

Receive Local Oscillator Null

With no signal connected to the subcarrier input, adjust R11 for a null in the signal on UØ2 pin 2, as observed with an oscilloscope.

Receive Local Oscillator Frequency

To improve the local oscillator null, there is normally a jumper installed on PØ5. Remove this jumper when measuring the local oscillator frequency. With no signal connected to the subcarrier input (thereby disabling the AFC), adjust R17 for the desired local oscillator frequency, as measured on PØ5-1. The desired local oscillator frequency is 455 KHz - SCA. where SCA is the desired SCA receive frequency. For example, to receive 11Ø KHz, the local oscillator frequency should be 345 KHz. Return the jumper to PØ5.

The local oscillator frequency can be adjusted to center the received signal in the FM discriminator. To accomplish this adjustment, measure the voltage on pin 7 of the LM324 using a digital voltmeter. Adjust R17 to yield exactly Ø volts DC.

Audio Select Jumpers

PØ3 is used to select whether DRC19Ø audio should be sent to the subcarrier generator, and whether received subcarrier audio should be sent to the DRC19Ø. If the DRC19Ø is to drive the subcarrier generator, put the top two programming jumpers in place. Otherwise, put them on one pin each only.

If the DRC19Ø is to receive demodulated subcarrier signals, put the bottom two programming jumpers in place.

Subcarrier Frequency	Local Oscillator Frequency
26 KHz	429 KHz
67 KHz	388 KHz
92 KHz	363 KHz
11Ø KHz	345 KHz

Adjustment PC1445 Direct Connect Modem

The direct connect modem board has no adjustments other than its address in the memory map. The board normally resides at $90E0 (See DCMDM in the symbol table in the firmware theory of operation section). This requires the jumpers to be installed as below. A 1 indicates the absence of a jumper, while a Ø indicates the jumper is present.

Top	1
	1
	1
Bottom	Ø

Adjustments PC1449 Status Transceiver Board

The status transceiver requires no adjustment, except for the setting of the site number that the receiver portion of the board is to display. This is encoded in binary on PØ5 with the least significant bit of the site number at the top. For example, to display the status of site 1, the top jumper is left open while the remainder of the pin pairs have the programming jumpers in place.

If the selected display site is 1ØØ, the LEDs will display the status of whatever site is being displayed on the LCD on the DRC19Ø front panel. For example, with the programming jumpers set to 1ØØ decimal (bottom to top Ø 1 1 Ø Ø 1 Ø Ø where Ø is jumper in place, 1 is jumper absent), and the operator keys in Ø 1 2 3, the LCD will show the reading at site Ø1, channel 23, and the LEDs will show the status of site Ø1.

With this in mind, the programming jumpers can be set so that the LEDs on the status transceiver display the status of whatever site is desired.

Jumper	Binary Weight
Top	1
	2
	4
	8
	16
	32
	64
Bottom	128

STX191 Adjustments

If the STX191 is used with a DRC190 that includes internal status, there are no adjustments on the STX191. The output and LED site number is determined by the programming jumpers on the status transceiver board in the DRC190.

If the STX191 is used with a DRC190 that does not include internal status, programming jumpers on the left board (viewed from the front of the unit) may be set to determine which site the LEDs and outputs of the STX191 are to reflect. This is encoded in binary on P05 with the least significant bit of the site number at the top. For example, to display the status of site 1, the top jumper is left open while the remainder of the pin pairs have the programming jumpers in place.

If the selected display site is 100, the LEDs will display the status of whatever site is being displayed on the LCD on the DRC190 front panel. For example, with the programming jumpers set to 100 decimal (bottom to top 0 1 1 0 0 1 0 0 where 0 is jumper in place, 1 is jumper absent), and the operator keys in 0 1 2 3, the LCD will show the reading at site 01, channel 23, and the LEDs will show the status of site 01.

Recall, however, that the two status receivers may be programmed to reflect the status of two sites instead of being cascaded.

With this in mind, the programming jumpers can be set so that the LEDs on the status transceiver display the status of whatever site is desired.

Jumper	Binary Weight
Top	1
	2
	4
	8
	16
	32
	64
Bottom	128

DRC190 Firmware Theory of Operation

This section will give you a general idea of the major portions of the DRC190 firmware. The firmware can be broken into two portions: The Basic interpreter and Everything Else.

The Basic Interpreter is licensed from Microsoft Corporation. As Microsoft has written many of the Basics in common usage, it should be familair to most.

The interpreter can either execute programs from memory (RAM) or can execute single line commands that are entered through the RS-232 port.

Several commands and functions have been added to Basic. These include METER, METER$, RAISE, RAISE$, LOWER, LOWER$, TIME, TIME$, DATE, DATE$, DAY, DAY$, LINE, plus several others. Most of these functions and commands request information from the "everything else" portion of the firmware. If this information is immediately available (such as TIME), it is immediately returned. If the requested data is not immediately available (such as METER), the Basic program awaits return of the data from the other portion of the firmware.

Basic is running in the "background", that is, there are several other processes interrupting it.

The interrupting processes include routines to handle:

Transmit through FSK modem.
Receive from FSK modem.
Check front panel keyboard.
Update front panel display.
Select an A/D channel.
Interpret the A/D sample and return it to requesting routine.
Update clock/calendar.

The front panel keyboard and LCD are considered a device that can transmit or receive data from other devices. Other devices include the A/D converter and Basic. As required, each of these devices generates messages in a message buffer. Part of the message includes to and from address for the message. Once the message is built, a routine called XFERMESS is called, transferring the message to the addressed device. If the addressed device is not at this site, the entire message is sent to the modem. When the modem receives a message, it is dumped into XFERMESS if the site number of this site matches the to site of the message.

Most of the devices that receive data through XFERMESS are not fast enough to take the data at high speed. Therefore, there are "circular" buffers at the input of several routines. There are circular buffers for the modem, the LCD, and the A/D converter.

Each of these interrupt routines is checked every 10 mS. The DUART is programmed to generate an interrupt every 10 mS.

The modem routines are quite interesting. It is necessary to insure that only one DRC190 transmits data at a time. This is accomplished by having all units time from the last valid character received. After a site delay (typically 50 mS) all site counters are incremented. If the site whose number is in the site counter has data to transmit, it brings up its carrier and starts transmitting data. If there is no data to be transmitted at this site, the lack of data is detected by all sites. All sites then increment their site counters, enabling the next site to transmit. This sequence is continued until either a site responds, or we reach the highest site number in the system. After the highest site number in the system is allowed to respond, we start timing for site Ø again. This scheme allows every site to respond while insuring that no contention takes place.

Timing from the last valid character instead of from carrier drop allows the system to be used on noisey circuits, which might false the carrier detect circuit (note, the falsing of the carrier detect circuit can be minimized by increasing the time constant of the lock detect filter, but this slows the system down). With a noisey circuit, the modem carrier detect would trip on the noise, preventing a site from transmitting data. The "time from last valid character" scheme checks parity on each character as it is received from the modem and insures that valid carrier is present. If these two conditions are met, the site delay timer is reset. This happens each time a character is received. SiteDelay mS after the last character of a transmission is received, the site delay timer increments the site counter, authorizing one site at a time to transmit data. Since all sites received the same data, the site delay timers and site counters are in synchronism, insuring orderly communications between the sites. A hex FE is transmitted during the site delay time on carrier bring up. Since the serial data is transmitted least significant bit first, the space start bit is followed immediately by a space data bit. With 8 bit words and even parity, this forces the parity bit to be a mark during this "data leader". Since the only space bits are the start bit and the least significant data bit, which immediately follows the start bit, the receiving serial port synchronizes properly immediately. The reception of "data leader" characters resets the site delay timer, preventing any additional sites from being authorized to transmit. All sites should stay in synchronism based on this resetting of the site delay timer. To insure further that they stay in synchronism, upon reception of a valid message (all byte parities and the message checksum are valid) the "from site" byte of the received message is stored in the site counter. This insures that all sites agree that the site that is transmitting the data (and from which they have just received a message) is indeed the site authorized to transmit. When the authorized site is finished transmitting data, the lack of reception of valid characters at each of the sites causes the site delay timer to not be reset. When this timer times out, the site counters at all sites are advanced, authorizing the next site in the sequence to transmit, if it has anything to transmit. Synchronism is insured by updating the site counters at each site with the site number that originated the valid message just received. This scheme results in all sites having equal access to the system and reduces delays before a higher numbered site is authorized to transmit.

Each message includes a flag sequence followed by a message header. The flag sequence is used to indicate the beginning of the header. It is important that the beginning of a message be identified without error, as the message header information indicates which site originated the message, and identifies where the message is going. The message header is preceeded by a 2 byte flag sequence, hex AA55. AA55 is alternating ones and zeroes, with the bits inverted between the first and second byte. After completing the reception of a message or dropping out of the reception of a message because it was invalid (parity or checksum error), the routine waits for the AA55 sequence. It then assumes that the header follows. Although an AA55 sequence could occur in the data portion of a message, the routine is not looking for a flag during this portion of the message. If the receive portion should lose synchronism and detect an AA55 in the data portion of a message as a flag, it is quite unlikely that the required 8 bit checksum would show up in the message. For this reason, "byte stuffing" is not used to insure that a flag sequence occurs only when we wish to transmit a flag.

You can see the data transmitted or received by the modem by setting DEBUG=1 (using the Basic statement DEBUG=1). If DEBUG is 1, received data (in hexadecimal) will be sent to the terminal plugged into the RS232 port. The data has this format:

$FE Leader data used to sync system and allow for carrier detect delays
$AA55 Flag sequence used to detect the begin of a message header
To Site Number in HEX
To Channel number in Hex
To Device Number in Hex
From Site number in Hex
From Channel number in Hex
From Device Number in Hex
Byte Count for following ASCII message, if any. No message makes this ØØ
Ø or more bytes of ascii message
8 bit checksum

The entire message as shown above will be printed. In addition, if the checksum indicates that the message is valid, an exclamation point will be printed at the end of the message.

If DEBUG is set to 2 (Basic DEBUG=2), the data transmitted by the modem will be displayed. This data takes the same form as above, except no exclamation point is printed.

To return the DRC19Ø to normal, type DEBUG=Ø on the terminal.

Symbol Table

Below is printed the symbol table for the DRC19Ø firmware revised as of July 2Ø, 1989. Further revision of the firmware (to add features) will change the addresses shown in the following table. Revised manuals or revised symbol tables will be available.

The symbol table shows the name of the symbol followed by the address of that symbol, in hexadecimal. The symbol table can be referred to in order to insure that programming jumpers are set correctly, or that PEEK and POKE statements to provide functions not supported by Basic are using the correct address. Note that PEEK and POKE will require the conversion of the hex address to decimal.

pages 179 through 183 deleted

---- SYMBOL TABLE ----

	B8DF
	B8DF
ABINT	B76C
ABS	CØØE
ADØ	9ØØØ
ADBLAST	Ø441
ADBUF	Ø3DE
ADBUFIN	Ø3D9
ADBUFOUT	Ø3DB
ADCAL	EBØ2
ADCOM	DF62
ADCOM1	DF6A
ADCOM11	EØ4C
ADCOM2	DFB7
ADCOM2Ø	EØ6D
ADCOM22	EØ72
ADCOM222	EØ8F
ADCOM23	EØ94
ADCOM24	EØ99
ADCOM26	EØB6
ADCOM27	EØCØ
ADCOM28	EØC9
ADCOM29	EØD3
ADCOM295	EØDD
ADCOM3	DFE5
ADCOM3Ø	EØE4
ADCOM4	DFEE
ADCOM42	EØØB
ADCOM5	EØ28
ADCOM7	EØ2D
ADCOM8	EØ3A
ADCOM9	EØ43
ADCOMDE	Ø45A
ADCOMGE5	DFC5
ADCOMGET	DFBA
ADCOMPT	Ø458
ADCOMRTA	Ø45B
ADCOMTMP	Ø45E
ADCOMTMR	Ø463
ADD16BIT	E2F9
ADDATA	Ø44B
ADDEND	ØØ98
ADDIND	B86C
ADDTMP	AB3E
ADDXAB	AB3C
ADDXB	AB3B
ADINIT	E2C2
ADIRQ	E11A
ADIRQ1Ø	E11F
ADIRQ2Ø	E13D
ADIRQ4Ø	E187
ADIRQ41	E193
ADIRQ44	E1CØ
ADIRQ46	E1D3
ADIRQ48	E1DC
ADIRQ5Ø	E1FD
ADIRQ52	E212
ADIRQ54	E234
ADIRQ56	E242
ADIRQ58	E24A
ADIRQ6Ø	E24E
ADIRQ62	E25Ø
ADIRQ65	E271
ADIRQ66	E26Ø
ADIRQ67	E272
ADIRQ68	E27B
ADIRQ7Ø	E296
ADIRQ72	E2A6
ADIRQ75	E2B9
ADIRQPTR	Ø464
ADNTVAL	Ø44D
ADOVRFL	Ø3DD
ADPRFN	Ø531
ADPRRTN	Ø53Ø
ADRABI	Ø115
ADRGAB	Ø117
ADSCAN	DFCF
ADSCAN5	DFC8
ADSCMX	Ø44E
AFFRTS	AD89
ANDMSK	ØØCØ
ANDOP	B628
APR	D983
ARGEXP	ØØD8
ARGHO	ØØD9
ARGLO	ØØDB
ARGMO	ØØDA
ARGSGN	ØØDC
ARISGN	ØØDD
ARYPNT	ØØC5
ARYSIZ	Ø19A
ARYSTR	BA6F
ARYTAB	ØØ9E
ARYVA2	BA5Ø
ARYVA3	BA52
ARYVAR	BA4E
ASC	BB71
ASC2	BBB1
ASC2ARG	FFFF
ASC5	BB9F
ASC7	BBA7
ASCIIBCD	E369
ASCIIFAC	D643
ASCMORS5	EAB2
ASCMORS8	EACØ
ASCMORSE	EAA4
ATN	C4Ø6
ATN1	C4ØD
ATN2	C419
ATN3	C429
ATN4	C43Ø
ATNCON	C431
ATNFIX	A823
AUG	D997
BANKSEL	9ØCØ
BASBUF	Ø466
BASBUFRD	Ø484
BASCHAN	Ø488
BASECHO	Ø1C1
BASMRLS	Ø486
BASRETX	Ø48D
BASSITE	Ø487
BASSTA$2	D5AA
BASSTA$4	D5AA
BASSTA$5	D5D2
BASSTA$8	D5DB
BASSTA$9	D5E5
BASSTA1Ø	D572
BASSTA12	D57D
BASSTA13	D588
BASSTA14	D58F
BASSTAT	D542
BASSTAT$	D598
BASSTAT2	D542
BASSTAT5	D56A
BASSTRNG	Ø485
BASTIME	Ø48A
BCDASCII	D83C
BCDBIN	E2DD
BINASCII	D839
BINBC1	DBØ1
BINBC2	DBØ8
BINBC3	DBØF
BINBCD	DAF3
BINLCD	FA6C
BITPOS	E2C9
BITPOS2	E2DØ
BITPOS4	E2D6
BITS	ØØD7
BLTLOP	ABØ4

BLTLOP2	AB11
BLTU	AAF7
BLTUC	AAF9
BRASCII	FB65
BREAKOK	AEEE
BREAKSWI	ØØØØ
BRKTXT	AAC8
BRTABLE	FB75
BSERR	B7DD
BSERR7	B8A9
BSPAC	B2D8
BUF	ØØØ8
BUFBOT	Ø2BD
BUFCHECK	E6ØD
BUFCHK1	E61C
BUFCHK2	E61F
BUFFULL	D167
BUFFULL8	D183
BUFFULL9	D184
BUFGET	E62F
BUFGET5	E641
BUFGET8	E64C
BUFIN	Ø2B8
BUFLEN	ØØ7Ø
BUFLNM	ØØØ6
BUFM8	E6ØA
BUFM9	E6ØB
BUFMESS	D1Ø2
BUFMESS1	D127
BUFMESS2	D12C
BUFMESS4	D143
BUFMESS7	D14A
BUFMIN	ØØØ7
BUFMT	E5FF
BUFOUT	Ø2BA
BUFPTR	ØØDF
BUFPUT	D154
BUFPUT5	D162
BUFTOP	Ø2BF
C1	FC52
C2	FC5C
C3	FC63
C4	FC76
C5	FC89
C6	FCA8
C7	FCBF
CADADDR	Ø449
CALØ	EB11
CALØØ	EB12
CALØ1	EB17
CALØ2	EB49
CALØ3	EC8A
CALØ4	ECB7
CALØ6	ECBF
CALØ7	ECEB
CALØ9	EDØ1
CAL1Ø	ED5E
CAL12	ED8F
CAL13	ED95
CAL14	EDA6
CAL15	EDA9
CAL16	EDBØ
CAL2Ø	EDC1
CAL22	EDD9
CAL24	EDEA
CAL26	EDF9
CAL3Ø	EE16
CAL32	EE28
CAL34	EE37
CAL38	EEBD
CAL4Ø	EEE1
CAL42	EEED
CAL44	EEF9
CAL46	EFØ3
CAL48	EF5B
CAL5Ø	EF67
CAL51	EF6F
CAL52	EFAA
CAL56	EFF5
CAL565	EFFA
CAL57	FØ18
CAL58	FØ24
CAL6Ø	FØ3F
CAL62	FØ49
CAL63	FØAA
CAL64	FØBØ
CAL65	FØC2
CAL67	FØEB
CALASCII	Ø564
CALASCPT	Ø56A
CALCHPTR	Ø541
CALCURSR	Ø54Ø
CALDP	Ø563
CALKEY	FAA4
CALKEY5	FAB7
CALKEY8	FABF
CALKEY9	FACØ
CALKEYN	FA9F
CALPTR	Ø532
CALSFBUF	Ø536
CALSFPTR	Ø534
CALSTRNG	Ø543
CALTIME	Ø445
CALTXTØ	EB29
CALTXT1	EB5B
CALTXT1Ø	EF49
CALTXT11	EF8F
CALTXT12	EFC3
CALTXT13	EFD7
CALTXT16	F1ØØ
CALTXT17	FØ88
CALTXT2	EC97
CALTXT3	ECCC
CALTXT4	ED39
CALTXT6	EE44
CALTXT7	EE66
CALTXT8	EE7D
CALTXT9	EE9D
CARDROP	Ø442
CASSW	ØØØØ
CAT	BACD
CCHAN	Ø452
CCHEAD	AC31
CHANCHG	Ø491
CHARAC	ØØ78
CHEAD	AC2A
CHEADA	AC28
CHKCLS	B5BF
CHKCOM	B5C5
CHKCON	AC4Ø
CHKCON3	AC55
CHKCON4	AC62
CHKCON5	AC61
CHKNU2	B924
CHKNUM	B495
CHKOPN	B5C2
CHKSTR	B496
CHKVAL	B497
CHNG	FDDB
CHR$	BB5D
CHR$DO	BB6Ø
CHRGET	ØØE1
CHRGO2	BC2E
CHRGO5	B5CE
CHRGOT	ØØE9
CHROUT	D1C3
CHRRTS	ØØF8
CHSTOR	ØØF9
CLASSØ	ØØC5
CLASS1	ØØD8
CLASS2	ØØDB
CLASS3	ØØDB
CLEAR	AF84

Symbol Table

CLEARC	ADA2
CLRCGO	AFD6
CLRFAC	CØ83
CLRPØ	C486
CLRSTK	ADB5
CLRVAL	B663
CNCOK	ØØØ4
CNTDP	Ø11D
CNTWFL	Ø111
COLIS	AD3E
COMADO	B2B8
COMBYT	BC5C
COMMODE	Ø499
COMPC	AF39
COMPRT	B2A9
COMRATE	9BF7
COMRX	E94B
COMRX1Ø	E97Ø
COMRX2	E954
COMRX4	E965
COMRX6	E96A
COMTIMER	Ø573
COMTX	E857
COMTX1Ø	E8DC
COMTX11	E8E8
COMTX12	E9Ø7
COMTX2	E867
COMTX2Ø	E9ØF
COMTX22	E92D
COMTX24	E93B
COMTX4	E86C
COMTX6	E87F
COMTX7	E896
COMTX75	E8A2
COMTX8	E8C8
COMTXPTR	Ø571
COMTXR2	E836
COMTXREQ	E947
COMTXRQ	Ø595
COMTXRX	E85Ø
COMTXT1	E8A9
COMTXT2	E8BØ
COMTXT3	E9Ø8
CONINT	BC25
CONT	AF6F
CONTH	AC69
CONTRT	AF8Ø
CONTW	ØØØF
CONUPK	BEB6
COPNUM	B177
COPRAM	C49E
COPY	B14C
COS	C376
COSC	C3E2
COSFIX	A81D
COUNT	ØØ7A
CR	ØØØD
CRDO	B29B
CRDO2	B2A5
CRDONE	AD65
CRLF	DC88
CRTSEL	D331
CRTSELØ	D344
CRTSTR	Ø1CD
CRUNCH	ACDD
CSTIMER	Ø127
CSTIMOUT	ØØFF
CTLOKN	9CØ8
CTXRCVD	Ø594
CURLIN	ØØAC
CURMEM	AFAD
CURTOL	ØØDF
CWCOUNT	Ø52A
CWID1Ø	E9F7
CWID11	EAØ5
CWID12	EAØE
CWID14	EA1A
CWID16	EA26
CWID9Ø	EA34
CWIDIRQ	E979
CWIDIRQØ	E986
CWIDIRQ1	E987
CWIDIRQ2	E98C
CWIDIRQ4	E9A3
CWIDIRQ5	E9B3
CWIDIRQ6	E9BA
CWIDIRQ7	E9CF
CWIDIRQ8	E9D6
CWIDPT	Ø525
CWINIT	EA5C
CWINTRVL	9C15
CWOFF	EA56
CWON	EA53
CWSHIFT	Ø529
CWTEMP	Ø52B
CWTIMER	Ø523
CWTXT1	E99C
CZLOOP	AC37
DATA	BØ3F
DATAN	BØ44
DATATK	ØØ83
DATBK	B3AØ
DATE	9FFD
DATEF	D895
DATEFN	D8A7
DATEFN2	D8BB
DATEFN9	D8E1
DATES	D855
DATETK	ØØDE
DATLIN	ØØB6
DATLOP	B3E9
DATPTR	ØØB8
DATRICON	ØØØØ
DAY	9FFC
DAYF	D9E5
DAYN	D9F8
DAYS	D9C2
DAYSTR	DAØB
DAYSTR1	DAØE
DAYSTR2	DA14
DAYTK	ØØDF
DBLQTE	ØØ22
DCHAN	Ø451
DCMDM	9ØEØ
DCMDMINI	FEEE
DCMDMRAT	9C26
DCMSET1Ø	F659
DCMSET12	F662
DCMSET14	F67F
DCMSET2Ø	F696
DCRXIRQ	D234
DCRXIRQ2	D23C
DCRXIRQ4	D244
DCRXIRQ6	D269
DCRXIRQ8	D26E
DCRXIRQA	D279
DCTXST	Ø574
DEBUG	Ø1CØ
DEBUGS	CEAE
DEC	D9B9
DECALPH9	FAEE
DECALPHA	FAE7
DECCNT	ØØC9
DECMORS2	EA94
DECMORS4	EA9B
DECMORS6	EAA2
DECMORSE	EA8D
DEF	B8E9
DEFFIN	B986
DEFPNT	ØØCD
DEFRTS	B98A
DEGREE	ØØD6
DEL1	DD31

DEL9	DD34
DELAY	DD2E
DESELAD	DØE2
DESELECT	DØF5
DESELLCD	DØB6
DESELMDM	DØCC
DESTIN	Ø52E
DEVCFW	ØØØØ
DEVLCF	ØØØ1
DEVPOS	ØØØ2
DEVPRM	DE2B
DEVS	CEC6
DEVS2	CEE6
DEVS4	CEEA
DEVS6	CEEF
DEVS8	CEF2
DEVSEL	D34B
DEVSEL5	D363
DEVSEL6	D366
DEVSEL8	D374
DEVSEL9	D37A
DEVSELER	D38Ø
DEVWID	ØØØ3
DIM	B6B2
DIM1	B6B3
DIM3	B6AF
DIMCON	B6B5
DIMFLG	ØØ7B
DIMRTS	B882
DIRIS	AF5D
DISCSTAT	9ØFØ
DISPLA	B238
DISPTYP	ØØØ1
DITTIME	ØØØ5
DITTIMER	Ø524
DIV1Ø	BF11
DIVIDE	BF34
DIVNRM	BF81
DIVSUB	BF69
DNTCPY	B159
DOCMP	B6A3
DOCOND	BØ78
DOGCOUNT	Ø191
DOMASK	ØØ8Ø
DOMIN	B5D6
DOPRE1	B51A
DOPREC	B4F4
DOREL	B64E
DORES	ØØ7D
DPCNT	Ø11E
DPTFLG	ØØCA
DRC	FFFF
DSCPNT	ØØCF
DSCTMP	ØØD1
DTFST3	D945
DTFSTR	D8EE
DTFSTR1	D8F1
DTFSTR2	D8FD
DTRDCDF	CFAC
DTRDCDFØ	CFC1
DTRDCDF2	CFC4
DTRDCDFF	CFCC
DUART	AØØØ
DUARTINI	FEB1
DVØERR	BF8B
DVAR	BA7D
DVAR2	BA91
DVAR3	BA9B
DVARS	BA77
DVARTS	BAA1
EATEM	B6DA
ECHOS	CEBA
EEPBANK	Ø197
EEPBOOT	9C29
EEPCOUNT	Ø57Ø
EEPDEL	CBB6
EEPDEL2	CBBB
EEPDEL4	CBBE
EEPDELR	CBC3
EEPDEST	Ø56E
EEPDISAB	FB2E
EEPENAB	FB1B
EEPFULL	CBBØ
EEPGM	FAEF
EEPGM9	FB1A
EEPINIT2	FCF6
EEPINIT4	FCFF
EEPINIT6	FD27
EEPINIT8	FD2D
EEPINIT9	FD7C
EEPINITA	FD81
EEPLDMB	C9B9
EEPM1	FD1D
EEPM2	FD4B
EEPM3	FD7C
EEPROM	9C29
EEPROMØ	98ØØ
EEPROMS1	CAAØ
EEPROMS2	CAA8
EEPROMSN	CBC6
EEPROMSV	CA7E
EEPROMSW	CA8C
EEPROMTK	ØØE8
EEPSRC	Ø56C
EEPSVER2	CB39
EEPSVER3	CB54
EEPSVERM	CB1C
EEPSVERR	CB13
EEPSVMS1	CB84
EEPTIMER	Ø444
EEPVERR	CB9D
END	AF45
ENDCHR	ØØ79
ENDCODE	FF6B
ENDCON	AF51
ENDHFRAM	Ø596
ENDLIN	AE94
ENDMEM	ØØAA
ENDREL	B4DØ
EORMSK	ØØC1
EPROMTYP	6A78
EQL	FE82
EQLFRM	D711
EQULTK	ØØC3
ERR	AAB5
ERRBS	ØØ1Ø
ERRCN	ØØ2Ø
ERRDD	ØØ12
ERRDIR	B8DF
ERRDVØ	ØØ14
ERRF	D63A
ERRFC	ØØØ8
ERRFIN	AB6C
ERRFIN1	AB77
ERRFIN2	AB82
ERRGO1	B8E6
ERRGO2	B9E9
ERRGO3	B7E2
ERRGO4	B49E
ERRGO5	B44E
ERRID	ØØ16
ERRLS	ØØ1C
ERRNF	ØØØØ
ERRNUM	Ø489
ERROD	ØØØ6
ERROM	ØØØC
ERROR	AB4B
ERROV	ØØØA
ERRRG	ØØØ4
ERRSN	ØØØ2
ERRSO	ØØ1A
ERRST	ØØ1E
ERRSTR	D4AF

Symbol Table

ERRTAB	AA8F
ERRTM	ØØ18
ERRUF	ØØ22
ERRUS	ØØØE
EVAL	B573
EVALØ	B576
EVAL1	B57A
EVAL2	B57D
EVAL3	B59A
EVAL4	B5AF
EVLADR	BC61
EXCHQT	BØ4D
EXIGNT	B42A
EXP	C2CC
EXP1	C2DD
EXPCON	C2AB
EXPSGN	ØØCC
FAC	ØØD1
FACEXP	ØØD1
FACHO	ØØD2
FACLO	ØØD4
FACMO	ØØD3
FACOV	ØØDE
FACSGN	ØØD5
FADD	BD25
FADD1	BD44
FADD2	BD5B
FADD3	BD52
FADD4	BD4E
FADD5	BD96
FADDC	BD31
FADDH	BD15
FADDT	BD28
FADFLT	BD72
FAILSAFE	DB24
FALSE	ØØØØ
FBUFFR	Ø1ØØ
FBUFPT	ØØDF
FCB	ØØØ3
FCC	ØØØ3
FCERR	B7EØ
FCERR1	AF81
FCOMP	CØ12
FCOMPC	CØ39
FCOMPS	BFED
FCSIGN	BFEB
FDB	ØØØ4
FDCEND	C26Ø
FDECPT	ØØBE
FDIV	BF1D
FDIVF	BF18
FDIVT	BF1F
FEB	D974
FEND	D7Ø8
FEND1	D7ØB
FFDISP	A849
FFLOOP	AAD5
FFRESW	A9EA
FFRTS	AAF3
FFTOKN	ØØE5
FHALF	C24A
FIN	CØ8A
FIN1	CØAB
FINC	CØA6
FINDADD1	DØ11
FINDADD2	DØ16
FINDADDR	DØ1Ø
FINDIG	C1Ø4
FINDIV	CØEB
FINDP	CØDC
FINDSTR	DA2C
FINDSTR3	DA3Ø
FINDSTR9	DA39
FINE	CØE1
FINEC	CØCD
FINEC1	CØCA
FINEC2	CØD2
FINEDG	C121
FINGO	B61A
FINI	AC2Ø
FININ1	ACCD
FININL	B29Ø
FINLOG	C115
FINMUL	CØF5
FINNOW	B75B
FINPTR	B759
FINQNG	CØFD
FINREL	B5Ø1
FLAG	AA55
FLINRT	AD8D
FLOAT	BFF6
FLOATAD	E33C
FLOATAD5	E359
FLOATC	CØØ1
FLOATS	BFFD
FMULT	BE69
FMULTT	BE6B
FNDFOR	AADØ
FNDLIN	AD73
FNDLN1	AD75
FNDLOP	AD77
FNDOER	B927
FNDVAR	BA2E
FNLRTS	AD8E
FNTK	ØØB6
FNWAIT	BCFA
FONE	BE12
FOR	AE19
FORPNT	ØØCØ
FORPSH	B533
FORSIZ	ØØ1Ø
FORTK	ØØ81
FOUT	C153
FOUT1	C15E
FOUT11	C214
FOUT12	C22Ø
FOUT14	C22B
FOUT15	C233
FOUT16	C1CA
FOUT17	C244
FOUT19	C242
FOUT2	C1D1
FOUT2Ø	C246
FOUT3	C185
FOUT4	C17D
FOUT5	C19D
FOUT6	C1B3
FOUT7	C17B
FOUT8	C1CC
FOUT9	C195
FOUTBL	C24E
FOUTC	C156
FOUTCM	C2ØA
FOUTCP	CØ1C
FOUTYP	C1ED
FPRAM	Ø128
FPWR1	C28C
FPWRRT	C249
FPWRT	C269
FPWRT1	C271
FR4	C3ED
FRE	B8AC
FREFAC	BB25
FREINX	BB3E
FRERRC	ØØ26
FRERTS	BB5Ø
FRESPC	ØØA6
FRESTF	C465
FRESTR	BB22
FRETMP	BB27
FRETMS	BB43
FRETOP	ØØA4
FRETRT	BB4Ø

FRI	DA63	GETSTK	AB22	IDASCII	9C16
FRMEVL	B4A1	GETYES2	FACB	IDCHRPT	Ø527
FRMNUM	B493	GETYES4	FAD7	IF	BØ6Ø
FS2	DB35	GETYES6	FADA	IN2H	FDE9
FS25	DB4A	GETYESKY	FAC3	IN2HO	FDF8
FS27	DB55	GFDIV	C3DF	INCALPH9	FAE6
FS3	DB61	GFMULT	C311	INCALPHA	FADF
FS5	DB84	GIVABF	B8CB	INCFAC	BDDA
FS6	DB87	GIVDBL	B8C7	INCFRT	BDE4
FS65	DB8F	GMOVMF	C373	INCH	FE6B
FS7	DB9Ø	GNEGOP	C42D	INCHR	ACDØ
FSBATNØ	C74E	GOFLOT	B6AC	INCMORS2	EA7D
FSBATN1	C759	GOFUC	BC1C	INCMORS4	EA84
FSBCLKØ	C77A	GOMLDV	C2DA	INCMORS6	EA8B
FSBCLK1	C785	GONE	AEAD	INCMORSE	EA76
FSBDTAØ	C764	GONE2	AEB6	INCRTS	ACDC
FSBDTA1	C76F	GONE3	AEB4	INDEXBY2	D2FØ
FSCLR	DBA1	GONE4	AEBC	INDEXBY9	D2F5
FSCLR2	DBA8	GONE5	AEC8	INDEXBYT	D2EC
FSDISP	AA63	GOODCH	ACB8	INDICE	ØØD3
FSRCVD	Ø514	GOOMER	C5ØA	INDLOP	B784
FSREQN	9BFB	GOOVER	BEF5	INHEX	FDF9
FSRESW	A8E8	GOPTNW	BB6C	INIT	C483
FSSET	DBCØ	GORDY	AF6C	INIT1	DD3F
FSSET2	DBC6	GORTS	BØ18	INIT2	DD54
FSTIMER	Ø521	GOSGFL	BB53	INIT4	DD6Ø
FSTIMOUT	9C27	GOSUB	AFE6	INKCHR	Ø11C
FSTOKN	ØØ9E	GOSUTK	ØØ8C	INKEY$	D186
FSUB	BD1A	GOTARY	B7D1	INKEY$2	D1A1
FSUBT	BD1D	GOTO	BØØ2	INKEY$9	D1B7
FUNDSP	A8Ø9	GOTOTK	ØØ88	INLIN	AC7A
GARBA2	BA2C	GPOLYX	C3BF	INLINC	AC82
GARBAG	BA1E	GRBPAS	BAA6	INLINN	AC74
GARBFL	ØØ7D	GRBPNT	ØØCD	INLOOP	B388
GET2	D6F8	GRBRTS	BAA5	INLPN1	B861
GET27	D71Ø	GRBTOP	ØØCB	INLPNM	B84D
GETADR	BC64	GREATK	ØØC2	INPCOM	B12E
GETBKAD9	C9FB	GRT	ØØØØ	INPCON	B383
GETBKADR	C9D6	GTBYTC	BC1F	INPFLG	ØØ7F
GETBPT	AD26	HANDF	FFFF	INPPTR	ØØBA
GETBYT	BC22	HANG	AF29	INPRT	C13B
GETCMP	B694	HANG1Ø	AEF2	INPRTS	B429
GETDEF	B847	HANG12	AEFF	INPTK	ØØ84
GETFNM	B914	HANG14	AFØC	INPUT	B325
GETFOR	B442	HAVEND	C4F4	INPUTØ	B349
GETIOAD5	D3Ø2	HAVFOR	B45Ø	INPUT1	B353
GETIOAD8	D3Ø5	HFINIT	DD36	INSTAT	CE5Ø
GETIOADR	D2F6	HFRUNP	CFD6	INT	CØ68
GETNUM	BC58	HIGHDS	ØØC5	INTEGR	ØØ78
GETSFPTR	E2ED	HIGHTR	ØØC7	INTID2	B765
GETSPA	B9FE	HOUR	9FFB	INTIDX	B762

Symbol Table

INTIMER	Ø58E	KEYSCAN	E651	LDMDMBUF	DØ48
INTRTS	CØ89	KEYSCAN2	E66E	LDMSG	FA86
INTXT	AABC	KEYSCAN3	E671	LDMSGBUF	FB41
IOADTBL	D3Ø7	KEYSCAN8	E699	LDXAB	AB42
IOOFF	9ØØØ	KEYSCAN9	E6A8	LEFT$	BBB8
IOON	AØØØ	KEYSCN85	E6A3	LEN	BB51
IOXTMP	Ø24Ø	KEYTABLE	E816	LEN1	BB56
IRQ	DDDE	KLOOP	ACED	LENFPRAM	ØØ69
IRQRAM	ØØØØ	KLOOP1	ADØ8	LESSTK	ØØC4
ISARY	B77D	LASCII	D846	LET	B117
ISCLSØ	B617	LASTPØ	ØØFE	LETRTS	B176
ISCLS3	B619	LASTPT	ØØ83	LF	ØØØA
ISCNTC	AF12	LCD	A2ØØ	LINCHK	B28A
ISCRTS	AEE6	LCDBLAST	Ø3D8	LINEF	DA73
ISFUN	B5EA	LCDBUF	Ø393	LINES	DA9Ø
ISLETC	B717	LCDBUFIN	Ø38E	LINETK	ØØEA
ISLRTS	B71F	LCDBUFOT	Ø39Ø	LINGET	BØAØ
ISSEC	B6D9	LCDCURSR	F2CA	LINLIN	AC6D
ISVAR	B5DE	LCDINIT	E582	LINLIN1	AC7Ø
ISVRET	B5E1	LCDIRQ	E5B6	LINNUM	ØØBØ
JAN	D96C	LCDIRQ1	E5C7	LINPRT	C144
JMPTBL	CFD3	LCDIRQ3	E5D4	LIST	ADC5
JUL	D992	LCDIRQ5	E5E3	LIST4	ADDØ
JUN	D98D	LCDIRQ6	E5FØ	LISTENS	CFØ3
KBDDEV	Ø496	LCDIRQ7	E5F8	LISTENS2	CF1Ø
KBTIMER	Ø48E	LCDIRQ8	E5FB	LISTENS4	CF19
KEYASC2	E8ØE	LCDIRQ9	E5FE	LISTRT	AD72
KEYASCII	E8ØB	LCDMSG	EØEA	LOAD	C84Ø
KEYCOM	E6BØ	LCDMSG1	EØFC	LOAD1Ø	C8A9
KEYCOM1	E6B5	LCDOVRFL	Ø392	LOAD12	C8BA
KEYCOM2	E6EØ	LCDPUT	FA73	LOAD13	C8CF
KEYCOM29	E6F3	LCDSO	C54B	LOAD132	C8EØ
KEYCOM3	E716	LCDSTR	FA81	LOAD135	C8E8
KEYCOM31	E719	LCLTXT1	EB7E	LOAD137	C8FE
KEYCOM33	E71E	LCLTXT2	EBAØ	LOAD14	C92C
KEYCOM34	E731	LCLTXT3	EBC4	LOAD2	C871
KEYCOM35	E73B	LCOUNT	Ø244	LOAD4	C873
KEYCOM37	E73D	LD1ØØ	BF7D	LOAD6	C886
KEYCOM4	E74C	LDADBUF	DØ62	LOAD7	C891
KEYCOM5	E75Ø	LDBAS	DØ6B	LOADEEP	C97F
KEYCOM6	E75C	LDBAS9	DØ95	LOADEEPØ	C99Ø
KEYCOM7	E769	LDCALBUF	DØ96	LOADEEP1	C99F
KEYCOM8	E773	LDECAY	Ø497	LOADEEP2	C9A7
KEYCOM81	E78D	LDECAYCT	Ø498	LOADERR	C86E
KEYCOM82	E7A2	LDFONE	AE5A	LOADERR1	C86B
KEYCOM83	E7AA	LDLCDBUF	DØ3F	LOADS1	FDA7
KEYCOM84	E7AF	LDLCDCK	DØ1B	LOCAL	Ø447
KEYCOM85	E7B5	LDLCDCK9	DØ3E	LOCALØ	EBEØ
KEYCOM8A	E788	LDMBUFF	CF97	LOCAL2	EBF4
KEYCOM9	E7F6	LDMDMBU5	DØ58	LOCAL3	EC1Ø
KEYCOM95	E7FC	LDMDMBU7	DØ5F	LOCAL33	EC19

LOCAL35	EC2E
LOCAL5	EC68
LOCAL6	EC74
LOCALST	Ø1AD
LOFBUF	ØØFF
LOG	BE33
LOG1	BE3B
LOG2	BE2F
LOGCN2	BE16
LOGEB2	C2A7
LONGI	FFFF
LOOPDN	B481
LOPFDA	B7B8
LOPFND	B7Ø3
LOPPTA	B7FD
LOPREL	B4B8
LOWDS	ØØC9
LOWERF	D39Ø
LOWERS	D3A2
LOWETK	ØØE1
LOWTR	ØØCB
LPOPER	B4A8
LPTSW	ØØØØ
LSTATK	ØØB3
LSTNPØ	Ø127
LUKALL	BØ1Ø
MAIN	AB9D
MAIN1	ABBE
MAR	D97D
MAXLINE	Ø248
MAXLINEF	DAB3
MAXLINES	DADØ
MAXLINTK	ØØDC
MAXSITE	9BFA
MAY	D989
MBUFF	Ø1D9
MBYTCNT	Ø253
MCRLF	FC45
MDIVINIT	E35E
MDMBLAST	Ø38D
MDMBUF	Ø2C6
MDMBUFIN	Ø2C1
MDMBUFOT	Ø2C3
MDMCHR	Ø125
MDMCYCLS	Ø593
MDMOVRFL	Ø2C5
MDMSPAD	D28Ø
MDMSPAD6	D293
MDMSPAD7	D29D
MDMSPAD8	D2A2
MDMSPAD9	D2A2
MDMSPD	Ø23F
MDMSPDF	D2A3
MDMSPDS	D2A9
MDMSPDS3	D2C3
MDMSPDS5	D2C9
MDMSPDS9	D2DF
MDMSPDSH	D2D8
MDMSPDSL	D2BC
MDMSPDTK	ØØE9
MDMST	Ø124
MDMSTAT$	CCBC
MDMTRN	D2EØ
MDRET	BEE8
MEMLOP	C4EA
MEMOK	C5ØD
MEMSIZ	ØØA8
MERROR	FEØC
MESLEN	E1ØE
MESSAGE	Ø24D
MESSAGEF	CF84
MESSAGES	CF2C
MESSAGS5	CF63
MESSGTK	ØØED
METERF	D383
METERFØ	D3A9
METERF1	D3C6
METERF15	D3F7
METERF17	D4ØC
METERF18	D469
METERF2	D47Ø
METERF22	D47F
METERF23	D4A4
METERF24	D4B9
METERF25	D4CD
METERF27	D4D8
METERF3	D4DB
METERF5	D4E5
METERF6	D4FØ
METERF7	D4F3
METERFA	D4FB
METERFB	D51A
METERFC	D525
METERLS1	D51F
METERLS2	D52A
MFRCHAN	Ø251
MFRDEV	Ø252
MFRSITE	Ø25Ø
MID$	BBDD
MID2	BBEA
MINCHG	DF4Ø
MINCHG52	DF49
MINCHG53	DF54
MINCHG54	DF5B
MINCHG55	DF5E
MINCHG9	DF61
MINDET	Ø592
MINUTE	9FFA
MINUTK	ØØBC
MK48TØ2	FFFF
MLDVEX	BEE9
MLENGTH	Ø1D8
MLOOP	ABE5
MLTPL1	BE8E
MLTPL2	BE8F
MLTPL3	BEA6
MLTPLY	BE89
MMX	FDA7
MMXLF	FDCØ
MMXO	FDE6
MNXT	FDBC
MODEMTSØ	DC15
MODEMTS7	DC39
MODEMTS8	DC4Ø
MODEMTS9	DC5B
MODEMTSA	DC1E
MODEMTSB	DC2C
MODEMTST	DBFF
MODEMTX	E485
MODEMTX9	E497
MODTSTDØ	DC8B
MODTSTD1	DCAB
MODTSTD2	DCCA
MODTSTD3	DCEA
MODTSTD4	DDØA
MON	DA41
MONITOR	FBCE
MONOUT	FE75
MONRES	FF2B
MONTH	9FFE
MONVEC	FFFF
MORCOM	B2BA
MORLIN	BØA5
MORSETBL	EAC2
MOV1F	BFB2
MOV2F	BFAD
MOVAF	BFDA
MOVBYT9	FB64
MOVBYTE1	FB4E
MOVBYTES	FB4A
MOVFA	BFCD
MOVFA1	BFCF
MOVFM	BF99

Symbol Table

MOVFR	BF9Ø	NODATT	AD4Ø	OUTBEL	ACC1
MOVINS	BBØ7	NODEL	ABF1	OUTCH	D1C3
MOVLP	BB14	NOPOLL	D1AB	OUTCHØ	D1CF
MOVMF	BFB8	NORM1	BDA8	OUTCH1	FE76
MOVSTØ	D954	NORM2	BD9B	OUTCH2	D1D6
MOVSTR	BBØ9	NORM3	BD77	OUTCH3	D1DD
MOVSTRNG	D95Ø	NORMAL	BD76	OUTCH32	D1FE
MOVVF	BFB6	NOSEC	B6E3	OUTCH35	D2Ø1
MTOCHAN	Ø24E	NOSTREAM	Ø446	OUTCH5	D2Ø2
MTODEV	Ø24F	NOTABR	B2DD	OUTCH6	D2Ø5
MTOSITE	Ø24D	NOTCNC	ACAC	OUTCH7	D21D
MUL1Ø	BEF8	NOTCNU	AC9Ø	OUTCH75	D22B
MUL1ØR	BFØC	NOTDIM	B8ØB	OUTCH8	D22E
MULDIV	BED2	NOTEVL	B72C	OUTDO	B2F9
MULDV1	BEEE	NOTFDD	B7E5	OUTHAL	FE23
MULDV2	BEFØ	NOTFNS	B72Ø	OUTHEX	FEØE
MULLN2	BE66	NOTIT	B7ØF	OUTHEXS	FBC3
MULSHF	BDEA	NOTOL	AE2B	OUTHL	FE14
MULT1Ø	E313	NOTQTI	B361	OUTHR	FE18
MULTK	ØØBD	NOTSTR	B6EF	OUTM	FE2Ø
MULTRT	BEB5	NOTSTT	AECF	OUTQST	B2F7
MUSTCR	AD1Ø	NOTTK	ØØB9	OUTSPC	B2F4
MXG	FDB8	NOV	D9BØ	OVERFLOW	Ø2BC
MXHDST	98ØC	NOWGET	B3B8	OVERR	BDE5
N.Ø999	C12F	NOWLIN	B4Ø5	PACK	DB11
N.9999	C133	NTHIS	AD56	PACK1	DB21
N.MIL	C137	NTHIS1	AD59	PACK9	DB23
N32768	B75E	NUMINS	B3C5	PARCHK	B5BA
NEGFAC	BDCE	NUMLEV	ØØ1D	PCRLF	DC7A
NEGFCH	BDD1	NUMTMP	ØØØ3	PDATA	DC7D
NEGHLF	BE2B	NXTCMP	B68D	PDATA9	DC87
NEGOP	C29F	NXTCON	AE79	PDATACR	DC78
NEGRTS	C2A6	NZLINK	ØØØ5	PEEK	BC74
NEWCHR	B24B	OCT	D9A8	PEEK5	BC8A
NEWSGO	B47E	OK	FEØ9	PFORMT	B1C2
NEWSTT	AE7C	OKGOTO	BØ6F	PFORMT1	B1CF
NEXT	B43B	OLDAD	FFFF	PFORMT1Ø	B213
NEXTLI	FE86	OLDLIN	ØØAE	PFORMT11	B216
NINCH	FE27	OLDTXT	ØØB2	PFORMT14	B222
NINCH2	FE2A	OMERR	AB49	PFORMT16	B224
NINCH5	FE2A	OMERR1	AFD9	PFORMT18	B22A
NINCH55	FE32	ONEON	AE6D	PFORMT2	B1DA
NINCH6	FE43	ONGLOP	BØ8C	PFORMT4	B1EB
NINCH8	FE56	ONGLP1	BØ95	PFORMT5	B1F4
NINCH9	FE58	ONGOTO	BØ8Ø	PFORMT6	B1FC
NINCHT	FE5C	ONGRTS	BØ9F	PFORMT8	B2Ø8
NINCHT9	FE6A	OPMASK	ØØC4	PFORMT9Ø	B22E
NMARY1	B7C8	OPPTR	ØØC2	PFORMT92	B23Ø
NOBREAK	AEE7	OPTAB	A861	PFORMT95	B234
NOBRKFLG	Ø126	ORFIN	B647	PFORMT99	B237
NOCNC	ADDC	OROP	B627	PI2	C3E5

PLOOP	ADFØ	QCHNUM	B548	RESPUL	AD17
PLUSTK	ØØBB	QFOUND	BØ13	RESRCH	AEØ5
POKE	BCAB	QINLIN	B37D	RESTOR	AEE1
POKE5	BCE7	QINT	CØ3E	RETRIES	Ø48B
POKER	ØØBØ	QINT1	CØ5C	RETRIESF	D53C
POLCAT	DEØ6	QINTGO	B77A	RETRIESL	Ø48C
POLCAT7	DE1F	QINTRT	CØ5B	RETRIESS	D53Ø
POLCAT8	DE2Ø	QISHFT	CØ4E	RETRIETK	ØØEC
POLCAT9	DE23	QMOVE	BB18	RETU1	BØ32
POLCATA	DE25	QOP	B541	RETURN	BØ19
POLLRX	E44E	QOPRTS	B57Ø	RIGHT$	BBD6
POLLRX8	E45F	QPLUS	CØA2	RL$RET	BB6E
POLLRX9	E476	QPREC	B4ED	RLBYTE	Ø453
POLLTX	E479	QPREC1	B514	RLEFT	BBBB
POLLTX9	E483	QRND	BDC7	RLEFT1	BBC2
POLY	C314	QSHFT	BF57	RMB	ØØØ2
POLY1	C316	QVARIA	B13E	RMONTH	ØØØ7
POLY2	C322	RADD.C	C33C	RMUL.C	C338
POLYPT	ØØDF	RAISEF	D389	RND	C34Ø
POLYX	C3Ø5	RAISES	D39B	RND1	C35A
POS	B8D7	RAISTK	ØØE2	RNDIT	BDB4
POSINT	B768	RAM	Ø596	RNDRTS	BDCD
PREAM	BCØØ	RAMCOD	C5ED	RNDSHF	BDB9
PREAMR	ØØ98	RAMGOT	C5F5	RNDX	Ø1ØD
PRGBOT	CFD3	RAMMCW	9ØAE	ROLSHF	BEØ9
PRINT	B23D	RAMRTS	C6Ø4	ROM	E857
PRINT1	B24Ø	RAMTEST	ØØØØ	ROMADR	A8ØØ
PRINTC	B25Ø	RANDRT	C337	ROUND	BDB2
PRINTK	ØØ94	RASCII	D84A	RSTSTK	B69F
PRIT3	AEØF	RASCII9	D854	RTRFPRA5	EAF9
PRIT4	ADEE	RDATE	ØØ2Ø	RTRFPRAM	EAF4
PRNTSEL	Ø24C	RDYJSR	Ø112	RUN	AFDC
PRTRTS	B324	READ	B38Ø	RUNC	AD9D
PRTSEL	D3ØF	READY	AB8F	RUNC2	AFFA
PRTSELØ	D326	REALIO	FFFF	RXBUF	Ø49E
PRTSEL1	D32B	REARTS	AB3A	RXBUFPT	Ø49C
PRTSEL2	D32E	REASON	AB28	RXBYTCNT	Ø5ØD
PRTSEL3	D32F	REDDY	AAC1	RXCKSUM	Ø5ØC
PRTSEL4	D33Ø	REM	BØ73	RXDBUGTX	E44A
PRTSTR	Ø1C2	REMER	BØ52	RXDEBUG9	E473
PTDORL	B511	REMN	BØ47	RXIRQ	E383
PTRGET	B6BB	REMRTS	BØ43	RXIRQ1Ø	E3AC
PTRGT1	B6BF	REMTK	ØØ8E	RXIRQ11	E3BC
PTRGT2	B6C1	RESCR1	AEØ8	RXIRQ12	E3D1
PTRGT3	B6CD	RESER	AD16	RXIRQ14	E3F9
PULSTK	B551	RESET	FE8E	RXIRQ2	E387
PUSHF	B52D	RESFIN	AEE4	RXIRQ2Ø	E3FE
PUSHF1	B529	RESHO	ØØ95	RXIRQ22	E4Ø3
PUTNEW	B9EØ	RESLO	ØØ97	RXIRQ24	E413
PUTNW1	B9EC	RESLST	A876	RXIRQ3Ø	E436
Q	ØØ24	RESMO	ØØ96	RXIRQ4	E393

Symbol Table

RXIRQ6	E398
RXIRQ7	E39C
RXIRQ8	E39D
RXIRQ9	E3B5
RXIRQPTR	Ø49A
RXIRQRES	E499
RXTIMER	Ø5ØE
RYEAR	ØØ89
SAT	DA6A
SAVE	C9FF
SAVE1Ø	CA5F
SAVE12	CA6B
SAVE2	CA27
SAVE3	CA2D
SAVE4	CA2F
SAVE6	CA42
SAVE7	CA4D
SAVE8	CA5Ø
SAVEERR	CA2A
SAVQUO	BF47
SBATN	ØØØ1
SBCLKIN	ØØØ4
SBCLKOUT	ØØØ2
SBCLOSE	ØØEØ
SBCMD	C824
SBCMD2	C82B
SBCMND	CC4Ø
SBCMND2	CC67
SBCMND3	CC6A
SBCMND4	CC6C
SBCMND6	CC7F
SBCMND7	CC8A
SBCMND8	CC9D
SBCMNDA	CCA8
SBDIR	CBCC
SBDIR1Ø	CC24
SBDIR12	CC34
SBDIR14	CC3C
SBDIR2	CBD8
SBDIR4	CBDA
SBDIR6	CBFF
SBDIR7	CCØ8
SBDIR8	CC16
SBDTAIN	ØØ1Ø
SBDTAOUT	ØØØ8
SBERRN	ØØ24
SBERROR	C7Ø3
SBERRORA	C6E5
SBERRORB	C6E9
SBERRORC	C6ED
SBERRORD	C6F1

SBERRORE	C6F5
SBERRORF	C6F9
SBERRORG	C6FD
SBERRORH	C7Ø1
SBINIT	C66E
SBLDX1	Ø19C
SBLDX2	Ø19E
SBMLA	ØØ2Ø
SBMSA	ØØFØ
SBMTA	ØØ4Ø
SBPA	Ø196
SBRXBY8	C815
SBRXBYT2	C7F4
SBRXBYT4	C8ØØ
SBRXBYT5	C8Ø3
SBRXBYTE	C7EB
SBTEMPX2	Ø194
SBTKEOI	C6A4
SBTKEOI1	C6B1
SBTKEOI2	C6BF
SBTKEOI4	C6CA
SBTKEOI6	C6D6
SBTKHS	C674
SBTKHS1	C681
SBTKHS2	C691
SBTKHS3	C697
SBTURNA2	C7DB
SBTURNAR	C7CE
SBTXBYT2	C726
SBTXBYT4	C72D
SBTXBYT5	C73B
SBTXBYT8	C748
SBTXBYTE	C71F
SBTXCHAR	C715
SBTXCHAS	C718
SBTXEOI	C71D
SBUNL	C834
SBUNL2	C837
SBUNT	C83B
SBXTEMP	Ø192
SCAN	D5EC
SCAN2	D6Ø7
SCAN9	D62B
SCANCAN	D633
SCANCHK	D62C
SCANMAX	Ø577
SCANMIN	Ø576
SCANSITE	Ø575
SCRATH	AD91
SCRTCH	AD93
SECCHGØØ	DEE2

SECCHGØ1	DEEA
SECCHGØ2	DEF3
SECCHGØ3	DEF3
SECCHGØ4	DEFC
SECCHGØ5	DFØ5
SECCHGØ6	DFØE
SECCHGØ7	DF17
SECCHGØ8	DF24
SECCHGØ9	DF27
SECCHG1Ø	DF35
SECCHG11	DF3C
SECCHNG	DEDA
SECDET	Ø59Ø
SECOND	9FF9
SELAD	DØD1
SELECT	DØE7
SELECT8	DØF2
SELLCD	DØA5
SELMDM	DØBB
SEP	D99E
SERRATE1	9BF8
SERRATE2	9BF9
SETUP	Ø48F
SFØ	98ØD
SFPTR	Ø44F
SGN	BFF4
SGNFLG	ØØD6
SHFARG	BF5A
SHFTR2	BDED
SHFTR3	BEØ7
SHFTRT	BE11
SHIFTR	BDFD
SIGN	BFE7
SIGNRT	BFF3
SIN	C37C
SIN1	C3AC
SIN2	C3AF
SIN3	C3BC
SINCON	C3F1
SINFIX	A81F
SITECHAN	Ø492
SITECHG	Ø49Ø
SITECTR	Ø443
SITEDEL	9BF6
SITENUM	9BF5
SITEOK	E378
SIZEOK	BAEE
SNERR	B5D1
SNERR1	AE91
SNERR2	BØ2F
SNERR3	BØ8A

SNERR4	B317	STATOUT	CCDA	STRLT2	B9AB
SNERR5	B526	STATPNT	Ø1AE	STRLT3	B9A7
SNGFLT	B8DC	STATREP	CE4C	STRLTI	B9A5
SOURCE	Ø52C	STATREQ	DBDE	STRNG	AD51
SPCTK	ØØB7	STATRMES	DBF8	STRNG1	ØØDD
SQR	C26Ø	STATRQ9	DBF7	STRNG2	ØØDF
SQRØ.5	BE23	STATRQST	DBEØ	STROU2	C15Ø
SQR2.Ø	BE27	STATUS	D397	STROUT	B2E3
ST2TXT	BC53	STATUSS1	CE7E	STRPR2	B2EA
START	A8ØØ	STATUSS2	CE81	STRPRT	B2E6
STASGN	B681	STATUSS6	CEA8	STRPXB	B2E9
STASHV8	C964	STATUSST	CE6E	STRSIZ	ØØØ4
STASHVAR	C92F	STATUSTK	ØØE3	STRSPA	B99D
STAT	FBCE	STCLTX1	F2D4	STRSPC	ØØ64
STATCA1D	F37Ø	STCLTX2	F2EC	STRTXT	B592
STATCA1F	F375	STCLTX3	F3Ø8	STRXADR	Ø1AØ
STATCA3Q	F39E	STCLTX4	F317	STRXEND	Ø1AD
STATCAL	F342	STCLTX5	F326	STRXRAM	Ø1A1
STATCAL1	F35D	STEPTK	ØØBA	STTXCT	Ø1BE
STATCAL2	F38A	STFEND	C483	STTXEND	Ø1BE
STATCAL3	F396	STFLOP	AED2	STTXRAM	Ø1B2
STATCAL4	F3AF	STFTAB	C456	STUFFH	AD28
STATCAL6	F3BE	STKINI	ADAF	STVPNT	AAEF
STATCAL8	F3D3	STKTMP	ØØ98	STXBUF	C2Ø6
STATCALA	F3E5	STKTOP	ØØA2	STXFOR	B445
STATCALC	F3E8	STMDSP	AA27	SUB16BIT	E3Ø9
STATCALE	F3F1	STOLOG	B64B	SUBFLG	ØØ7E
STATCHNG	Ø1BF	STOLOP	AC1Ø	SUN	DA3A
STATDP	Ø1BØ	STOP	AF44	SVAR	BA46
STATFN	FFFF	STORDO	BDØ8	SVARS	BA44
STATIN	CCEA	STOUTBY8	CE49	SWAP	BØCA
STATINIT	CCC6	STOUTBYT	CE35	SWAP5	BØE6
STATINV	98ØØ	STOUTPLS	CE27	SWAP6	BØEB
STATIR1Ø	CD4C	STPEND	AF4F	SWAP7	BØF7
STATIR12	CD74	STR$	B98B	SWAP8	B1Ø3
STATIR14	CD96	STR1	AD49	SWAP9	B1ØF
STATIR2Ø	CDB4	STRAD2	B99F	SYNCHR	B5C7
STATIR22	CDC3	STRCMP	B669	SYSSE1ØØ	F8D1
STATIR24	CDCB	STRDN2	B3CB	SYSSE1Ø2	F8FØ
STATIR25	CDD9	STRDON	B248	SYSSE1Ø4	F9Ø9
STATIR26	CDF5	STREAMER	E5Ø3	SYSSE1Ø6	F94F
STATIR28	CDFD	STREND	ØØAØ	SYSSE1Ø8	F957
STATIR3Ø	CEØB	STRFI1	B9C5	SYSSE11Ø	F982
STATIR32	CE1Ø	STRFI2	B9D9	SYSSE112	F99B
STATIR34	CE1E	STRFIN	B9CØ	SYSSE913	F81F
STATIR4Ø	CE2Ø	STRFPRA5	EAEB	SYSSE915	F82E
STATIRQ	CCFA	STRFPRAM	EAE6	SYSSE945	F899
STATIRQ2	CCFF	STRFRE	BA13	SYSSE947	F89C
STATIRQ4	CD24	STRGET	B9B2	SYSSET	F418
STATIRQ6	CD3D	STRINI	B99B	SYSSET1Ø	F497
STATIRQ9	CD4B	STRLIT	B9A4	SYSSET12	F4AE

Symbol Table

SYSSET14	F4BF	SYSTXT11	F1AC	TIMER7	DEA3
SYSSET16	F4CE	SYSTXT13	F1CE	TIMER8	DEAC
SYSSET2	F43Ø	SYSTXT15	F1FØ	TIMER9	DEB5
SYSSET2Ø	F4F9	SYSTXT17	F21Ø	TIMERF	D719
SYSSET3	F438	SYSTXT19	F22F	TIMERF2	D72C
SYSSET3Ø	F513	SYSTXT21	F249	TIMERINT	DE39
SYSSET32	F525	SYSTXT23	F26A	TIMERIRQ	DE4A
SYSSET34	F52C	SYSTXT25	F279	TIMERRTS	DED9
SYSSET36	F541	SYSTXT26	F289	TIMERS	D658
SYSSET38	F555	SYSTXT28	F2A8	TIMERS2	D668
SYSSET4	F44F	SYSTXT3	F13C	TIMERTBL	Ø578
SYSSET4Ø	F55E	SYSTXT3Ø	F9E4	TIMES	D68C
SYSSET42	F56F	SYSTXT32	F9FB	TIMES2	D6D3
SYSSET44	F57F	SYSTXT34	FA15	TIMES3	D6D9
SYSSET46	F58B	SYSTXT36	FA31	TIMES5	D6E1
SYSSET48	F5A2	SYSTXT37	FA4B	TIMETK	ØØDD
SYSSET5Ø	F5B2	SYSTXT5	F158	TMFST1	D7CE
SYSSET52	F5B9	SYSTXT7	F171	TMFST15	D7DC
SYSSET6	F46Ø	SYSTXT9	F18C	TMFST2	D819
SYSSET6Ø	F5D3	TABER	B2C1	TMFST3	D838
SYSSET61	F5DC	TABTK	ØØB4	TMFST5	D833
SYSSET62	F5F9	TAN	C3C2	TMFSTR	D7A5
SYSSET65	F61Ø	TANFIX	A821	TMFSTR1	D7A8
SYSSET67	F61C	TANSGN	ØØ8Ø	TMSTF	D7B5
SYSSET68	F625	TEMP	Ø456	TOP	FC48
SYSSET69	F642	TEMPF1	ØØC5	TOTK	ØØB5
SYSSET71	F659	TEMPF2	ØØC9	TRACE	C64A
SYSSET73	F6A2	TEMPF3	ØØCD	TRACE9	C66D
SYSSET74	F6E1	TEMPPT	ØØ81	TRAM	Ø123
SYSSET75	F6FD	TEMPST	ØØ85	TRMNO1	B31A
SYSSET76	F73B	TEMPX1	ØØ91	TRMNOK	B3ØF
SYSSET77	F74A	TEMPX2	ØØ93	TRMOK	B3D7
SYSSET78	F768	TEMPX3	Ø11F	TROFF	C646
SYSSET79	F781	TEMPX4	Ø121	TRON	C634
SYSSET7A	F78B	TEN.C	BFØD	TRUE	FFFF
SYSSET8	F46F	TENEXP	ØØCB	TRYAG2	BAØ1
SYSSET8Ø	F7AØ	THENTK	ØØB8	TRYAGN	B2FC
SYSSET85	F7A9	THU	DA5A	TSECCHG	DF3D
SYSSET87	F7C7	TIMEF	D746	TSECDET	Ø591
SYSSET88	F7EØ	TIMEFN	D75C	TSTOP	B4B5
SYSSET89	F7EA	TIMEFN2	D77Ø	TUE	DA48
SYSSET9Ø	F7FF	TIMEFN9	D798	TVAR	BA3C
SYSSET91	F8ØB	TIMER1	DE62	TWOPI	C3E9
SYSSET92	F851	TIMER1Ø	DECD	TXCKSUM	Ø511
SYSSET93	F882	TIMER11	DED9	TXCNT	Ø513
SYSSET94	F888	TIMER12	DED9	TXIRQ	E4A2
SYSSET95	F8A3	TIMER2	DE86	TXIRQØ	E4AA
SYSSET96	F8B4	TIMER3	DE87	TXIRQ1	E4B5
SYSSET97	F8C5	TIMER4	DE8A	TXIRQ1Ø	E4F5
SYSSET99	F9D5	TIMER5	DE92	TXIRQ11	E531
SYSTXT1	F11A	TIMER6	DE9A	TXIRQ12	E53D

TXIRQ13	E54Ø
TXIRQ14	E548
TXIRQ16	E55Ø
TXIRQ165	E557
TXIRQ17	E558
TXIRQ175	E566
TXIRQ18	E56B
TXIRQ2	E4B6
TXIRQ3	E4B6
TXIRQ4	E4BC
TXIRQ6	E4CC
TXIRQ8	E4E5
TXIRQ9	E4EF
TXIRQ99	E57F
TXIRQPTR	Ø5ØF
TXTBEG	ØØB4
TXTIMER	Ø512
TXTPTR	ØØEA
TXTTAB	ØØ9A
UMLCNT	B8A3
UMLRTS	B8A8
UMULT	B883
UMULTC	B891
UNPRTS	B572
UNPSTK	B54F
USEGET	B1BA
USEGET9	B1C1
USERR	BØ2A
USING	B17A
USING1	B19Ø
USING2	B19B
USING3	B19E
USING5	B1A2
USING7	B1AD
USINGTK	ØØE7
USRJMP	Ø119
USRLOC	Ø11A
VAL	BC31
VAL1	BC39
VALRTS	BC57
VALTYP	ØØ7C
VARDEL	DD25
VARDELØ	DD26
VARDEL1	DD29
VAREND	B4ØE
VARNAM	ØØBC
VARPNT	ØØBE
VARSIZ	Ø198
VARTAB	ØØ9C
VARTXT	ØØBØ
VIA488	A1ØØ
WAITER	BDØC
WARMS	A8Ø3
WATCHDG2	C61D
WATCHDG4	C627
WATCHDOG	C6ØB
WED	DA5Ø
WORDS	C56D
WSBCLKØ	C79Ø
WSBCLK1	C7A1
WSBCLK4	C793
WSBCLK5	C7A4
WSBDTAØ	C7BØ
WSBDTA1	C7BF
WSBDTA4	C7B3
WSBDTA5	C7C2
WSBEXIT	C79D
XFERMESS	CFE6
XFERRTS	CFD5
XFERTBL	CFFC
XFROMME	CFEØ
XHI	Ø23D
XLO	Ø23E
XTEMP	Ø454
YEAR	9FFF
ZERITA	B83Ø
ZERO	B799
ZEROF1	BD91
ZEROFC	BD9Ø
ZEROML	BD93
ZERRTS	BD95

Reading CAD Schematics

The DRC190 was designed using a design automation system from Dasoft Corporation (Berkeley, CA). This system handles schematic capture, net list generation, schematic plotting, circuit board routing, and circuit board artwork plotting. The schematic plotting uses the newer international symbols for components, which many people may not be familiar with. This section describes the interpretation of the schematics.

The schematic for a board is normally broken into several pages to keep the plots down to a reasonable size. We've tried to put break the schematics into logical blocks for each page. The theory of operation section of the manual describes the operation of the board page by page of the schematic.

A component is shown on the schematic as a rectangle, perhaps with a notch in it. If there is no notch, the device has only inputs and outputs. The inputs are on the left, and the outputs are on the right. Many components have bi-directional lines, so we've tried to assign them in some logical manner (ie, processor lines on the left, peripheral lines on the right). If there is a notch in the component block, a three letter descriptor of the device (ie, CMP for comparator) is listed inside the block at the notch. Lines on the left above the notch are "control inputs", while lines on the right above the notch are "control outputs". Control inputs might be interrupt inputs on the processor, or interrupt outputs on a peripheral device.

For each input and output is a descriptor of the function of that input or output inside the component rectangle. These might be DØ, D1, D2, etc for data lines. Immediately outside the rectangle is the corresponding pin number on the device. If the input or output is active low, a small circle or bubble appears where the signal line joins the component rectangle. Note that the signal designation inside the chip tells what happens when the line is true. For example, a pin labeled CE (chip enable) enables the chip when the line is high if there is no bubble outside the symbol. It enables the chip when the line is low if there is a bubble outside the symbol.

As we move away from the symbol, the next item found on the net is the pin number on this chip. Immediately outside the pin number is the "net name".

Net names are assigned to all signals on the board. Whenever it is desired to have a particular signal show up on a specified pin of a component, the corresponding net name is associated with that pin. For example, the processor and all peripheral chips and all memory chips on the processor board need the eight data lines (DØ through D7). You'll see that these net names have been listed in the appropriate places on the schematics. The schematic router and the printed ciruit board router connect together all pins that have the same net name. The connections on the schematic are not critical, as someone evaluating the schematic does not generally follow the lines around the drawing, but instead looks for the same net name showing up on the different devices. For that reason, the schematic router sometimes does not route a line, due to lack of space. The printed circuit router, of course, must always route a line. We've tried to choose standard net names, so that if you are having trouble with a particular chip, you can see that data shows up on these pins, address on these pins, IRQ on this pin, etc., without having to follow lines all over a large drawing. In addition, nets that leave a particular page of a schematic are listed on the sides of the schematic (if room is available), followed by a list of the page numbers that the net also appears on.

Finally, in accordance with the STD bus standard, active low net names end with an asterisk (*). The plotter prints the asterisk as a small block following the net name. For example, WE* is the active low Write Enable, OE* is the active low Output Enable, and R-W* is high for a Read, and low for a Write.

STD Bus Theory of Operation

The DRC19Ø is built around the STD bus, a standard 8 bit microcomputer bus. The STD standard calls for boards to be 4.5 x 6.5 inches. Some boards in the DRC19Ø are expaned to 4.5 x 9.6 inches to allow for additional circuitry. Most standard STD bus boards can be used in the DRC19Ø. Note, however, that the 68ØØ STD does not provide for memory refresh. Dynamic RAM boards with on-board refresh (such as those available from Systek) can be used in the DRC19Ø. Those dynamic memory boards that expect the processor to do memory refresh cannot be used. Most I/O boards can be used.

The pin out of the STD bus as used in the DRC19Ø is listed below.

Component Side		Solder Side	
1	+5 VDC Power	2	+5 VDC Power
3	Ground	4	Ground
5	No Connection	6	No Connection
7	D3	8	D7 - Most Significant Data Bit
9	D2	1Ø	D6
11	D1	12	D5
13	DØ - Least Significant Data Bit	14	D4
15	A7	16	A15 - Most Significant Address Bit
17	A6	18	A14
19	A5	2Ø	A13
21	A4	22	A12
23	A3	24	A11
25	A2	26	A1Ø
27	A1	28	A9
29	AØ - Least Significant Address Bit	3Ø	A8
31	WR* - Active Low Write Strobe	32	RD* - Active Low Read Strobe
33	IORQ* - Active Low I/O Request	34	MEMRQ* - Active Low Memory Request
35	IOEXP - Grounded in DRC19Ø	36	MEMEXP - Grounded in DRC19Ø
37	No Connection	38	Phase 2* - Active low system clock
39	No Connection	4Ø	R/W* - Hi Read, Lo Write
41	No Connection	42	No Connection
43	No Connection	44	IRQ* - Active Low Interrupt Request
45	No Connection	46	NMI* - Non-Maskable Interrupt
47	SYSRES* - Active Low Reset	48	PBRES* - Grounded by reset button
49	No Connection	5Ø	No Connection
51	No Connection	52	No Connection
53	Ground	54	Ground
55	+12 VDC	56	-12 VDC

Many of these pins are self explanatory. For those less familiar with microcomputer operation, the following descriptions are provided.

The data bus (D7-DØ) allow bi-directional communications between the processor board and other boards in the system. When the processor board is sending data to another board, the R/W* line is low, indicating the processor is trying to write data. When the processor wants to input data, it leaves the R/W* line high, indicating it wants to read. The R/W* line is driven by the processor whether it wants to access devices on the processor board or off the board, or whether the device being accessed is a memory or Input/Output device. Data is passed on D7-DØ, 8 bits at a time, with D7 being the most significant bit.

A15-AØ form the address bus. The processor drives this bus. The processor sets up the address that it wishes to communicate with (perhaps reading an instruction or data, or writing data to either a memory location or an I/O device), sets IORQ* low if the address corresponds to an I/O device, or sets MEMRQ* low if the address corresponds to a memory location, sets up the data, if this is a processor write, then strobes RD* or WR* for a read or write strobe. The processor latches up the read data on the trailing positive edge of the RD*. The external device latches up the data on the trailing edge of the WR* strobe.

The 68BØ2 processor uses memory mapped I/O. This allows the same STAA (store accumulator A) instruction to store the contents of an accumulator in a memory location, or in one of the registers of an I/O device. Other processors (Intel and Zilog series) use separate OUT instructions.

The memory map for the DRC19Ø with standard 32K RAM on the processor board is shown below:

Address Range	Device
ØØØØ - 7FFF	U21, 32 Kbyte RAM chip on processor board
8ØØØ - 8FFF	RAM on optional RAM board
9ØØØ - 9ØØF	Analog to digital converter board Ø
9Ø1Ø - 9Ø1F	Analog to digital converter board 1
9Ø2Ø - 9Ø2F	Analog to digital converter board 2
9Ø3Ø - 9Ø3F	Analog to digital converter board 3
9Ø4Ø - 9Ø4F	Analog to digital converter board 4
9Ø5Ø - 9Ø5F	Analog to digital converter board 5
9Ø6Ø - 9Ø6F	Analog to digital converter board 6
9Ø7Ø - 9Ø7F	Analog to digital converter board 7
9Ø8Ø - 9Ø8F	Analog to digital converter board 8
9Ø9Ø - 9Ø9F	Analog to digital converter board 9
9ØCØ - 9ØCØ	Optional RAM board bank select register
9ØEØ - 9ØEF	Direct Connect Modem Board
9ØFØ - 9ØFF	Disk/Status I/O board
98ØØ - 9fFF	U2Ø 2 Kbyte EEPROM on processor board
AØØØ - AØØF	U6 DUART on processor board
A1ØØ - A1ØF	U16 VIA for IEEE488 and CW ID on processor board
A2ØØ - A2Ø1	Front panel LCD
A8ØØ - FFFF	U19 EPROM on processor board

When a bank selected RAM board is used, the RAM on the processor board, which is not bank selected, and is therefore common to all banks, is reduced to 8 Kbytes. The memory map for such a system is shown below.

Address Range	Device
0000 - 1FFF	U21, 8 Kbyte RAM chip on processor board (common)
2000 - 8FFF	RAM on optional RAM board (bank selected)
9000 - 900F	Analog to digital converter board 0
9010 - 901F	Analog to digital converter board 1
9020 - 902F	Analog to digital converter board 2
9030 - 903F	Analog to digital converter board 3
9040 - 904F	Analog to digital converter board 4
9050 - 905F	Analog to digital converter board 5
9060 - 906F	Analog to digital converter board 6
9070 - 907F	Analog to digital converter board 7
9080 - 908F	Analog to digital converter board 8
9090 - 909F	Analog to digital converter board 9
90C0 - 90C0	Optional RAM board bank select register
90E0 - 90EF	Direct Connect Modem Board
90F0 - 90FF	Disk/Status I/O board
9800 - 9fFF	U20 2 Kbyte EEPROM on processor board
A000 - A00F	U6 DUART on processor board
A100 - A10F	U16 VIA for IEEE488 and CW ID on processor board
A200 - A201	Front panel LCD
A800 - FFFF	U19 EPROM on processor board

Theory of Operation 1441A Analog To Digital Converter Board

The analog to digital converter board selects external samples through reed relays and presents the selected sample to the analog to digital converter. The sample is measured and sent to the processor through the STD bus. In addition, processor instructions can drive the control (Raise, Lower, Fail Safe, and Channel Select) outputs. The description of the circuit will be broken down by schematic page number for simplicity.

Page 1: STD Bus Interface, VIA

UØ2 (the Versatile Interface Adapter) is interfaced to the STD bus in somewhat standard fashion. UØ3 compares the address on the bus with that set up by the jumpers on PØ1. If these addresses agree and IORQ* (Input Output Request) is driven low by the memory map PROM on the processor board, BOARD-SEL* goes low. This enables the VIA UØ2 and the data transceiver UØ1. The direction of data transmission is determined by the STD-R/W* line on pin 1 of UØ1. This line is high when the processor board is attempting to read from an external device. If BOARD-SEL* is low, UØ1 will drive the STD bus with data provided by UØ2. If the STD-R/W* line is low, the processor is attempting to write to an external device. If BOARD-SEL* is low, UØ1 will take data from the bus and present it to UØ2.

In a similar manner, UØ2 watches BOARD-SEL* and STD-R/W* to determine if and in which direction data is to be sent. In addition, UØ2 uses PHASE 2 (STD-P2* inverted by UØ4A) to synchronize the data transfer with the processor. It also uses STD-AØ through STD-A3 (the four least significant processor address lines) to select which of the 16 registers in UØ2 is to be addressed.

UØ2 is programmed by the DRC software to generate an interrupt each time the A/D converter finishes a conversion. The IRQ* output of UØ2 goes low when UØ2 requests an interrupt. This is double inverted (giving sufficient drive to drive the bus) by UØ4B and UØ4C. Since UØ4 is an open-collector device, sections of RØ3 are used as pull-up resistors where necessary. A pull-up is not required on the STD-IRQ* since it is pulled up on the processor board.

The outputs of UØ2 drive the remainder of the circuitry. CHANØ through CHAN9 are programmed high by the DRC program as necessary to select one of the ten channels of metering and control. DELAY-RES* floats high on power-up since the peripheral lines of UØ2 are set to input on reset. Once all initialization of the VIA is complete, DELAY-RES* is programmed low, enabling the 5 volt supply to the remainder of the board. This prevents all the reed relays and all Raise, Lower, and Channel Select output from being enabled on system reset.

FAILSAFE1 and FAILSAFE2 are driven by the DRC program as appropriate. If all failsafe requirements are being met, FAILSAFE1 is high, turning on the FAILSAFE1* output. If a failsafe requirment is not being met (a required site is not responding), the FAILSAFE1 output goes low, turning off the FAILSAFE1* output. FAILSAFE2 is normally low, causing FAILSAFE2* to be high. If the operator uses the "Lock out control from elsewhere" feature of the calibrate sequence, FAILSAFE2* goes low as long as control is locked out. This may be used to drive a "local" alarm.

RAISE and LOWER are driven by the DRC software as needed to generate raise and lower control signals. These signals are NANDed with the channel select outputs to provide active low Raise and Lower outputs for each metering channel.

BUSY, BSY.CLK and POLARITY are inputs to UØ2. BUSY indicates that an A/D conversion is in process. UØ2 is programmed to generate an interrupt on the trailing negative edge of BUSY. The DRC program then takes the A/D data and sets up for the next conversion.

POLARITY indicates the polarity of the sample that has just been converted. If POLARITY is high, the reading is positive.

The A/D converter determines the digital conversion of the sample voltage by integrating the sample signal for 1Ø,ØØØ counts of a 125 KHz clock. It then counts clock pulses as it "de-integrates" a reference voltage until the integrator output crosses zero volts. During these two periods, the BUSY output of the A/D is high. If we count the clock pulses during the time that BUSY is high, and subtract 1Ø,ØØ1 (an extra clock pulse sneaks in), we get the result of the A/D conversion.

The A/D board utilizes the counter in the VIA to count the pulses on PB6 (which is BUSY anded with CLOCK, hence BSY.CLK). On detecting the end of a conversion (by the interrupt generated by BUSY), the DRC program reads the counter in VIA, performs the required arithmetic, and reinitializes the counter. A little software trick here: The counter in the VIA counts down. By initializing the counter to 1Ø,ØØ1, and having the A/D cause the counter to count down, the counter ends up with the resulting A/D conversion IF the reading is negative (1ØØØ1-COUNT). If the reading is positive, the software takes the two's (binary) or ten's (decimal) complement to yield the conversion. On page five we'll see why this trick was used!

CB1 goes high at the beginning of a conversion and goes low at the end of a conversion. The negative edge generates an interrupt indicating that the conversion is finished. In addition, the VIA clocks the data on CB2 (which is open, so it floats high) into the least significant bit of a shift register on each positive edge CB1. At the start of a conversion, the shift register is cleared. At the end of a conversion, the software checks to insure that the shift register holds the number 1, and nothing higher. This insures that only one conversion has been accumulated in the counter. It is possible (when doing disk accesses, or other operations that leave interrupts disabled for relatively long periods of time), to miss the end of a conversion, resulting in an invalid conversion in the VIA counter. If this occurs, the shift register will have a number higher than 1, and the firmware will throw out the conversion. For this reason, CB2 <u>must</u> be left open or high.

<u>Pages 2 & 3: Channel Select Relays</u>

These circuitry on these pages selects one of the ten floating sample voltages and sends it to the A/D converter on page 5. The appropriate relay is driven by the appropriate section of UØ5 or UØ6 in response to a channel select signal from the VIA on page 1. The reed relays (KØ1 through K1Ø) have internal spike suppression diodes to prevent high voltage transients that would appear when the relay is released. In addition, the high side of the relay coil is driven by +5-delay, generated by the circuitry on page 6. This prevents the reed relays from being activated until the system has been initialized.

R15, R16, R17 and R18 combine the switched samples to drive the A/D conveter (on page 5). These resistor networks provide some isolation between samples should a reed relay fail to release.

<u>Page 4: Control Line Drivers</u>

U14 through U22 are quad 2 input NAND gates with high current open collector outputs. One of the two inputs of each NAND gate is tied to the G input (an active high enable input). If, for example, CHANØ is high, and RAISE is high, section A of U14 will pull RAISEØ* low, driving external equipment. Lower commands are handled in a similar manner. The channel select lines are NANDed with a steady high (+5-DELAY after system initialization) to give the CHANOUT* outputs. These lines are pulled low when a particular channel is being accessed. These lines are typically used to drive tower select lines of antenna monitors.

U14 through U22 each include clamp diodes from the outputs to pins 2 and 7. These pins are tied to the CLAMP line, which has a 3Ø volt zener diode to ground (DØ3). These diodes and the zener conduct when the voltage on a control line exceeds 3Ø volts. This protects the output transistors in the chips from voltage transients from the external equipment.

<u>Page 5: A/D Converter</u>

UØ8 does the actual A/D coversion. Note that most of the devices on this page operate on a floating power supply provided by UØ7, which takes the +5 volt supply and converts it to a floating 12 volts.

The floating 12 volts is converted to a floating +/- 5 volts by DØ2, DØ1 and RØ9. UØ8 uses the +/- 5 volts (+FLOAT and -5-FLOAT) for its power supply. U1Ø operates on +FLOAT and FDIGITLGND, the floating digital ground. U1Ø operates on a net of 5 volts.

UØ9 is a temperature stabalized voltage reference. It contains a temperature controlled oven and a reference voltage circuit. The heater portion of UØ9 operates directly on the 12 volt output of UØ7 (+FLOAT and -FLOAT). The reference voltage generated by UØ9 (LM399H) is 6.95 volts. The top of this reference is connected to +FLOAT (the floating +5 volt supply). The low side of the reference (UØ9 pin 2) is a very stable 6.95 volts below the floating +5 volts. This 6.95 volt reference is divided down to 1 volt by precision resistors RØ6 and RØ7. RØ8 provides current through the reference from -FLOAT. We end up with about 6 volts dropped across RØ7 and 1 volt across RØ6. This gives us a precise 1 volt reference voltage across RØ7, with the high side being approximately at the floating digital ground, and the low side about 1 volt below that. The low side of the reference is connected to the analog ground and the negative sample input of the A/D (UØ8). This results in the analog ground (and the sample common mode) being about 1 volt below the digital ground, well within the common mode capabilities of the A/D.

The +SAMPLE and -SAMPLE are provided to the A/D from the sample selecting circuitry on pages 2 and 3. R2Ø (series) and R21 (shunt) form an optional voltage divider reducing the +SAMPLE..-SAMPLE signal to DIV SAMPLE..-SAMPLE. The A/D board is normally supplied with a jumper in the R2Ø position and R21 open. By adding these resistors (use ones with a very low tempco), samples higher than the maximum 2 volts the A/D will accept.

Other analog circuitry on page 5 includes the integrating capacitor CØ1, the auto-zero capacitor CØ2, the integrating resistor RØ5 and the reference capacitor CØ3.

During the sample integrate and reference de-integrate phases of the A/D conversion, CØ1 and RØ5 are used in a standard operational amplifier integrator. During the auto-zero phase, CØ2 is charged with a voltage to compensate for the offset voltages of the operational amplifiers and comparators in the analog to digital converter. Also during the auto-zero phase, CØ3 is charged to the reference voltage so that a floating reference is available for the reference de-integrate phase of the conversion. A floating reference is required since a different polarity of reference voltage must be applied to the integrator to cause it to integrate back towards zero output depending upon whether the polarity of the sample voltage caused the integrator output to go positive or negative during the sample integrate phase.

U1Ø is a high speed CMOS NAND Schmitt trigger (74HC132). This chip is operating on the 5 volts between +FLOAT and F-DIGITLGND.

U1ØA operates as an inverting Schmitt trigger oscillator at 12Ø KHz. CØ4 is charged and discharged between the high and low trigger points of the input of U1ØA through R1Ø. This 12Ø KHz square wave provides the required clock to UØ8. It also is NANDed with F-BUSY (the floating busy signal) by U1ØD. The output of U1ØD (F-BSY.CLK*) drives the LED in opto-coupler U11 through current limiting resistor R13. U11 inverts the F-BSY.CLK* and "unfloats" it to BSY.CLK. The VIA (UØ2 on page 1) counts these pulses to determine the A/D conversion.

U1ØB similarly drives U13 through R11. This unfloats F-BUSY to BUSY. The VIA uses BUSY to determine when a conversion has been completed.

U1ØC similarly drives U12 through R12. This unfloats F-POLARITY to POLARITY. The VIA uses POLARITY to determine whether the sample is positive or negative (a high indicates the sample voltage is positive).

Finally, R14 provides necessary pull-up resistors for U11, U12 and U13, since their outputs are open collector.

Use of the counter in UØ2 allows the complete A/D to float with only three opto-couplers. The alternative methods available to float the A/D would be to use about 11 opto-couplers to transfer the multiplexed BCD, busy and polarity signals, or to add circuitry to serialize the data.

Page 6: Output Connectors and Delayed+5 Generator

This page of the schematic shows the connections to the header connectors which then connect to the rear panel. Note that the numbering systems used on header connectors and D connectors (used on rear panel) are different.

R19 and QØ1 take the +5-volts supply and provide +5-delay once the system has been initialized and the VIA has pulled DELAY-RES* low. +5-DELAY powers the reed relays and the control line drivers. These are not enabled until the system has been initialized, preventing all the reeds from being pulled in on reset.

Theory of Operation 1442D Processor Board

The 1442D processor board consists of the following portions of the DRC190 remote control system:

- Processor
- Watchdog Timer
- Dual Serial Port (1 for modem, 1 for RS232)
- Parallel I/O (for modem control and keyboard scanning)
- 1200 Bit/Second half duplex modem
- 32K bytes of EPROM holding system program
- 2K bytes of TimeKeeper RAM holding calibration and setup data
- 32K bytes of RAM holding temporary data, pointers, and Basic program
- Parallel I/O providing IEEE488 port and CW ID for radio links
- STD bus interface to remainder of system

We'll take the schematic in order of page number and give you an idea what's going on.

Page 1: Processor, Memory Map Decode, Control Decoder, Watchdog Timer

The processor (U01) uses an 8.00 MHz crystal (Y02) to generate the 2.00 MHz bus clock (E or Phase2). This results in a bus cycle time of 500 nS and data strobes (E, phase 2, RD*, OE* and WR*) of 250 nS.

The NMI* (NonMaskable Interrupt) and IRQ* (maskable Interrupt ReQuest) lines are pulled up by R06 and R05. They are pulled low by I/O devices on this board and other boards when an interrupt is required.

In data transfers throughout the system, data is latched by the receiving device on the trailing (falling) edge of E. E is a continuous 2.00 MHz clock. It continues even if the processor is not doing an external data transfer (this occurs if the processor is doing an internal calculation, such as calculating a relative address). To prevent invalid access to external devices, the processor provides a VMA (Valid Memory Address) line. This line is high if the processor intends to do a data transfer on this cycle of the E line. VMA is low if no transfer is to be done.

The processor also outputs the R/W* line. This line is high when the processor is reading data (inputting from an external device) and low when the processor is writing data (outputting to an external device). If there is no data access in this cycle, R/W* remains high.

The approximate timing of all the processor generated signals is:

0 nS	Address and R/W* lines go to required state. VMA goes high if an external data transfer to occur in this cycle.
50 nS	E goes high
210 nS	Write data to external devices is valid
240 nS	Read data from external devices must be valid
300 nS	E goes low, latching data. (E high for 250 nS)
310 nS	Read data from external devices can be released (hold time)
320 nS	Address, R/W* and VMA lines go invalid
320 nS	Write data to external devices goes invalid (hold time)

500 nS Start at top of list again

U04 converts from the 65/68 bus (E, R/W*, VMA) to the Intel Bus (OE*, WE*). U04 is a 3 line to 8 line decoder with enables and active low outputs. When E is high, R-W* is high, VMA is high, and SYSRES (system reset) is low, The Y3 output of U04 goes low, driving the OE* (Output Enable) line for the rest of the system. For devices using the Intel control bus (the DUART and memory), the low OE* line causes these devices to put their data on the data bus. Similarly, if the same conditions are true except that R-W* is low, U04's Y2 output (WE* or Write Enable) goes low, causing the addressed device to latch the data then present on the data bus. The processor has written to the device.

SYSRES is included in the control decoding to prevent writes to the TimeKeeper RAM should the system have under voltage. SYSRES is high immediately after power up, immediately after pushing the rear panel reset button, or if the watchdog timer generates a system reset. Including SYSRES in the control decoding of U04 insures that the TimeKeeper RAM will not get any bad writes during processor shut down.

U02 is the memory map decoding PROM. Driven by the 5 most significant address lines, U02 breaks the 64 Kbyte address map into 32 blocks, each 2 Kbytes. Note that a Kbyte is considered to be 1024 bytes. U02 is enabled only when VMA* is low, which is when the processor is about to do an external data transfer. Chip Select signals are generated by U02 only when the processor is in an external data transfer cycle.

The outputs of U02 are the chip select lines (active low) to the various I/O and memory devices on the system bus. If VMA* is high, indicating that we do not want to select any devices, R01 and R02 pull all the chip select lines high, deselecting all devices.

The CS* lines select the below listed devices:

CS0*	On Board I/O
CS1*	Off board data transceiver
CS2*	U19 EPROM
CS3*	U20 TimeKeeper RAM
CS4*	U21 RAM
CS5*	Not used
CS6*	Off board I/O (STD IORQ*)
CS7*	Off board memory (STD MEMRQ*)

The actual memory map is shown on the next page.

U03 further divides the 2 Kbyte block of address space selected by CS0* into 256 byte blocks. Each of these blocks is assigned to an I/O device on the processor board. These chip select lines are:

CS00*	DUART
CS01*	VIA for IEEE488 and CW ID
CS02*	Front panel LCD

Processor Board Theory of Operation

DRC19Ø Memory Map PROM

for systems with 8 Kbytes of RAM on processor board
(external bank selected RAM)

System Address	PROM Address		PROM Data								
Hex	Hex	Octal	D7	D6	D5	D4	D3	D2	D1	DØ	Selected Device
ØØØØ	ØØ	ØØ	1	1	1	Ø	1	1	1	1	On Board RAM
Ø8ØØ	Ø1	Ø1	1	1	1	Ø	1	1	1	1	On Board RAM
1ØØØ	Ø2	Ø2	1	1	1	Ø	1	1	1	1	On Board RAM
18ØØ	Ø3	Ø3	1	1	1	Ø	1	1	1	1	On Board RAM
2ØØØ	Ø4	Ø4	Ø	1	1	1	1	1	Ø	1	Off Board RAM
28ØØ	Ø5	Ø5	Ø	1	1	1	1	1	Ø	1	Off Board RAM
3ØØØ	Ø6	Ø6	Ø	1	1	1	1	1	Ø	1	Off Board RAM
38ØØ	Ø7	Ø7	Ø	1	1	1	1	1	Ø	1	Off Board RAM
4ØØØ	Ø8	1Ø	Ø	1	1	1	1	1	Ø	1	Off Board RAM
48ØØ	Ø9	11	Ø	1	1	1	1	1	Ø	1	Off Board RAM
5ØØØ	ØA	12	Ø	1	1	1	1	1	Ø	1	Off Board RAM
58ØØ	ØB	13	Ø	1	1	1	1	1	Ø	1	Off Board RAM
6ØØØ	ØC	14	Ø	1	1	1	1	1	Ø	1	Off Board RAM
68ØØ	ØD	15	Ø	1	1	1	1	1	Ø	1	Off Board RAM
7ØØØ	ØE	16	Ø	1	1	1	1	1	Ø	1	Off Board RAM
78ØØ	ØF	17	Ø	1	1	1	1	1	Ø	1	Off Board RAM
8ØØØ	1Ø	2Ø	Ø	1	1	1	1	1	Ø	1	Off Board RAM
88ØØ	11	21	Ø	1	1	1	1	1	Ø	1	Off Board RAM
9ØØØ	12	22	1	Ø	1	1	1	1	Ø	1	Off Board I/O
98ØØ	13	23	1	1	1	1	Ø	1	1	1	U2Ø TimeKeeper RAM
AØØØ	14	24	1	1	1	1	1	1	1	Ø	On Board I/O
A8ØØ	15	25	1	1	1	1	1	Ø	1	1	U19 EPROM
BØØØ	16	26	1	1	1	1	1	Ø	1	1	U19 EPROM
B8ØØ	17	27	1	1	1	1	1	Ø	1	1	U19 EPROM
CØØØ	18	3Ø	1	1	1	1	1	Ø	1	1	U19 EPROM
C8ØØ	19	31	1	1	1	1	1	Ø	1	1	U19 EPROM
DØØØ	2A	32	1	1	1	1	1	Ø	1	1	U19 EPROM
D8ØØ	2B	33	1	1	1	1	1	Ø	1	1	U19 EPROM
EØØØ	2C	34	1	1	1	1	1	Ø	1	1	U19 EPROM
E8ØØ	2D	35	1	1	1	1	1	Ø	1	1	U19 EPROM
FØØØ	2E	36	1	1	1	1	1	Ø	1	1	U19 EPROM
F8ØØ	2F	37	1	1	1	1	1	Ø	1	1	U19 EPROM

DRC19Ø Memory Map PROM

with 32 Kbytes of RAM on processor board
(no external bank selected RAM)

System Address Hex	PROM Address Hex	PROM Address Octal	PROM Data D7	D6	D5	D4	D3	D2	D1	DØ	Selected Device
ØØØØ	ØØ	ØØ	1	1	1	Ø	1	1	1	1	On Board RAM
Ø8ØØ	Ø1	Ø1	1	1	1	Ø	1	1	1	1	On Board RAM
1ØØØ	Ø2	Ø2	1	1	1	Ø	1	1	1	1	On Board RAM
18ØØ	Ø3	Ø3	1	1	1	Ø	1	1	1	1	On Board RAM
2ØØØ	Ø4	Ø4	1	1	1	Ø	1	1	1	1	On Board RAM
28ØØ	Ø5	Ø5	1	1	1	Ø	1	1	1	1	On Board RAM
3ØØØ	Ø6	Ø6	1	1	1	Ø	1	1	1	1	On Board RAM
38ØØ	Ø7	Ø7	1	1	1	Ø	1	1	1	1	On Board RAM
4ØØØ	Ø8	1Ø	1	1	1	Ø	1	1	1	1	On Board RAM
48ØØ	Ø9	11	1	1	1	Ø	1	1	1	1	On Board RAM
5ØØØ	ØA	12	1	1	1	Ø	1	1	1	1	On Board RAM
58ØØ	ØB	13	1	1	1	Ø	1	1	1	1	On Board RAM
6ØØØ	ØC	14	1	1	1	Ø	1	1	1	1	On Board RAM
68ØØ	ØD	15	1	1	1	Ø	1	1	1	1	On Board RAM
7ØØØ	ØE	16	1	1	1	Ø	1	1	1	1	On Board RAM
78ØØ	ØF	17	1	1	1	Ø	1	1	1	1	On Board RAM
8ØØØ	1Ø	2Ø	Ø	1	1	1	1	1	Ø	1	Off Board RAM
88ØØ	11	21	Ø	1	1	1	1	1	Ø	1	Off Board RAM
9ØØØ	12	22	1	Ø	1	1	1	1	Ø	1	Off Board I/O
98ØØ	13	23	1	1	1	1	Ø	1	1	1	U2Ø TimeKeeper RAM
AØØØ	14	24	1	1	1	1	1	1	1	Ø	On Board I/O
A8ØØ	15	25	1	1	1	1	1	Ø	1	1	U19 EPROM
BØØØ	16	26	1	1	1	1	1	Ø	1	1	U19 EPROM
B8ØØ	17	27	1	1	1	1	1	Ø	1	1	U19 EPROM
CØØØ	18	3Ø	1	1	1	1	1	Ø	1	1	U19 EPROM
C8ØØ	19	31	1	1	1	1	1	Ø	1	1	U19 EPROM
DØØØ	2A	32	1	1	1	1	1	Ø	1	1	U19 EPROM
D8ØØ	2B	33	1	1	1	1	1	Ø	1	1	U19 EPROM
EØØØ	2C	34	1	1	1	1	1	Ø	1	1	U19 EPROM
E8ØØ	2D	35	1	1	1	1	1	Ø	1	1	U19 EPROM
FØØØ	2E	36	1	1	1	1	1	Ø	1	1	U19 EPROM
F8ØØ	2F	37	1	1	1	1	1	Ø	1	1	U19 EPROM

Processor Board Theory of Operation

The only remaining circuitry on page 1 of the processor schematic involves U31, the reset generator - watchdog timer. U31A forms a standard schmitt trigger astable multivibrator. On power up, CØ3 is discharged, so CØ3-2 follows CØ3-1 up to +5 volts. This places +5 volts on the input of U31A, forcing its output (SYSRES*) low, resetting the processor and all other chips in the system that have a hardware reset. When the output of U31A is low, the bottom of CØ3 (pin 2) is pulled low quickly through DØ5 and R44. When the input of U31A is pulled below +1.78 volts, the output of U31A goes high. When the output of U31A is high, DØ5 is reverse biased, preventing any current from going through DØ5 and R44. Instead, pin 2 of CØ3 is slowly pulled high through R43. When CØ3-2 goes above +2.78 volts, the output of U31A goes low. This results in the output of U31A going low for 18 mS after power up. It then stays high for about 25 seconds.

The 25 second period is the amount of time it takes for U31A to pull CØ3-2 high through R43 (1 M). If CØ3-2 is pulled low, this time is reset. U16-16 (on page 8) generates pulses when the DRC19Ø firmware is operating properly. These pulses are capacitively coupled through C24 to U31B, resulting in pulses out of U31B. When U31B is low, it pulls CØ3-2 low through DØ8 and R44, resetting the 25 second timeout. The input to U31B is capacitively coupled so that should we get a processor crash when WATCHDOG* is high, R47 will pull the input of U31B low, releasing CØ3-2 and allowing it to time out. A steady high or low voltage from U16 will not reset the watchdog timer. A series of pulses must be presented. This insures that a firmware failure (processor crash) results in a system reset.

Finally, U31C and DØ9 can force CØ3-2 high, generating a system reset in response to a low pulse on the PBRES* line, which is grounded by the rear panel reset button.

Page two: DUART, Keyboard Interface, RS-232 Interface, Clock

Page two of the 1442 schematic is centered on the DUART. The processor interface of the DUART is pretty standard. The address lines AØ-A3 select one of 16 internal registers of the DUART. WE* and OE* cause the DUART to latch up data on the bus or to output data to the bus if the chip is selected (CSØØ* low). SYSRES is normally low, but goes high when the system is reset. This resets the registers inside the DUART. UØ6 (the DUART) can request an interrupt by pulling the IRQ* line low. A timer in UØ6 is normally programmed to generate an IRQ* every 1Ø mS. YØ1 works with the oscillator in UØ6 to generate a 3.6864 MHz clock. This is used by UØ6 to run the internal baud rate generators for the serial ports.

Serial port A goes through UØ7 and UØ8, converting the data from TTL to RS232 levels, and the reverse. The RS232 data is presented to PØ2, which connects to J22 on the rear panel. Note that the pin numbers on J22 do not correspond with the pin numbers on PØ2 because of the different numbering methods for double row header connectors and D connectors. In addition, input 2 (UØ6-36) is driven by the DCD or DTR line of an external modem or terminal through UØ8-B. This allows Basic applications programs to determine the state of this line, often used to determine if an external modem is receiving data carrier.

Serial port B goes directly to the 12ØØ bit/second modem.

UØ6 also includes a parallel input port and a parallel output port. The input port serves the following purposes.

IPØ is driven by the KEY line. Between keyscans, the row lines of the keyboard are driven low. The column lines are pulled high by RØ3. If a key is pressed, one of the column lines will be pulled low by the short in the keyboard at the row/column intersection. This low will cause the output of U27A (key) to go high.

IP1 is driven by the demodulator portion of the modem. This line is high if the modem is receiving a carrier. This line is used to insure that modem data is accepted only when a carrier is present.

IP3 through IP6 are driven by the column lines from the front panel keyboard. Once we have determined that a key is closed, the row ouput lines can be pulled low one at a time until a low shows up on one of the column lines. Once that happens, the low row and low column lines correspond to the row/column code of the position of the pressed key. The DRC19Ø program debounces this in software to insure the key closure is valid. It also converts the key code to ASCII for use elsewhere in the program.

UØ6 also has a parallel output port. This port serves these purposes.

OPØ drives the modulator portion of the modem. When this line (XMIT*) is low, the modem carrier is turned on.

OP1 drives U13C on page 6 of the schematic. When this line (SPKRCOM) is high, a reed relay (KØ1) moves the front panel speaker from the output of the speaker driving amplifier to the input of the line driving amplifier. The speaker then serves as a microphone in the intercom function of the DRC19Ø.

OP2 drives U13B on page 6 of the schematic. When this line (LINE-EN) is high, reed relay (KØ2) connects the primaries of line driving transformers TØ3 and TØ4 to the line driving amplifier (U14). When the line is not enabled, the open primaries of TØ3 and TØ4 provide a high secondary impedance, preventing this unit from loading the line. This allows other units on the line to drive the line. This is a sort of audio "Tri-State" system.

OP3 drives U15 on page 7 of the schematic through DØ3 and R33. When this line (SPKRMUTE) is high, speaker driving amplifier U15 is driven into saturation, preventing audio from going through U15 to the speaker. This mutes the speaker. In addition, SPKRMUTE is pulsed low for 1Ø mS each time a key on the front panel keyboard is pressed. This gives an audible click each time a key is pressed, providing audible feedback.

OP4-OP7 drive the row lines on the keyboard. These idle low. When a key is pressed, a column line is pulled low, causing KEY, the output of U27A to go high. Once the key closure has been detected, OP4 through OP7 are pulsed low one at a time. When one of the row lines being low causes one of the column lines to go low, the key has been found. It is at the intersection of the low row and column lines.

This completes the description of the circuitry on page 2 of the 1442 processor board schematic.

<u>Page 3: FSK Generator</u>

This page covers the FSK generator. This is a simple application of the XR22Ø6CP chip (U1Ø).

CØ7 provides DC bypass required by U1Ø.

The oscillator section of U1Ø oscillates at a frequency determined by CØ4 and R14 or R15. The selection of R14 or R15 is based on whether U1Ø pin 9 is high or low. If pin 9 is high (Mark condition), R15 is selected, causing U1Ø to operate at 22ØØ Hz. If pin 9 is low (Space condition), R14 is selected, causing U1Ø to operate at 12ØØ Hz.

The amplitude modulator in U1Ø is disabled (forced to 1ØØ% carrier) by gounding pin 1. A sine (rather than triangle) output is selected by setting R16 to 2ØØ ohms.

Pin 3 is normally biased to +6 volts through R11, R12 and R1Ø. CØ5 holds the junction of R11 and R12 at AC ground. R1Ø pulls pin 3 towards AC ground, reducing the audio level at pin 3. This serves as the output level adjustment.

U13F grounds pin 3 when no output is desired. This forces the output amplifier in U1Ø to cut off due to lack of bias. U13F has an open collector output, so it has no effect if the input of U13F (XMIT*) is low. Finally, the output amplifier in U1Ø takes the signal present on pin 3 and amplifies it by 1 (actually amplifies the current) and gives the output on pin 2. The DC component of this is removed by CØ6 and is presented to the line driving amplifier (on page 6) as TXAUDIO1.

When in MODEMTST, causing U1Ø to be enabled, approximately 2.8 V P-P on +6 VDC is present on U1Ø pin 2.

Page 4: Input Summing Amplifier

The input summing amplifier is a simple unity gain summing amplifier that sums the communications line inputs from TØ1 and TØ2. The output is sent to the FSK demodulator (U12 on page 5) and the speaker amplifier (U15 on page 7) for use in the intercom mode.

R19, R18 and R17 determine the gain of the summing amplifier. R2Ø and R38 reduce the output of the summing amplifier down to a level suitable for the speaker driver amplifier (page 7), since the received audio is sent to the speaker in the intercom receive mode. CØ8 provides DC blocking between the input summing amplifier and the speaker amplifier.

C1Ø provides DC blocking between the input summing amplifier and the FSK demodulator. C1Ø, when combined with the 2ØK input impedance of the XR2211 FSK demodulator, forms a high pass filter with a cut off frequency of about 6ØØ Hz. This rejects any AC hum that may be present on the received audio.

Page 5: FSK Demodulator

Page 5 of the 1442 processor schematic is the FSK demodulator portion of the 12ØØ bit/second modem. This is a standard application of the XR2211CP demodulator chip.

The XR2211 is a phase locked loop FSK demodulator. The free-running frequency of the Voltage Controlled Oscillator (VCO) is set using CØ9, R22 and R21. The VCO normally free-runs at 17ØØ Hz.

The input signal from the summing amplifier (page 4) is applied to U12 pin 2. U12 amplifies this signal and applies it to two phase detectors (called the loop phase detector and the quadrature phase detector). The other input of each phase detector is driven by the VCO. The VCO signal fed to the quadrature phase detector is shifted by 9Ø degrees from the signal feeding the loop phase detector.

The preamp in U12 allows the chip to operate with receive levels between 1Ø mV RMS and 3 V RMS. This gives a minimum receive level of -37.8 dBm (6ØØ ohm line).

The output of the loop phase detector appears on U12 pin 11. The internal resistance of this output along with C12 form the loop low pass filter, rejecting the double carrier output of the phase detector while allowing the 12ØØ bit/second data to pass around the loop. The filtered output of the phase detector goes through R26 to the VCO through pin 12. The value of R26 determines the loop gain, which sets the capture and lock range of the PLL. As the input signal changes frequency, the loop phase detector detects the changing phase relationship between the VCO and the input signal, generating a changing VCO control voltage, causing the VCO to track the input signal. Since the control voltage into the VCO is proportional to the frequency, the control voltage ends up being proportional to the incoming frequency. This is compared with a reference voltage (available on pin 1Ø of U12) with hysteresis provided by R27. In addition, R25 and C11 provide a data low pass filter, further reducing any 2 times carrier components present while allowing the 12ØØ bit/second data to pass.

The "sliced" demodulated FSK appears on pin 7 of U12. At this point, a Mark state (22ØØ Hz) is low, and a Space state (12ØØ Hz) is high.

The quadrature phase detector is used to detect the presence of a data carrier. This phase detector compares the incoming signal (U12 pin 2) with a 9Ø degree shifted output of the VCO. The resulting signal has two components. One component is an AC signal with a frequency of twice the VCO (the sum of the incoming signal and the VCO frequency). The other component (the "difference" frequency) is a DC component proportional to the amplitude of the incoming signal. The quadrature phase detector portion of U12 forms a synchronous AM demodulator. The output of the quadrature phase detector is available on pin 3 of U12. R23 determines the "gain" of the quadrature phase detector, while C1Ø filters the output of the phase detector, removing the double VCO component.

The filtered output of the quadrature phase detector is compared with the reference voltage (once again, available on pin 1Ø) to determine if sufficient signal is present to indicate carrier presence. R24 provides positive feedback around the carrier detect comparator, providing hysteresis. This hysteresis insures that any double carrier frequency signal does not get through the comparator. The reference voltage is (V+/2)-65Ø mV. With our 5 volt supply, this works out to 1.85Ø volts. As long as the voltage on pin 3 is over 1.85 volts, U12 pin 6 (RX-CAR) will be high. It is quite interesting to watch the voltage on pin 3 with a DC coupled scope as the DRC19Ø operates. When the data carrier first comes up, the voltage on pin 3 approaches 5 volts, vastly exceeding the required 1.85 volts to indicate carrier presence. As the carrier is keyed (with FSK), a ripple appears in the voltage on pin 3. This is due to the slight delay in the VCO tracking the incoming signal, which results in a varying phase error out of the quadrature phase detector. As long as the minimum voltage of the ripple does not go below 1.85 volts, U12 will continue to indicate carrier presence. A solid carrier detect with keying is guaranteed with a -3Ø dBm input and typically achieved with -4Ø dBm.

The carrier detect portion of the FSK demodulator is quite critical to the proper operation of the system. Carrier detect is checked as each character of data is received from the modem. If valid carrier is not present, the data is thrown out and the modem receive routine is reset.

Processor Board Theory of Operation

Page six: Line Driver, Speaker Driver, External TX Key

U14 of the PC1442 processor board acts as a simple summing amplifier. Most inputs use U14 as an inverting amplifier.

The CWID signal from the Morse code identifier is amplified by 5 (R29/R39) and presented to relay KØ2.

The TXAUDIO1 signal from the FSK modulator is amplified by 1 (R31/R29) and presented to relay KØ2.

When the SPKRCOM signal from the DUART is high, U13C activates relay KØ1. This switches the front panel speaker from the output of the speaker driver amplifier (COM-OUT) to the non-inverting input of U14. The speaker then acts as a microphone whose signal is amplified by 1ØØ1 (1+(R29/R3Ø)) and applied to KØ2. C13 keeps U14 a voltage follower as far as DC is concerned, preventing excessive output offset voltage. R32 provides a bias path for the non-inverting input of U14 when KØ1 is not activated.

At this point, we can have one of various signals present at the input of KØ2 (LINE-OUT). When the LINE-EN output of the DUART (on page 2 of the schematic) goes high, U13B activates KØ2, connecting the output of U14 to the communications lines through TØ3 and TØ4. When KØ2 is released, the primaries of TØ3 and TØ4 are left open, presenting a high impedance to the communications lines. This allows other DRC19Øs in the system to drive the line withou contention. This forms a sort of audio "Tri-State" bus.

Finally, the output of U13B (LINE-EN*) is double inverted by U13D and U13E and brought out the rear panel of the DRC19Ø as TXKEY*. Note that U13 is an open collector device, so LINEEN** out of U13D is pulled up by section 4 of R28 on page 5 of the schematic. Since TXKEY* is low whenever the DRC19Ø is putting data on the communications line, this can be used to key an external UHF TRL transmitter. Due to transmit bring-up and receiver squelch delays, it is necessary to increase the site delay time from the normal Ø.Ø5 seconds to about Ø.2Ø seconds when such a system is used. This is handled in the calibration and setup of the DRC19Ø.

When the DRC19Ø utilizes an internal UHF telemetry transceiver (Neulink CP-4Ø3U/TMC), T4 is replaced with a voltage divider to drop the transmit audio level down to 8 mV RMS to drive the transceiver.

Page 7: Speaker Driver Amp

The speaker driver amp is an almost standard application of the LM38Ø-8 (U15). The non-inverting input of U15 is driven by the output of the input summing amplifier (U11 on page 4) divided down by R2Ø and R38 (also on page 4). Further, CØ8 on page 4 blocks DC allowing the internal biasing network of U15 to operate.

The output of U15 passes through C15, removing the +6 volt DC component. R35 prevents C15 from discharging due to leakage when the speaker is not connected. The COM-OUT signal from C15 goes to KØ1 on page 6 where it is sent to the front panel speaker unless we are in the intercom talk mode (when the speaker is used as a microphone and the output of U15 is ignored).

Finally, the SPKRMUTE output of the DUART on page 2 drives DØ3, R33, R34, and C14. SPKRMUTE is at +5 volts when the speaker is to be muted (which is most of the time). This is divided down by R33 and R34 and applied to the inverting input of U15. This forces the output of U15 to ground, preventing any audio present on COM-IN from driving the speaker.

When SPKRMUTE is low, the inverting input of U15 drops to Ø volts, allowing the signal on COM-IN to be amplified by U15 and drive the speaker. Since SPKRMUTE is driven by a transistor in the DUART, it cannot actually go all the way to ground. It does, however, go close enough to ground to not exceed the "knee voltage" of DØ3, insuring that the voltage into the inverting input of U15 is indeed Ø when the speaker is unmuted.

SPKRMUTE is pulsed low for about 1Ø mS each time a key on the keyboard is pressed to provide audible feedback of the keyboard operation. This "clicks" the speaker as U15 is pulled out and then back into saturation. C14 provides some pulse shaping. Since the keyboard pulse is actually unmuting the speaker, any data present on the input to the DRC will be heard through the speaker during the speaker click. This causes the occasional chirp instead of click during keyboard operation.

Page 8: IEEE488, Display Connector, CW ID

This page of the schematic is built around U16 (with the exception of PØ4, the display connector).

PØ4 provides the required data and address lines to the front panel display module. In addition, a chip select line (CSØ2) is provided by UØ5E. UØ5E inverts the active low chip select (CSØ2*) from the on board I/O decoder UØ3 on page 1. Also, CSØ2* is inverted by UØ5E, NANDed with the processor clock E by U27B, and finally inverted by UØ5F. This signal (CSØ2.E) is applied to pin 14 of PØ4. PØ4 is a 16 pin socket that will accept cables from a variety of displays. Some displays (such as IEE Daystar) require 16 pin connectors. These displays have an active high chip select input that can be driven by the memory map decoding (inverted by UØ5E). Some displays (such as the Sonicor) require only 14 pins. These displays do not have a separate chip select input. The E input needs to be gated with the chip select line. This is handled by U27B and UØ5F. When a 14 pin display is used, the bottom two pins (pins 8 & 9) of PØ4 are not used. The display appears to be a standard I/O device to the processor. Control (display clear, cursor positioning, etc.) and display data are written to the display. Status of the display can be read by the processor. R37 and R36 provide a variable bias to the display to adjust the viewing angle. The multiplexed LCD has a limited viewing angle. Adjustment of R37 optimizes the contrast at the desired viewing angle.

The left side of U16 is the standard processor bus interface. In addition, the IRQ* output of U16 is tied to the IRQ* input of the processor on page 1, allowing U16 to request an interrupt.

The majority of U16 is devoted to the IEEE488 instrumentation port. DI01 through DI08 are the 8 parallel bi-directional data lines for the bus. These lines are bufferred by U17 and U18 before being presented to the rear panel connector (J25) through PØ5 and the associated cable.

The other IEEE488 lines (DataAVailable, NotDataACcepted, EndOrIdentify, InterFaceClear, ATtentioN, RemoteENable, ServiceReQuest and NotReadyForData) are similarly bufferred by U17 and U18. The HI-TALK output of U16 drives the direction select lines of U17 and U18, allowing the lines to be biidirectional.

At this writing (14 January 1987 the software for the IEEE488 interface has not been written. A more detailed description of the operation of this portion of the circuit will be written when it works!

Processor Board Theory of Operation

U16 is also used to generate a 1.628 KHz tone for the Morse Code CW Identifier. U16 divides the system clock (2.ØØ KHz on the E line) to 1.628 KHz and presents it to PB7 (CWTONE). In addition, under software control, the CWTONE output is enabled and disabled as required to generate the Morse code identifier. The DC component of the 5 volt square wave CWTONE is removed by C2Ø. R4Ø and C21 form a low pass filter, turning the square wave to a triangle wave. This amplitude of this tone is set by R41 and sent to the line driving amplifier on page 6.

The WATCHDOG* output of U16 is pulsed low at various times in the program to reset the watchdog timer on page 1. See page 1 for a detailed description of the watchdog timer circuitry.

EEPROM-WP is high when we wish to protect the processor board TimeKeeper RAM (U2Ø on page 9) from a write. The vast majority of the time, the TimeKeeper RAM is only read. It may be read to find calibration scaling factors, communications timing, the system "boot" program, etc. The TimeKeeper RAM is written to only during calibration, set up, or a SAVE EEPROM command from Basic. Further, since the TimeKeeper RAM holds important data that could keep the system from operating if it were corrupted, it is important that the data not be disturbed in the occasional system crash. EEPROM-WP is pulled up by section 5 of RØ3 on page 2. This output of U16 is normally programmed high and into the input mode (similar to WATCHDOG*). EEPROM-WP is ORed with WE* by an OR gate formed by DØ6, DØ7, R45, and sections 2 and 3 of RØ9 on page 9. Prior to doing a write to TimeKeeper RAM, the DRC firmware sets EEPROM-WP low, allowing WE* to force EEPROM-WE* (on page 9) low. When the DRC is not writing to the TimeKeeper RAM, EEPROM-WP is set high and into the input mode. Once again, it is unlikely that a system crash would do the proper sequence to generate a false write enable to the TimeKeeper RAM.

<u>Page 9: Memory</u>

The three memory devices are put directly on the processor bus. Note that the 68BØ2 bus does not have WE* or OE* signals, so these are generated by the control decoder UØ4 on page 1.

U19 is a 27256 EPROM (32 Kbytes). Note that not necessarily all of the chip is enabled. It can be brought up in 2 Kbyte blocks as necessary by the memory map PROM UØ2 on page 1.

U2Ø is a CSF Thompson MK48TØ2 TimeKeeper RAM. This chip includes almost 2 Kbytes of low power static RAM (to hold calibration data and SAVE EEPROM programs), a real time clock/calendar and a lithium battery. This chip keeps track of the time, date, and calibration data in a power failure. This chip is used to hold the label and units characters for each metering channel, the metering curve (linear, square law, etc.) and the calibration scaling factor for each metering channel. In addition, this chip holds the site delay, site number, maximum site number, fail safe site numbers, control enable site numbers, communications and terminal bit rates, and the Morse CW identifier. It also holds a small Basic "boot" program that is executed on power up. This program is saved to TimeKeeper RAM using SAVE EEPROM and loaded on power up or using LOAD EEPROM.

As mentioned above, DØ6, DØ7, R45, RØ9, and an output of U16 protect the TimeKeeper RAM from false writes.

U21 is a 32 Kbyte or 8 Kbyte static RAM, depending on how the system was assembled. The standard system utilizes a 32 Kbyte RAM for U21. Some system utilize an 8 Kbyte RAM for U21, allowing the remainder of the RAM space in the memory map to be bank switched. Typically, a second bank of RAM is battery backed to allow non-volatile storage of a program using the statements SAVE EEPROM 2,8192 and LOAD EEPROM 2,8192. System RAM holds temporary data and pointers for the whole system. In addition, system memory holds a Basic program and defined variables for an automatic logging and control program. The system RAM is expanded by plugging an additional memory board into the system STD bus.

Page 1Ø: STD Bus Drivers

U24 and U25 drive the STD bus with the addresses out of the processor at all times (no off board device can access memory on the processor board). U26 drives the control lines of the STD bus with the appropriate lines from the processor or derived from the processor.

U23 acts as a bi-directional data transciever, passing data to and from the STD bus. The direction of data transfer is determined by the R/W* line on pin 1. When R/W* is high, data is taken from the bus and sent to the processor. When R/w* is low, data is taken from the processor and sent to the STD bus. Pin 19 enables U23. All the data lines on both sides of the chip are in a high impedance state unless pin 19 is low. Pin 19 is driven low by CS1* from UØ2 on page 1 when the processor is accessing an off-board address. The rest of the time, U23 allows the STD bus to float, and allows other devices on the processor board to drive the processor bus.

Power, Disk and Status Interface Theory of Operation

Theory of Operation: PC1443C Interface

The PC1443C (revision C) interface board serves a couple of purposes. It handles the interface between the system and the various power supplies and it provides a couple of serial interfaces for operating the disc drives and status panels.

Power Supply Interface

The power supply interface portion of the PC1443 takes power from the main supply and feeds it to the system backplane. In addition, it charges the UPS battery.

The battery is charged by regulating 17 volts DC (taken across the +5VDC and -12VDC outputs of the main power supply) down to the required 14.ØØ volt charging voltage. This is accomplished by programmable regulator UØ1 along with R13, RØ2, and RØ3. RØ1 provides current limiting on the regulator.

The UPS module converts 14 VDC to about +/- 1ØØ volts DC. This is connected to the filter capacitors of the off-line rectifier circuit of the main power supply. When the AC line is present, it charges these capacitors to approximately 16Ø volts. This back biases diodes in the UPS module, preventing it from providing any power. When the UPS module is idling, it draws about 5Ø mA from the 14 volt supply. The negative 14 volt input to the UPS module is routed through R11 on the 1443 board. When the UPS is idling, the voltage developed across R11 is not enough to cause QØ1 to conduct, so the voltage at the adjust input of UØ1 is determined by the voltage divider action of RØ1, RØ2 and RØ3. This allows the 1443 board to charge the battery when the UPS is idling (the AC line is present).

Should the AC line fail, the voltage across the main power supply input capacitors drops to 1ØØ volts, forward biasing the output diodes in the UPS module. The UPS module takes over providing DC to the primary switcher in the main power supply. This increased load on the output of the UPS module increases its input current to above 1ØØ mA. This current causes sufficient voltage drop across R11 to cause QØ1 to conduct, pulling down the adjustment input of UØ1. This drops the output voltage of the charging circuit, back biasing DØ1, preventing the battery from being charged when there is no AC line present. When AC is absent and inverter input current is high, a diode across R11 (physically located between the battery and the inverter) handles the majority of the current, limiting the voltage drop across R11 to a little over Ø.7 volts. This insures a maximum voltage is provided to the inverter while still allowing for an inverter current sense. Locating the diode betwen the battery and the inverter (instead of on the circuit board) eliminates the voltage drop in wiring to the board and back. This charger shutdown circuit prevents power from the battery being used to charge the battery. . . a losing proposition.

PØ1 of the 1443 board connects to the battery, the UPS module and the main power supply. In addition, it connects to the rear panel system reset button. The PBRESET* (Push Button Reset, active low) line is routed directly from PØ1 to the back plane through the 1443 edge connector. In addition, RØ4 provides current limited +5 volts to run the LED in the reset button.

The wiring of PØ1 is listed below.

PØ1 pin	Wire Color	Connects to
1	Brown	+5 volts from power supply
2	Red	+5 volts from power supply
3	Orange	Common from power supply
4	Yellow	Common from power supply
5	Green	+12 volts from power supply
6	Blue	-12 volts from power supply
7	Violet	Battery +
8	Gray	Battery -
9	White	UPS + input
1Ø	Black	UPS - input
11	Black/White	Reset switch
12	Black/White	Reset switch
13	Red/White	Reset switch LED +
14	Orange/White	Reset switch LED

Disc Interface

The remainder of the 1443 board provides interface to disc drives and status panels. Each of these uses a serial data bus. They are driven with software through UØ4.

PØ3 drives the Commodore serial bus, which is used to interface with a disk drive. UØ4 drives the bus through sections of UØ5 under software control. The Commodore serial bus signals are described below:

The ATN output of UØ4 is inverted and converted to an open collector output with a pull-up resistor by UØ5A and a portion of RØ7. The active low ATN* output is driven low when the DRC19Ø is sending an interface instruction to a disk drive or other device on the bus. These interface instructions tell specific devices to listen or talk on the bus. An actual bus transaction is described a bit later in this section.

The CLK output of UØ4 is inverted by UØ5B yielding CLK*, another open collector bus signal pulled up by another section of RØ7. The clock line is a bidirectional line that can be driven by the DRC19Ø or by the outside device. When the CLK* line is driven by an external device, its state is read by the PB2 input of UØ4. The DRC19Ø drives PB1 when it wishes to drive the clock line. When the DRC19Ø is to receive clock signals, PB1 is set low, allowing CLK* to go high. The external device is now able to drive the CLK* line to the desired state, which is detected on the PB2 input of UØ4.

In a similar manner, PB3 and PB4 of UØ4 and UØ5C drive the DATA* line and receive data off the line.

A request for a directory from the disk drive might appear as follows:

We will send a command to the disk drive (typically device 8, the primary address), with a secondary address of Ø, telling it to listen to the bus. The command is typically expressed as MLA8,Ø (My Listen Address 8, Secondary address Ø).

Sending this command on the serial bus goes like this. Through the VIA on the disk drive board (UØ4), we force the serial bus ATN* line low, telling other devices on the bus we are about to send a command character. We then send the primary address over the bus.

To send a character over the bus, we do the following. We start with a bus handshake to insure that all the devices on the bus are ready to receive the character. This same handshake and character transmit sequence is used to transmit any character over the bus. Since the ATN* line is low, the devices on the bus know that this is a command character.

The serial bus "talk" handshake goes like this. The CLK* line is forced low and the DATA* line is released. The other devices on the bus should pull the DATA* line low. If this does not occur, the DRC19Ø indicates a serial bus error (SB error). The CLK* line is released. The DRC19Ø then waits for all devices on the bus to release the DATA* line. Since all devices are open collector, any device can hold the DATA* line low, indicating it is not yet ready to receive a character. When all devices are ready, DATA* is released, the DRC19Ø detect this, forces the CLK* line low, and starts transmitting the character.

A byte is sent over the bus following the talk handshake. The byte is sent in this manner. The least significant bit of the byte to be transmitted is placed on DATA*. If the LSB is Ø, DATA* is driven low. If the LSB is 1, DATA* is released, and is pulled up by the pull up resistor. The CLK* line is pulsed high, then low, causing the devices on the bus to capture the bit. The next bit of the byte is put on the DATA* line, and the CLK* line is pulsed high. The byte is transmitted with no handshake between the bits. After the last bit (the MSB of the byte) has been transmitted, the DRC19Ø releases the DATA* line and waits for the external device to pull DATA* low. This "frame handshake" indicates that the 8 bits have been received properly. If the frame handshake fails, an SB error results.

We have just sent the primary address of the device we are telling to listen over the bus. We then send a secondary address of Ø (actually, $FØ is sent, indicating a secondary address of Ø). The ATN* line is then released, indicating we are finished sending the command.

Device 8, the disk drive, is now listening to the serial bus. We transmit the character "$", which indicates we want the directory, without driving ATN* low, since this is data instead of a command. The $ is sent in a manner similar to the other characters sent over the bus, except that since this is the only data character to be sent, we flag it with an "EOI" (End or Identify, here we are using it to identify the end of the message). We use a talk EOI handshake instead of the previously discussed talk handshake. The two handshake sequences are the same except that when we release the DATA* line and wait for the listening devices to let the DATA* line go high, we do not immediately respond to the high DATA* by forcing the CLK* line low. Instead, we leave DATA* high. This "not pulling" CLK* low is recognized by the addressed device as an indication of EOI. It acknowledges the EOI by pulsing DATA* low. When DATA* is again released by the receiving device, the "talk EOI" handshake is complete. The character is then transmitted in the normal manner.

In going further in our directory command, the addressed drive is now told to "unlisten". The UNL command does not include an address, and it tells all devices to stop listening. The UNL command consists of a $3F sent over the bus with ATN* low.

The disk drive is then given permission to talk on the bus. This command (MTA8.Ø) is sent in the same manner as the previous MLA command (with ATN* low). The MTA command adds $4Ø to the primary address (yielding $48 for drive 8), and, as previously, adds $FØ to the secondary address.

Since the disk drive had just received a command, and is now expected to talk on the bus, a bus turn-around sequence is called. This turn-around sequence goes as follows: The DATA* line is forced low by the DRC190. The CLK* line is released. We wait for the addressed device to pull the CLK* line low (if this fails, an SB error results). This completes the bus turn-around, with the addressed drive ready to send data.

The DRC190 then receives a byte off the serial bus. The DATA* line is driven low by the DRC190. The CLK* line is released (although it is being held low by the disk drive waiting to talk). We then wait for the disk drive to release the CLK* line. On detecting this, we handshake with the drive by releasing the DATA* line. We then wait for the device to pull CLK* low. If this does not occur within 200 uS, the drive is sending an EOI, indicating that the next character to be sent is the last one in the message. If we detect EOI, it is acknowledged by pulsing DATA* low, then high. Finally, we wait for CLK* to go high, indicating the drive has placed a bit of data on DATA*. This bit (the LSB of the byte being received) is taken off the DATA* line. The bit is a 1 if DATA* is high. The DRC190 then waits for CLK* to go low, then high again, indicating the next bit is on the DATA* line. This repeats eight times to capture the 8 bits. The frame handshake occurs at the beginning of the next received character.

The first two bytes sent by the drive are thrown out, since they represent a load address that is not used in the DRC190.

The Commodore 1541 sends the directory in the same form as a Commodore Basic program (which is different from the DRC190 form) where the line number represents the number of block used by the file. In a line of Basic, the first 2 bytes represent a link to the next line of the program. On a 6502 based machine (such as Commodore or Apple), the link is in low byte, high byte format. On 680x machines, such as the DRC190, the link is stored in high byte, low byte form. The link is address of the next link in the program. During a line search (such as in a GOTO or GOSUB), the program jumps from link to link checking line numbers until the desired line is found. The use of links makes line finding faster, since each line need not be scanned.

The next two bytes of a Basic program represent the line number in low byte, high byte format for a Commodore, or high byte, low byte format in a DRC190 program. The line number is followed by the tokens and ascii characters of the program line, followed by a 00 byte, indicating the end of the line. The 00 byte is followed by the link of the next line. If the link has both bytes 00, the previous line was the end of the program. DRC190 Basic follows the 0000 link with another 00, indicating the end of a zero length line.

The directory command takes the data coming down the serial bus, evaluates it, and displays it. The evaluation consists of throwing out the first two bytes sent (the load address), then evaluating each line as it is sent.

As each line is received, the first two bytes represent the link to the next line. These are thrown out by the DRC190 in the directory command. The next two bytes represent the line number, which represents the size of the file about to be listed, in low byte, high byte form. These two bytes are reversed, converted to decimal, then displayed. Each byte from the serial bus is then sent to the display, to show the name of the file and its attributes, as sent by the disk drive. When a 00 byte is detected, indicating the end of a line, a carriage return line feed sequence is sent to the display. The next two link bytes are checked to see if they are 00, and thrown out if they are not. If they are 00, the directory listing is completed.

On completion of the directory listing, the DRC190 sends an UNT (untalk) command to the disk drives. The UNT command consists of a $5F with ATN* low. This takes all devices off the bus.

A "close file Ø" command is then sent to the drive. This takes the form of MLA8 (my listen address 8), followed by a $EØ sent with ATN* low. Finally, an UNL command is sent.

This description should give you some idea of how communications with disk drives operate. For further information, refer to the bibliography section.

Status Interface

The status interface uses another serial bus, although this is much simpler than the interface to the disk drives. The disk drive interface needs to send and receive variable length messages to and from various devices on the bus. It includes extensive handshaking, since the disk drive cannot be constantly watching the bus (it has to look at the disk now and then). The status panel, on the other hand, transmits and receives constant length messages with no handshaking. The bus requires more wires, but is quite simple.

UØ4 contains an 8 bit shift register that can be used to serially transmit or receive data. In each case, this data is sent or received through the CB2 line. UØ4 sends the shift register clock out on CB1. This is used to drive the shift registers in the status panels.

When status is being sent by the DRC19Ø to a local status display, PB5 is set low, indicating we are doing an output. This causes UØ6A to release STATUS-I-O so that it can be driven by UØ4 CB2. CB1 (the status clock) is programmed low, the desired logic state is put on CB2 (the STATUS-I-O) by the DRC19Ø, then CB1 is programmed high. The status panel receive shift registers capture the data bit on the positive edge of CB1 (which is inverted by UØ5F, causing the status panel shift register to capture the data on the negative edge of SCLK*). This is repeated until 5 bytes (4Ø bits) are sent to the status panel. The first 4 bytes are 32 channels of status. The last byte is the site number of the site the status information was received from. On completion of the transmission of these 5 bytes, LOAD-OUT is pulsed high (pulsing LOAD-OUT* low). The status panel routes LOAD-OUT* through a binary comparator that compares the from site number to the site number the particular panel is to display. If they match, the LOAD-OUT* pulse is passed on to the other receive shift registers, allowing them to latch the data that has been received. The latched data is used to drive the status panel front panel LEDs and drive the status outputs. If the address did not match, the shifted in data is not latched, and is ignored.

The DRC firmware alternates between reading and writing status to the status panel. During the read cycle, a byte of status (8 bits) is read every 1Ø mS. This continues until 12 bytes have been read in (12Ø mS). The DRC is expandable to 96 status lines, numbered Ø through 95. Status number 96 is used to indicate whether a site is in "local". As each byte of status is read in, it is compared with what was received previously. If they are different, a status change flag is set. Once the status read cycle is completed, the DRC checks the status change flag. If there has been a change in any status line, all 12 bytes of the current status are sent to all sites through the internal communications and the modem. If no status change has occured, the status is not transmitted (unless a specific request for current status is received). Once the status transmit portion of the routine is completed, the routine sends 12 bytes of status (one every 1Ø mS) to the local status panel. This status is the latest status received from this site or another site. It is a separate buffer than the transmit buffer described above. After the 12 bytes of status have been sent (lowest number last), a byte of status address is added. This is the site that the status message originated at. Should the received site agree with the site selected by the DRC front panel LCD/keyboard, the message is also retransmitted to the status panel with an address of 1ØØ (decimal). Once the complete status/address message is sent, the LOAD-OUT* line is pulsed low. This line is routed through the address comparator on the status receiver. Should the address match, the status is latched into the LED driver latches and the user port driver latches.

To read the status, the DRC19Ø programs UØ4 to treat CB2 as a shift register input and sets ST-IN-OUT* high, indicating we are about to do an input. UØ6A now routes the incoming status (from the transmitting status panel's transmit shift register) to STATUS-I-O. UØ4 pulses the LOAD-IN line high, causing the transmit shift register in the status panel to latch the current state of all 32 (expandable to 96) status lines. The status is then clocked out of the status panel shift register into the shift register in UØ4 in a manner similar to the shift out. Twelve bytes, representing the 96 status channels are shifted in. These are compared to the last status that was shifted in. If there is a difference, the new status is transmitted to all sites by the DRC19Ø.

STD Interface

Interface to the backplane (the STD bus) is provided by UØ2, UØ3 and UØ7A.

UØ3 compares the address present on the address bus with the board address set up on PØ2 (normally $9ØFØ). If the compared address lines match AND STD-IORQ is low, Board-Sel* goes low, enabling the remainder of the STD bus interface. Note that STD-IORQ is driven by the memory map PROM on the processor board. It is low when the processor is addressing off board I/O devices.

If Board-Sel* is low, UØ2 is allowed to transmit data. The direction of the data flow through this non-inverting transceiver is determined by the STD-R-W* line. If the line is high, the processor board is trying to do a READ, and UØ2 will take the data present on the A side (the board side) and present it to the B side (the STD bus side). If the STD-R-W* line is low, the processor is doing a write. The data present on the STD data bus will be presented to the data lines of UØ4.

If Board-Sel* is low, UØ4 is also enabled. Otherwise, it ignores the bus activity.

UØ7A inverts STD-P2* (the inverted processor phase 2 line, also called the E line) to become Phase2. This is presented to the Phase2 input of UØ4.

During a write to UØ4, the processor sets up the address lines to represent an address of UØ4 ($A8FØ to $A8FF). A few nS later, the memory map PROM on the processor board drives STD-IORQ* low, indicating we are addressing an off-board I/O device. A few nS later, the Board-Sel* output of UØ3 goes low. At the same time the address lines were set up, the STD-R-W* line was driven low by the processor to indicate we are going to do a write.

After the address and R-W* lines were set up, Phase2 goes high. About 125 nS after this, valid data from the processor is presented to the data lines of UØ4. After the Phase2 line has been high for 25Ø nS, it goes low, causing UØ4 to latch the data that was present on its data lines. The result of this write into one of the 16 registers inside UØ4 depends upon which register was written to (determined by the address lines connected to the RS Register Select inputs of UØ4). UØ4 could just present the written data to one of its output ports (port A or port B), or could use the data internally to set a timer, load a shift register, or set up the operation of the timers, shift registers and ports.

A read operation from the processor operates in a similar manner. The only real difference is that that STD-R-W* line is now high, causing UØ4 to drive the data lines instead of receiving data off the lines. These lines then drive UØ2, which then drives the STD bus with the data.

The final portion of the STD-Bus interface is handled by UØ7B and UØ7C. If UØ4 is programmed to generate an interrupt, it pulls IRQ* low when the interrupt needs servicing. This is inverted twice by UØ7B and UØ7C to increase the drive capability to drive the STD bus. Since UØ7 is an open collector inverter, RØ7 is used pull up the lines where necessary.

Subcarrier Transceiver Theory of Operation

Theory of Operation 1444 Subcarrier Transceiver

The 1444 subcarrier transceiver generates and demodulates subcarriers suitable for control and metering on STL and FM broadcast stations. The theory of operation will be covered by page number of the schematic.

Page 1: Subcarrier Modulator

UØ1 is a function generator that is set up to generate a sine wave at the desired subcarrier frequency. RØ1 adjusts the subcarrier waveform symmetry. RØ2 adjusts the triangle to sine wave converter to minimize the distortion of the subcarrier waveform. RØ1 and RØ2 are adjusted for minimum harmonic distortion of the subcarrier carrier, as measured with a THD analyzer or a spectrum analyzer. THD can typically be adjusted to about Ø.9%

Pin 1, the AM input, is grounded, forcing full subcarrier output.

RØ3, RØ5 and CØ2 provide a bias voltage at half supply. CØ2 places this bias voltage at AC ground. RØ4 provides the bias voltage to the input of the output amplifier while pulling that point towards AC ground to adjust the output level (subcarrier injection).

The subcarrier frequency is determined by CØ3, RØ6, RØ7, RØ8 and RØ9. Since RØ9 is small compared with the other resistors, it has little effect on the frequency. RØ6 is used to get close to the desired frequency while RØ7 sets the precise frequency. RØ8 prevents damage to UØ1 should RØ6 and RØ7 be set to zero ohms.

The audio to be carried on the subcarrier (TX-AUDIO) is fed to R1Ø and RØ9. The large division ratio of this voltage divider provides the required low level audio to give the desired deviation of the subcarrier (typically 1 KHz/volt). RØ9 is used to adjust the subcarrier deviation.

CØ4 removes the DC component of the resulting subcarrier.

Page 2: Subcarrier Receive Mixer

The subcarrier receiver or demodulator uses the super-hetrodyne principle. The entire base-band is up-converted to 455 KHz. At 455 KHz, the desired subcarrier is pulled out with a ceramic filter. Using an up-conversion places the image response of the receiver above 455 KHz, above the response available on STLs or FM receivers.

This page of the schematic covers the local oscillator - mixer (or converter) portion of the receiver.

As in the transmit portion of the subcarrier transceiver, UØ2 here generates a sine wave. R14 adjusts the triangle to sine converter in UØ2 to provide the sine wave. The frequency of the oscillator is determined by CØ6, R16, R17 and an AFC voltage from the discriminator on page 4. With no input signal, the AFC voltage is zero, allowing R17 to be adjusted so that the correct local oscillator frequency is measured on PØ5. Note that PØ5 normally has a shorting plug across it. This must be removed before measuring the LO frequency. After the LO has been measured, the jumper is replaced to improve the local oscillator null. Pin 11 of UØ2 is an open collector output of the oscillator signal. R15 provides pull up, making the test signal available on

PØ5 a 12 volt P-P square wave.

The local oscillator frequency should be adjusted to 455 KHz - SCA where SCA is the desired receive frequency. For example, if we wish to receive 11Ø KHz, the local oscillator frequency would be 345 KHz. The mixer uses low side injection to insure that the required frequency is always within range of UØ2. In addition, the AFC circuit is designed with low side injection in mind. If the local oscillator were changed to high side injection without a change in the AFC circuitry, the AFC would push the receiver away from the desired frequency (not a good idea!).

UØ2 includes a balanced AM modulator (one capable of generating double sideband supressed carrier). R11 adjusts the DC bias to the input of the balanced modulator so that there is no output on pin 2 when no subcarrier is being received.

When a subcarrier is received, the AC voltage is analog multiplied by the local oscillator signal in the balanced modulator, resulting in a signal at SCA + LO and one at LO - SCA where SCA is the received subcarrier frequency and LO is the local oscillator signal. The SUM signal ends up at 455 KHz, while the difference signal ends up substantially below 455 KHz. CØ5 AC couples the received subcarrier into the balanced modulator.

R12 and R13 provide the required +supply/2 bias to the input of the output amplifier of UØ2.

The mixer output appears on pin 2 of UØ2. The DC component is removed by CØ8.

Page 3: IF Amplifier

The mixer output from UØ2 on page 1 (with the DC component removed by CØ8) is applied to the IF filter YØ1 through R18. R18 along with the 6ØØ ohm output impedance of UØ2 provides the required driving impedance for YØ1. YØ1 pulls the subcarrier out of the up-converted base-band while rejecting the rest of the base-band.

QØ1 along with the associated resistors form the IF amplifier. The output of the IF amplifier is coupled to the discriminator through CØ9.

DØ5 and DØ6 form a clipper or limiter so that the IF level does not vary with received signal level. When in saturation, the collector voltage on QØ1 appears something like a half wave rectified sine wave (the negative half) with an amplitude of 1.3 volts peak to peak. If the subcarrier receive level is dropped to 15Ø mV peak to peak, the voltage on QØ1 drops to 5ØØ mV peak to peak. With these typical figures, the mixer conversion gain and IF amplifier gain can be checked.

Page 4: Discriminator

The discriminator is formed by R23, R24, DØ1, DØ2 and YØ2. The DC (or audio) voltage on DISCRIM-IO varies with the frequency of the IF signal applied to DISCRIM-IO. R25 and C11 form a low pass filter, allowing DC and audio to pass while rejecting the IF frequency.

The output of the low pass filter is amplified by 1Ø by UØ3A. The output of UØ3A is coupled through C13 to the subcarrier demodulator output. The demodulated output is sent through a low pass filter formed by R32 and C16 to further remove any IF components.

Subcarrier Transceiver Theory of Operation

The output of UØ3A is also filtered through R29 and C12 to remove the audio while leaving a DC voltage proportional to the frequency deviation from the center of the discriminator. This voltage is bufferred by UØ3B and applied to the local oscillator as an Automatic Frequency Control signal, insuring that any drift in the local oscillator frequency will not cause loss of the received subcarrier.

Page 5: Power Supply Filters

R3Ø, R31, C14, and C15 filter the power for the subcarrier transceiver insuring that power supply and processor noise do not interfere with the operation of the subcarrier circuitry.

Direct Connect Modem Theory of Operation

Theory of Operation 1445 Direct Connect Modem

The 1445 direct connect modem allows the DRC19Ø to communicate with data terminals or computers over dial up telephone lines. The board provides an auto-dial, auto-answer, 3ØØ/12ØØ bit per second modem. It also provides an RS-232 port and several TTL level I/O lines.

Page 1: DUART

This page includes the STD bus interface circuitry and the Dual Universal Asynchornous Receiver Transmitter.

UØ2 drives BOARDSEL* low if the STD-IORQ* line is low and the address on STD-A4 through STD-A7 matches the selected address set up on ADDRSEL4 through ADDRSEL7, as set up on PØ1. RØ1 provides pull-ups to +5-VOLTS when a pair of pins on PØ1 is left open.

BOARDSEL* enables UØ1 (the data bus transceiver) and UØ3 (the DUART) when the processor is addressing this board. STD-R/W* selects the direction of the data transfer. If STD-R/W* is high, the processor is doing a read and UØ1 inputs data on DØ to D7 (the on-board data bus) and outputs it to STD-DØ to STD-D7 (the STD data bus). This takes data from the DUART and sends it to the processor through the STD bus.

If the STD-R/W* line is low, the processor is doing a write and the data from the STD bus is sent to the data inputs of the DUART.

UØ3 is the DUART. STD-AØ to STD-A3 (the least significant address lines) are used to select registers in the DUART. STD-WR* is driven low by the processor board when it wants to write data to an external device (such as the DUART). STD-RD* is driven low by the processor when it wants to read data from an external device, such as the DUART. The STD-RD* and STD-WR* signals are ignored unless BOARDSEL* is low, indicating the processor wants to talk to this particular chip.

UØ7A inverts the STD-RES* signal, creating the active-high RESET signal, which resets the registers in UØ3 on power up.

The IRQ* output of UØ3 goes low when UØ3 wishes to interrupt the processor. This is doubled inverted and bufferred by UØ7B ana UØ7C and put on the STD bus. UØ7 is an open collector device, so RØ3 provides required pull ups. The current DRC firmware does not use the interrupt capability of the modem card.

YØ1 and CØ1 are used by the baud rate oscillator in UØ3. UØ3 contains programmable dividers that use the 3.6864 MHz signal to provide the different available communications rates.

PØ4 allows several of the DUART parallel input and output lines to be brought outside the DRC19Ø for user applications.

Direct access to these lines is not supported by Basic, other than through the use of PEEK and POKE statements.

These I/O lines are from the 2681 DUART on the modem card. The inputs and outputs are at TTL levels. These lines can be used as status/control lines, although precautions must be taken to protect the I/O lines from voltage transients. In addition, these lines are not supported direclty by Basic. They must be accessed using PEEK and POKE statements. As the firmware of the DRC19Ø is revised, the address of the direct connect modem DUART may vary. To find the address, refer to the symbol table in the Firmware section of the manual. Listed there is the hexadecimal address of DCMDM (Direct Connect Modem). Convert this to decimal and substitute it in the below listed PEEK and POKE statements.

To read the state of an input line from Basic, use the expression (2^N) AND PEEK(DCMDM+13). N is the bit number (N=6 for input 6). The expression will have the value Ø if the input was low, and a non-zero value if the input was high.

To program an output pin high, use the statement POKE DCMDM+14, (2^N). To program an output low, use the statement POKE DCMDM+15, (2^N). For further information, see the 2681 data sheet.

As an example, assume that DCMDM=9ØEØ hexadecimal, or 37Ø88 decimal. To program an output bit on PØ4, use the statements listed below:

Output Bit	Program High/Low	Statement
1	Low	POKE (37Ø88+14), 2
1	High	POKE (37Ø88+15), 2
2	Low	POKE (37Ø88+14), 4
2	High	POKE (37Ø88+15), 4
3	Low	POKE (37Ø88+14), 8
3	High	POKE (37Ø88+15), 8
4	Low	POKE (37Ø88+14), 16
4	High	POKE (37Ø88+15), 16
5	Low	POKE (37Ø88+14), 32
5	High	POKE (37Ø88+15), 32
6	Low	POKE (37Ø88+14), 64
6	Hign	POKE (37Ø88+15), 64
7	Low	POKE (37Ø88+14), 128
7	High	POKE (37Ø88+15), 128

To read the input lines from PØ4, use a PEEK(DCMDM+13). If the symbol table in the firmware theory of operation section indicates DCMDM is 9ØEØ hexadecimal, 37Ø88 decimal, then you should do a PEEK(37Ø88+13). ANDing the result of the peek with a mask will yield a Ø result if the selected bit is low, and a non-zero result if the selected bit is high. Examples appear below.

```
IF (PEEK(37Ø88+13) AND 2) <> Ø THEN DISPLAY "Input 1 is high"
IF (PEEK(37Ø88+13) AND 4) <> Ø THEN DISPLAY "Input 2 is high"
IF (PEEK(37Ø88+13) AND 8) <> Ø THEN DISPLAY "Input 3 is high"
IF (PEEK(37Ø88+13) AND 16) <> Ø THEN DISPLAY "Input 4 is high"
IF (PEEK(37Ø88+13) AND 32) <> Ø THEN DISPLAY "Input 5 is high"
IF (PEEK(37Ø88+13) AND 64) <> Ø THEN DISPLAY "Input 6 is high"
```

Input Ø of the DUART is driven by the HI-SPEED* output of the modem module. This line is used by the firmware to determine whether the modem is in the 12ØØ or 3ØØ bit per second mode.

Output Ø of the DUART drives the SW1 input of the modem module, determining the number of bits sent. On reset, SW1 is high, causing the modem to sent 8 data bits (7 bits of ASCII with a mark parity bit).

Direct Connect Modem Theory of Operation

Page 2: Modem & RS232 Interface

This page shows the modem module and the RS232 interface.

UØ4, the modem module, operates on +5 and -5 volts. RØ2 and DØ1 derive the -5 volts from the -12 volt supply. The serial output of the DUART is sent to the modem, and the serial output of the modem is sent to the DUART. Unused modem lines are left open. The HI-SPEED* output of the modem reflects the speed of the current call. This is sent to the DUART. Note, however, that HI-SPEED is changed only during a call. The DRC19Ø firmware initializes the modem at 3ØØ bits per second on power up. It then changes the modem speed in response to MDMSPD statements. It also changes the speed of the DUART serial port in response to a control-N W status message from the modem, indicating it has received a call at the wrong speed, and is about to change speed.

The modem RESET line is pulsed high on system reset, resetting the modem to its default conditions. The telephone line connects to the tip and ring connections of PØ3, which appears on J21 on the rear panel.

The second serial port of the DUART is converted to RS232 levels by UØ5 and UØ6. This appears on the rear panel at J23.

For further information on the modem, see the modem manual, reprinted in the back of this manual.

Direct Connect Modem Programming

Direct Connect Modem Operation

The H&F 1445 Direct Connect Modem card is built around the Cermetek CH177Ø modem module. This FCC approved module provides 3ØØ/12ØØ bit per second modem functions meeting Bell 1Ø3 and Bell 212 standards. It also provides auto anwer and dialing. Dialing is available with either tone or pulse. The manual on the CH177Ø is printed following this page.

The CH177Ø communicates with the DRC19Ø through a serial port on the modem card. Data is sent to the modem with a PRINT #2, statement. Commands are sent to the modem with a PRINT #2, CHR$(14); statement. The CH177Ø looks for CHR$(14), then treats the following characters as a command. The PRINT statement should not be terminated with a comma or semicolon so that a carriage return line feed sequence is sent to the CH177Ø to terminate the command.

Data can be received from the CH177Ø using INPUT#2, statements, or INKEY$(2) functions. Note that the INPUT#2 statement will wait for a carriage return to be received, which can prevent the DRC19Ø from taking other required actions. It is therefore suggested that most modem input be done using the INKEY$(2) function, and string inputs built using this function. After long periods of inactivity, the string building routine can be terminated.

The CH177Ø sends unsolicited status messages on a change in its status. These are preceeded by a control-N. The DRC19Ø firmware traps these messages and stores the latest one in MDMSTAT$. The line feed carriage return sequence following the status message is not trapped, and reaches the INKEY$(2) or INPUT#2 function or statement. For this reason, it is suggested that upon noting that a call has been answered, the receive buffer be cleared by executing a loop containing INKEY$(2). Otherwise, a remaining carriage return may terminate an input statement.

MDMSTAT$ has a character in it with the following meaning:

D - Modem has disconnected due to remote disconnect or disconnect command

R - The modem has sensed the line it is connected to is ringing

A - The modem has answered an incoming call or has sensed that an outgoing call has been answered

N - An outgoing call has not been answered

Note that MDMSTAT$ holds the last received status character. It may hold an R if the line rang once, but is not currently ringing. It may hold an N indicating that the last call was not answered, but there is no status on the current call yet. Most MDMSTAT$ characters are due to unsolicited status messages received from the CH177Ø. The exceptions are that when an END command (control-N E) is sent to the modem, MDMSTAT$ is changed to a D, even though the CH177Ø does not return a status message. In addition, the control-N W message does not ever appear in MDMSTAT$, INKEY$(2) or INPUT#2. This message indicates that the CH177Ø has sensed that the remote modem is at the wrong speed, and the CH177Ø is changing speed. This message is trapped by the DRC19Ø firmware, and changes the speed of the serial port driving the CH177Ø. It also updates the variable MDMSPD.

It is suggested that prior to starting a call an END command be sent to the modem, forcing MDMSTAT$ to D. You can then check for N or A for the current call.

MDMSPD indicates the current speed of the modem in hundreds of bits per second. It can be either 3 (for 3ØØ BPS) or 12 (for 12ØØ BPS). MDMSPD can be read or written to. An example of a read is:

```
IF MDMSPD=3 THEN GOTO 1234    :REM Skip long message if low speed
```

A write to MDMSPD changes the speed of the serial port running the CH177Ø and runs the modem through a speed change training sequence. This is typically used to originate a call to a specific terminal at a specified speed. An example of such a write is:

```
MDMSPD = 12                   :REM Change to high speed
```

A summary of the modem commands is listed below. For more detail, see the CH177Ø manual, reprinted following this section.

ANSWER PRINT #2, CHR$(14); "A"

Force the modem into answer mode. This might be used to change a voice call into a data call (answer a line that is not ringing), or to answer a ringing line before the ring counter reaches the programmed value.

BREAK PRINT #2, CHR$(14); "B 9"

Send a break (long space) for a multiple of 25Ø mS. The example sends a break for 9 25Ø mS periods (2.25 seconds). Valid numbers to use in the break command are 1 to 9.

COUNT PRINT #2, CHR$(14); "C 9"

Sets the ring counter. In the example, incoming calls will be answered in after 9 rings, while outgoing calls will be terminated if not answered in 13 rings (n+4). The COUNT command can use a number between Ø and 9. If Ø is used, the modem will not auto-answer calls. On power up, count is set to 2.

DIAL PRINT #2, CHR$(14); "D 'TB(8Ø5)541-Ø2ØØ'"

This example will originate a call using tone dialing to H&F. The TB indicates use of tone dialing. ()-@ are place holding characters and are ignored by the modem. No spaces are allowed in the dialing string (other than the required one after the D). If the call is answered by a modem within the programmed number of rings, MDMSTAT$ will change to A. If the call is not answered in the required number of rings, MDMSTAT$ will change to an N.

```
PRINT #2, CHR$(14); "D 'PB(8Ø5)541-Ø2ØØ'"
```

This example will originate a call using pulse dialing.

```
PRINT #2, CHR$(14); "D 'TB541-6116B765538B(8Ø5)541-Ø2ØØ'"
```

This example places a call through an alternate long distance carrier. The B characters in the dialing string cause a 2 second pause, allowing for answer of the access line, and then allowing for account number verification. Up to 32 digits may be included in a number string.

```
PRINT #2, CHR$(14); "D 'TB541-Ø2ØØ' 9"
```

This example will use tone dialing to call H&F and will retry 9 times, or until the call is answered. The number of retrys can be between Ø and 9. There is a 2 second pause between retries.

```
PRINT #2, CHR$(14); "D 'TB541-Ø2ØØZ'"
```

This example puts the modem in the sleep mode (zzzz) after placing the call. The modem stays off hook and does not bring up its modem carrier. This may be useful to auto-dial a voice call.

```
PRINT #2, CHR$(14); "D"
```

Redial the last number dialed and wait for answer.

```
PRINT #2, CHR$(14); "D 9"
```

Similar to above, but the call is retried 9 times. The number of retries can range from Ø to 9. There is a 2 second pause between retries.

END

```
PRINT #2, CHR$(14); "E"
```

End or terminate the current call. This puts the modem on-hook and sets MDMSTAT$ to D.

ORIGINATE

```
PRINT#2, CHR$(14); "O"
```

Force the modem off hook and into the originate mode. This could be used to convert a manually dialed voice call to a data call with the modem in originate mode (instead of answer mode). If no answer tone is heard from the remote modem within 17 seconds, MDMSTAT$ is changed to N. If an answer is heard within 17 seconds, MDMSTAT$ becomes A.

This list of modem commands should cover most requirements. There are a few other commands listed in the following documentation from Cermetek, but they must be used carefully, as the DRC19Ø firmware counts on the modem to be programmed in a certain manner. For example, using the NEW command to change the command character from control-N will prevent MDMSTAT$ or MDMSPD from being updated when required, as the DRC19Ø firmware traps control-N messages. The PROGRAM command can be used, as outlined in the Cermetek manual, but unsolicited status messages should not be disabled. The QUERY command can be used, although the character immediately following the command character (control-N) will be trapped by the DRC19Ø firmware (and thrown out, since it is not a valid character for MDMSTAT$) and not appear in the character string sent to INKEY$(2) or INPUT#2. RESET should not be used, as the DRC19Ø firmware goes through a power up initialization sequence to program the CH177Ø, and this will not be repeated when sending a RESET. The TEST mode can be run to check the modem operation, as outlined in the CH177Ø manual. Use of the UNLISTEN command is not suggested. The ZZZ command can be used, if desired.

The manual provided by Cermetek on the CH177Ø modem module is reprinted (with permission) starting on the next page. Note that our agreement with Cermetek requires modem problems be referred to H&F rather than Cermetek.

Cermetek
microelectronics

DATA MANUAL

CH1770

BELL 212A-TYPE 110/300/1200 BPS
INTELLIGENT MODEM COMPONENTS

1985 Cermetek Microelectronics, Inc. DOC #82 Printed in USA

SUMMARY OF CONTENTS

Contents | Page

1.0 FEATURES

- Small Size, PCB mount
  2.54"x3.74"x.75"
- 110/300/1200 bps Operation, Bell 212A and 103 compatible
- FCC Part 68 Registered Telephone Line Interface (DAA)
- Serial Modem Command Interpreter
- Intelligent Command Protocol
- Auto/Manual Dialing
- Auto/Manual Answer
- Auto Speed Select
- Auto Parity Select
- 3-Dialing Procedures
  (Dial Last, Immediate, Repeat Dial)
- DTMF and Pulse Dialing
- Diagnostic Test Mode
- Voice/Data Operation
- Asynchronous Operation
- TTL Host Interface Levels With RS-232C Type Lines
- Power: +/-5V

2.0 GENERAL DESCRIPTION

2.1 Configuration

The CH1770 is a Bell 212A-type modem component that uses the latest in LSI technology to implement a highly intelligent 110/300/1200 bps modem component in less than 9 square inches.

The CH1770 employs resident firmware to control every function of the modem. The CH1770 masks this firmware directly onto its resident controller.

2.2 Interface

The CH1770 interfaces to the telephone line through a built-in FCC registered data access arrangement (DAA) that directly connects to the telephone line through a user supplied RJ-11C jack. Because the DAA partially powers itself from the telephone line's loop current, telephone line connection must be made for correct DAA performance. Terminating TIP and RING lines with just a 600 ohm resistor for testing purposes is not adequate since no loop current is provided. An 18 volt D.C. floating power supply in series with a 600 ohm resistor will provide a suitable test termination.

Since the CH1770 is entirely controlled through the exchange of asynchronous commands on its serial data lines, TXD and RXD, it can be easily software controlled through the host's UART/USART without requiring additional serial or parallel control ports. Commands may be sent at either 110, 300 or 1200 bits per second (bps) using an ASCII format. Status is returned serially to the host using terse ASCII messages, making them easily decoded by the host's application software. The serial data interface is implemented using RS-232C lines but at TTL levels.

2.3 Operation

The CH1770 supports 13 different host commands, enabling such functions as:

- Auto-dialing
- Auto-answer
- Echo or no echo of commands
- Modem test diagnostics
- and many more

Dialing can be commanded to use either DTMF tone or rotary pulse dialing. Pause characters can be used to direct the CH1770 to pause during dialing. This enables the CH1770 to dial through PBX's, which commonly require the dialer to pause for an outside line, after dialing 9, before continuing to dial the rest of the number.

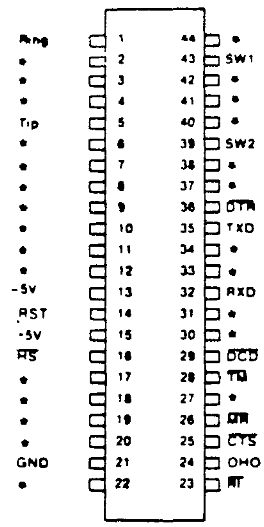
2.4 Data Speed and Parity

The CH1770 automatically adapts to the host's speed (110/300/1200 bps) and parity (odd, even, mark, and space) by using a simple learning sequence. If, however, a remote modem calls the CH1770 at a different speed and automatic speed adaption is enabled in the program register, it will automatically adapt to the remote modem's speed. The selected speed is indicated to the host at pin HS (high speed), and through a terse status message on RXD at the old speed.

3.0 MECHANICAL SPECIFICATION

3.1 Pin Configuration

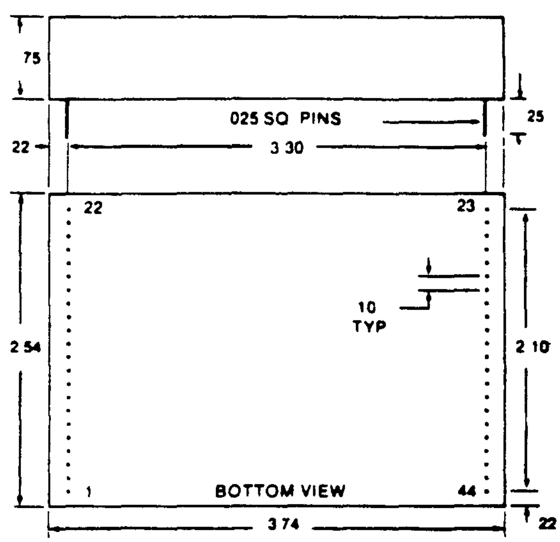
CH1770



NOTES: * Indicates a factory test point.
Make no connection to these pins.

3.2 Physical Demensions

CH1770



4.0 PIN DESCRIPTION

Telephone Line Interface:

RING; pin 1 TIP; pin 5	: direct connects to the telephone line's RING and TIP leads through a user-supplied RJ-11C telephone line jack.
OHO; pin 24	: OFF HOOK output. A high indicates that the modem is off hook. Drive Capability: 3 LSTTL Loads
$\overline{RI}$; pin 23	: ring indication output. A low level indicates that the telephone line is ringing. The modem will answer at the end of the number of ring signals which is set in the "count" command (see Section 7.0). At power up the modem auto-answers after the second ring. A ring signal must be greater than 100 msec. in duration with greater than 500 msec. between rings. Drive Capability: 3 LSTTL Loads

POWER

+5V; pin 15	: +5v power supply input
-5V; pin 13	: -5v power supply input
GND; pin 21	: signal and power ground return

MODE CONTROL LINES:

SW1; pin 43 Input SW1 is used by the CH1770 to set the component's serial data format as shown below:

SW1	Serial Format
LOW	9 data bits (including parity)
HIGH	8 bits (including parity) or 7 bits with 2 stop bits

The CH1770's asynchronous data format requires one start bit and at least one stop bit. For 7 bit data selection, 2 stop bits minimum are required. The number of data bits selected includes parity.

Input Load: 1 LSTTL Load.

SW2; pin 39 : Input SW2 is used to set the operation of the modem interface lines $\overline{CTS}$, $\overline{DSR}$, and $\overline{DCD}$. If SW2 is asserted low, these lines are asserted to the state of the $\overline{DTR}$ input, pin 36. If SW2 is forced high, normal RS-232C line signal sequencing is supported (see CH1770 Handshake Timing Diagram).

Input Load: 1 LSTTL Load.

SERIAL HOST INTERFACE:

TXD; pin 35 : serial transmit data input. Marking or a binary 1 condition is transmitted when high is asserted.

Input Load: 1 LSTTL Load

RXD; pin 32 : serial receive data output. Received marking or binary 1 condition is indicated by a high output.

Drive Capability: 2 LSTTL Loads

$\overline{MR}$; pin 26 : data set ready output. A low output on this pin indicates the modem is OFF HOOK in the data mode. If $\overline{MR}$ is set to follow $\overline{DTR}$, this pin will indicate when $\overline{DTR}$ has been asserted ON.

Drive Capability: 2 LSTTL Loads

$\overline{CTS}$; pin 25 : clear to send data output. When this signal is set low the CH1770 has set up the data call and is ready to transmit data.

Drive Capability: 2 LSTTL Loads

$\overline{DCD}$; pin 29 : receive data carrier detect output. When this output is set low, the received data carrier is present on the telephone line.

Drive Capability: 2 LSTTL Loads

$\overline{DTR}$; pin 36 : data terminal ready input. This input must be set low before the modem can answer or initiate calls. Once a call has been established this line can be used to disconnect the call by setting $\overline{DTR}$ high for greater than 50ms.

Input Load: 1 LSTTL Load

$\overline{TM}$; pin 28 : test mode output. This output is set low whenever the CH1770 is placed in the analog loopback test mode.

Drive Capability: 2 LSTTL Loads

$\overline{HS}$; pin 16 : high speed select output. A low on this level indicates the CH1770 is operating at 1200 bps. If $\overline{HS}$ is high, the modem is operating at 110 or 300 bps.

Drive Capability: 3 LSTTL Loads

MISCELLANEOUS SIGNALS:

RST; pin 14 : CH1770 reset input. A high applied on RST resets the modem to the idle state and asserts the phone line off hook. At power up, this pin must be asserted high for a minimum of 10 ms after the 5 volt supply has reached its operating region.

Input Load: 2 LSTTL Loads

Note: This pin internally has a 10K ohm resistor connected to GND and a 10uf capacitor connected to +5 volts.

5.0 TYPICAL APPLICATIONS

Because of the CH1770's small size and component configuration, it is ideally suited to integrate modem communications into a host data product. The addition of communications to such products as CRT terminals, personal or business computers, or workstations, can vastly expand the capability of the product, allowing it to support electronic mail, data base access, remote diagnostics, distributed networking and other such functions.

As can be seen in Figure 1, the CH1770 provides an efficient solution to 212A-type modem integration. Since its serial host interface supports RS-232C type lines at TTL levels, the CH1770 directly interfaces to virually any UART. All that is needed to complete the modem integration function is to provide power to the CH1770 and connect it to the telephone line through a modular telephone jack (RJ-11C-type). The telephone line interface is FCC registered and a registration label is included with each CH1770 for application to the outside of the host product. No recertification is necessary.

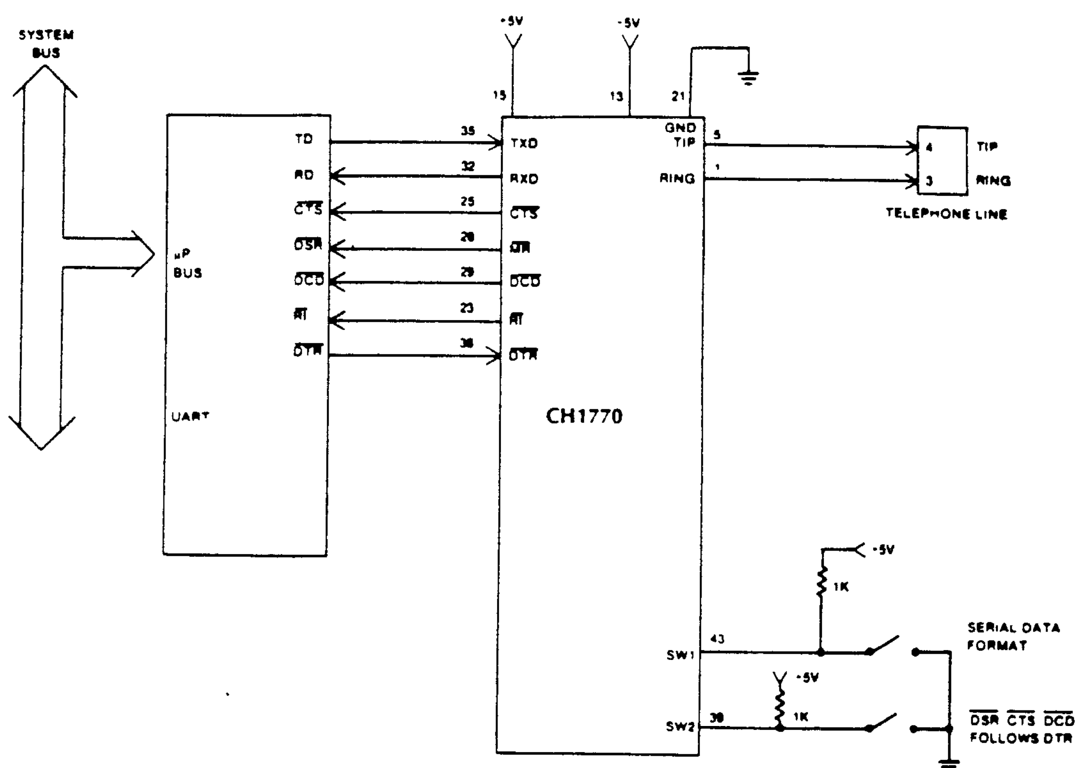
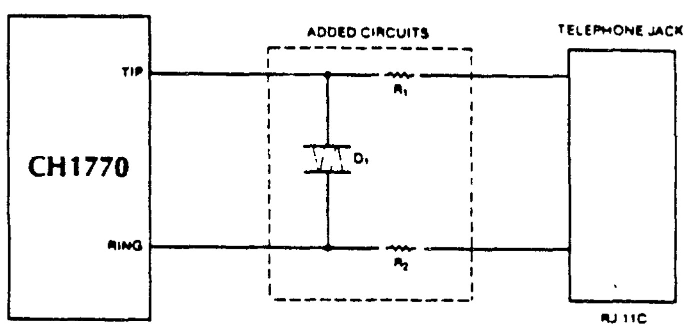


FIGURE 1: TYPICAL INTEGRAL 212A-TYPE MODEM

The telephone interface type is permissive, which means the product user is permitted to make the telephone line connection through a modular RJ-11C-type telephone jack. The FCC, however, requires that the end product user be provided with the installation rules and regulations from their Part 68 so it is necessary to have such information as presented in SECTION 8.0 (FCC GENERAL INFORMATION) in the end product's User's Manual.

The CH1770 can additionally be approved for Canadian telephone line connection. This must be done after the modem is installed in the host. The host system must then be submitted to Canadian DOC (Department of Communications) for approval. Because the DOC requires additional protection, the following additional telephone line interface circuitry, (shown in the dotted box), is needed. This circuitry is optional for FCC Part 68 registration in the U.S.A. (see Figure 2).



WHERE R_1, R_2 = 1Ω ¼W CARBON FILM RESISTORS
D_1 = 250V VARISTOR (PANASONIC ERZ-C10DK391)

FIGURE 2: ADDITIONAL CIRCUITRY NEEDED FOR CANADIAN DOC REGISTRATION

The integral modem of Figure 1 is controlled by the host product through the exchange of serial asynchronous commands, as detailed in SECTION 6.0 (MODEM CONTROL). Not all the CH1770 host interface lines need to be used. All that is needed by the CH1770 is transmit and receive serial data: TXD and RXD. All the other lines can be left unconnected, except $\overline{DTR}$ which must be low or ON before the modem can operate. This allows the CH1770 to adapt to virtually any UART environment.

The SW2 input allows further specification of the operation of the interface lines: $\overline{CTS}$, $\overline{DSR}$ and $\overline{DCD}$. When SW2 is asserted high, these lines follow the normal EIA-RS-232C specified handshake format. When SW2 is asserted low, however, $\overline{CTS}$, $\overline{DSR}$ and $\overline{DCD}$ follow the state of the $\overline{DTR}$ input. This unique option allows the CH1770 to operate with the most stubborn "smart" terminals. Many times "smart" terminals (and also the IBM PC) insist on $\overline{CTS}$, $\overline{DSR}$ and $\overline{DCD}$ all to be ON before they will enable the serial data interface. After a data call is set up, all of these lines are indeed ON so serial communication can take place. Since none of these lines are ON before a call is set up, the host's serial data channel is disabled, therefore making it impossible to send serial commands to the CH1770 to auto-dial, for example. This is also a common problem experienced by intelligent standalone modems. The CH1770 option to make $\overline{CTS}$, $\overline{DSR}$ and $\overline{DCD}$ follow $\overline{DTR}$ solves this problem.

The CH1770 can also implement a stand alone 212A-type modem. Figure 3 displays the minimum configuration modem. The entire standalone modem consists only of the CH1770, some RS-232C level converters, and LED an display driver.

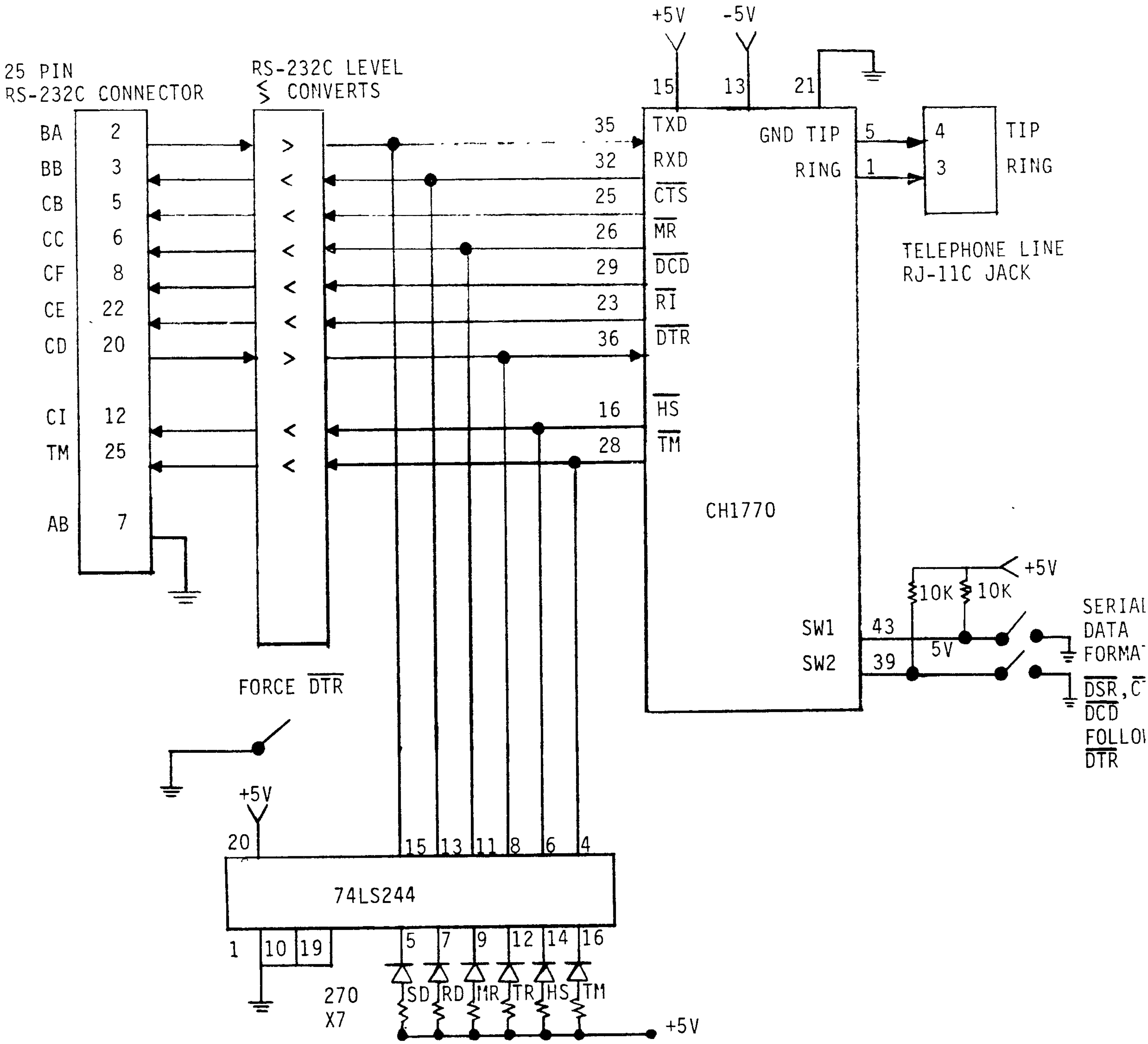


FIGURE 3: INTELLIGENT STANDALONE 212A-TYPE MODEM
(Minimum Configuration)

6.0 MODEM CONTROL

The CH1770 supports serial asynchronous communication. Before a call has entered the data mode, the CH1770 enables a built-in asynchronous command interpreter that allows the host product to issue modem commands serially over the transmit data line, TXD. Similarly, the CH1770 returns its status to the host over the receive asynchronous data line, RXD.

If the CH1770's $\overline{\text{DTR}}$ line is asserted ON (low) and no call is in progress, the command interpreter LISTEN's to data sent by the host in an effort to decode a command. If a command is recognized, it is immediately executed and a completion status is returned to the host.

After a data call is in progress, modem commands and data are sent over the same interface. It is therefore important for the CH1770 to understand when it should interpret the serial transmit data for commands. Use of the UNLISTEN command allows this discrimination.

6.1 Choosing Speed and Parity

The host can set the speed and parity settings of the CH1770 by sending it a four (4) character training sequence. The current speed is indicated by the CH1770 on output pin, $\overline{\text{HS}}$.

For the CH1770 to adapt to parity or speed it must first be idle. It cannot be in the midst of a data call or currently executing a command.

The $\overline{\text{HS}}$ pin is valid only during a connection. This pin is normally only used after answering a call since the modem adapts to the speed of the calling remote modem which is unknown to the local host.

The CH1770 has both a high and a low speed data channel. The high speed channel is set at 1200 bps, whereas the low speed channel is either 110 or 300 bps.

To set just the speed of the modem (110, 300, or 1200 bps), the host should send the CH1770 a <space><space> sequence. This will allow speed selection of either 110, 300, or 1200 bps. If both speed and parity are to be changed, the following sequence should be sent: <space><space>XY. X and Y must be upper case, and <space> is the character sent when the keyboard's space bar key is typed (decimal ASCII code 32).

Each training character sent must be followed by a 200 millisecond minimum pause where no characters are sent.

TRAINING CHARACTER	DESCRIPTION
<space><space>	adapt to host speed
<space><space>XY	adapt to host speed & parity

The CH1770 uses the trained speed to originate calls. On answer calls, however, the CH1770 adapts to the remote modem's speed. Answer speed selection is made between either high speed (1200 bps) or low speed (110 or 300 bps). The selection of either 110 or 300 bps in the low speed channel is controlled by the last low speed trained. For example, if the CH1770 had previously been trained to 110 bps, the modem will auto-select the answer speed at either 110 or 1200 bps. If 300 bps was the last trained low speed, as is the default state at power up, the modem will auto-answer select between 300 and 1200 bps.

6.2 Serial Line Protocol

All commands issued to the CH1770 are encoded in ASCII and are preceded by a single command character, <com>. Each command line is terminated by a carriage return, <CR>. At power-up <com> defaults to control N (decimal ASCII 14). It may be redefined, however, using the NEW command (see Section 7.0).

Multiple commands can be placed on one line separated by commas. The commands themselves consist of the command character followed by the command word, a delimiter, all arguments, and then the closing carriage return or comma. The maximum length of any command line, however, is 40 characters. If multiple commands are issued on a single line, only the first command shall be proceeded by a command character.

Two examples of commands would then look like the following:

```
<com>DIAL 'TB(408)555-1010'<CR>
<com>DIAL 'PB(408)555-1010',QUERY<CR>
```

In both cases, the notation <com> is equivalent to the single command character. The delimiter separating the command from the arguments is always a space. Only the first character of the command itself is significant. All remaining characters are ignored up to the first space following the command.

If the argument is not given in the command, the command will assume the default value as the argument. The arguments are all ASCII numbers and/or characters. The numbers themselves are ASCII encoded hex ('0' - '9' and 'A' - 'F').

Commands can be aborted while in progress by sending the CH1770 another command. The CH1770 will abort the current command upon receiving the new command's <com> character and then begin executing the new command after receiving the complete command line.

The command character itself can be transmitted by sending it twice in a row:

<com><com>

This would send the character a single time, if the modem is in the middle of a data call. There are two other methods of transmitting the command character. The first is to change the command character to another character and then transmit the former command character. The second way is to place the modem in the UNLISTEN mode and then transmit the character.

The CH1770 absorbs all commands without sending them on through to the telephone line.

The CH1770 returns its status to the host over the receive data line, RXD. All status messages are framed as shown below:

<com><status character><LF><CR>

A command character, <com>, precedes each message to let the host know that this is a status message and not data from the remote modem.

Two types of status messages exist; 1) Solicited, and 2) Unsolicited. Messages that result from the execution of a command are called solicited. These messages generally provide information about command completion.

Unsolicited messages result from external events such as telephone line ringing, telephone line hang-ups due to loss of carrier, and auto-answer line connection. On power-up or after execution of a RESET instruction, the unsolicited status messages are disabled. They may be enabled through the use of the PROGRAM command (see Section 7.0). Exact status message format is detailed in SECTION 7.0 (DETAILED COMMAND AND STATUS DESCRIPTION). The symbols <CR> and <LF> represent 'carriage return' and 'line feed,' respectively.

6.3 CH1770 Command Summary

COMMAND	DESCRIPTION
<com>Answer<CR>	force off-hook and answer call
<com>Break n<CR>	send break n x 250 msec
<com>Count n<CR>	ring and ringback counter 0 ignore ring signal, no auto answer 1-9 answer after n rings give up dialing after n+4 ringback signals
<com>Dial s<CR>	dial last, immediate, or until answered
<com>End<CR>	hang up
<com>New n<CR>	set new value of command character <com> to n
<com>Originate<CR>	force OFF HOOK and enter originate data mode
<com>Program n<CR>	set internal modem options
<com>Query<CR>	return modem status
<com>Reset<CR>	reset modem options to defaults
<com>Test n<CR>	start/stop the modem test
<com>Unlisten n<CR>	set CH1770 to LISTEN or UNLISTEN to commands during data transmission
<com>Zzzz<CR>	make modem quiet

6.4 CH1770 Status Summary

<com>A<LF><CR>	data call answered
<com>D<LF><CR>	modem disconnect
<com>N<LF><CR>	no answer or command execution failed

<com>R<LF><CR>	ring signal received
<com>W<LF><CR>	modem answer but host is at wrong speed
<com><LF><CR>	command complete acknowledgement
<com>?<LF><CR>	command entry error
<com><DIALED NUMBER><LF><CR>	number dialed status
<com><$H_1 H_2$><LF><CR>	$H_1 H_2$ represent hex status of the program register

6.5 Disconnect Sources

The CH1770 can be disconnected from a number of different sources once a data call has been established. The following events will disconnect a call:

- received long space (optional)
- END command
- $\overline{\text{DTR}}$ asserted off (high)
- RESET command
- hardware reset, pin 14

7.0 DETAILED COMMAND AND STATUS DESCRIPTION

To concisely describe the CH1770's commands and status messages, a few symbols will be used as defined below:

SYMBOL	DESCRIPTION
<com>	command character. Defaults to control N at power-up, but may be set to another character with the NEW command.
<LF>	line feed
<CR>	carriage return
[ ]	optional parameter
$[\ \ ]^n$	optional parameters that may occur 0 to n times
n...m	specifies the inclusive set n through m
<SP>	space
<letter>	A...Z a...z
<command>	command including argument if needed
<n>	number
<quote>	single quote or apostrophe
<number>	dialed number string

Each CH1770 command follows the following syntax:

<com><command>[,<command>]<CR>

Single or multiple commands may be given to the CH1770 as described in SECTION 6.0 (MODEM CONTROL) but each command line must fit within the 40 character command buffer. If this buffer is exceeded, an entry error is returned to the host:

<com>?<LF><CR>

The following command description will detail the correct syntax, function and an example for each command.

ANSWER

SYNTAX: <com>A[NSWER]<CR>

FUNCTION: Takes the CH1770 OFF HOOK, pauses nominally 2 seconds, then sends answer tone (2225Hz) on the telephone line.

If after nominally 17 seconds the originating modem has not completed the handshake connect sequence, the call is aborted, returning the status:

<com>N<LF><CR>

If the calling modem completes the handshake connect sequence and is at the same speed as the local CH1770, the following status is returned to the host:

<com>A<LF><CR>

If the handshake sequence is completed but the calling modem is at a different speed (110/300 or 1200 bps), the CH1770 returns the following status at the host's old speed:

<com>W<LF><CR>

then switches to the other speed. The CH1770 can be optioned to disable automatic speed, however. After returning the W status it maintains its speed setting and waits for a host command, such as END. This enables the host, if unable to switch speeds, to command an END to the connection before the speed switch is made (see the PROGRAM Command).

The same CH1770 operation occurs if a call is auto-answered, except the answer sequence is initiated automatically by telephone line ringing instead of from the ANSWER command. The same status messages result but because the answer operation was not initiated by the local host, these messages are classified as unsolicited and therefore only occur if unsolicited status messages are enabled (see the PROGRAM Command).

EXAMPLE: <com>ANSWER<CR>
or <com>answer<CR>
or <com>A<CR>
or <com>a<CR>

BREAK

SYNTAX: <com>B[REAK][<SP><n>]<CR>

Where <n> is 1...9

FUNCTION: Sends a break (long space) condition for n times 250 milliseconds. (If the BREAK command is given without an argument, an argument of 1 is assumed.) After the break is complete the following status is returned:

<com><LF><CR>

EXAMPLE: <com>B<SP>1<CR>

Sends a 250ms space

COUNT

SYNTAX: <com>C[OUNT]<SP><n><CR>

Where <n> is 0...9

FUNCTION: Sets the RING and RING-BACK counter. Incoming calls are auto-answered after n RINGS whereas auto-dialed calls give up waiting for answer tone after n + 4 RING-BACK tone cycles. If n is specified as 0, the CH1770 will not auto-answer calls. The completion of the command is signified by the returned status:

<com><LF><CR>

The counter power-up default count is 2.

EXAMPLE: <com>C<SP>4<CR>

The CH1770 auto-answers after the fourth RING and gives up auto-dialing after the eighth.

DIAL

There are 2 basic variations of the dialing command: dial last, and dial immediate. All dialing commands can be directed to dial using either DTMF tone or rotary-type pulse dialing.

Each dialing variation refers to a sequence of numbers (hereafter referred to as a "number string") as a source for the number to be dialed. For immediate-type dialing commands the number string is supplied with the command, whereas the dial last command refers to a number string previously entered in the CH1770.

Along with telephone number digits the number string can also contain control characters that direct the CH1770 to dial using tone or pulse dialing. Also, pause or wait characters can be inserted that enables tandem dialing through PBX's.

Each number string can be 32 digits long. If control characters are used, fewer numbers may be entered. Each digit occupies one number string position, whereas control characters occupy 2 or 4 positions.

The following characters are allowed in the number string and occupy the indicated number string positions:

Character	Number String Positions	Meaning
0...9	1	dialed digits
TB	4	dials the digits in the number string using DTMF tones.
PB	4	dials the digits in the number string using rotary-type pulses.
B	2	inserts a 2 second pause in the dialing sequence.
Z	2	if placed as the last character in the number string, the CH1770 terminates the dialing command without going into the originate data call mode. The CH1770 stays OFF HOOK with its modulator squelched.
@	2	place holding characters
)	1	place holding characters
(	1	place holding characters
-	1	place holding characters
<SP>		spaces are illegal

The following examples are typical number strings:

Number Strings	Meaning
TB767-1111	dials 767-1111 using tones after waiting 2 seconds.
PB767-1111	dials 767-1111 using pulses after waiting 2 seconds.
PB767-1111Z	same as the previous number string except after dialing 767-1111 the modem does not go into the originate data mode. It stays OFF HOOK with its modulator squelched waiting for the next command. This is very useful for placing voice calls.
TB7771234BB1234567840876711199	dials 777-1234 using tones, then a 4 second pause is inserted before the number 12345678 is dialed. Finally, the number (408)767-1111 is dialed, followed by a two digit access code, 99. This is a typical number string used for calling through long distance carrier facilities.

After a dialing command is given to the CH1770, the appropriate number string is interpreted to determine how the dialing process should proceed. As the number is dialed, the dialed number string is returned to the host in the form of a status message:

<com><NUMBER><LF><CR>

This enables the host to follow the progress of the number dialed. If the number string is terminated with Z however, no further action is taken by the CH1770. The modem is left OFF HOOK with the modulator squelched. This method of dialing is important if dialing is intended to reach a non-modem party.

After the completion of dialing, the CH1770 monitors the telephone line for modem answer tone.

A 'no answer' status message will be returned if the answer tone or some other timer reset signal is not received prior to timeout.

The following status messages are returned immediately after the dialed number message:

Call Progress Status	Meaning
<com>A<LF><CR>	data call answered
<com>N<LF><CR>	no answer, results from no modem answer tone within a 17 second period.

A. Dial last:

SYNTAX: <com>D[IAL][<SP><n>]<CR>

Where <n> is 0...9

FUNCTION: Dials the last number dialed and if <n> is specified, the CH1770 will retry the number n times or until answered. There is a 2 second pause between retries.

EXAMPLE: <com>D<CR>

Dials the last number dialed

<com>D<SP>5<CR>

Dials the last number dialed and retries up to 5 times if the call is unanswered.

B. Dial Immediate:

SYNTAX: <com>D[IAL]<SP><QUOTE><NUMBER><QUOTE>[<SP><n>]<CR>

Where <n> is 0...9
<NUMBER> is a telephone number of 32 digits or less, including control characters.

FUNCTION: Dials the telephone number specified in the command and if <n> is specified, the CH1770 will retry the number n times or until answered. There is a 2 second pause between retries.

EXAMPLE: <com>D<SP>'TB761-1111'<CR>

Dials 767-1111 using tone dialing.

E N D

SYNTAX: <com>E[ND]<CR>

FUNCTION: Ends the call in progress. The following status is returned to indicate command completion:

<com><LF><CR>

N E W

SYNTAX: <com>N[EW]<SP><n><CR>

FUNCTION: The command character <com> is replaced by the new command character <n> specified in the command argument. The power-up default command character is control-N. On command completion, the following status is returned using the new <com> character.

<com><LF><CR>

EXAMPLE: <com>N<SP>/<CR>

Replaces the current command character with / (slash). This is useful when the CH1770 is being controlled by a human through a 'dumb' terminal, since / is a printing character.

ORIGINATE

SYNTAX: <com>O[RIGINATE]<CR>

FUNCTION: Takes the CH1770 OFF HOOK and forces an originate mode call. This command is useful if a call is manually dialed and later a data call is desired.

If the remote modem's answer tone is not detected after nominally 17 seconds, the CH1770 returns the status:

<com>N<LF><CR>

If the call is answered, the following status is returned:

<com>A<LF><CR>

EXAMPLE: <com>O<CR>

Forces an originate call sequence at the modem's currently trained speed.

PROGRAM

SYNTAX: <com>P[ROGRAM][<SP>H_1H_2]<CR>

FUNCTION: Sets or displays the CH1770's internal option parameters. The argument, H_1H_2, is a two digit hex number that specifies the option configuration.

H_1 H_2

PARAMETER DESCRIPTION

0..disconnects on loss of receive carrier
1..does not disconnect on loss of receive carrier

0....does not assert OFF HOOK during modem test
1....does assert OFF HOOK during modem test

0......enables sending & receiving of long space on disconnect
1......disables sending & receiving of long space on disconnect

0........must always be zero

0..........enables answer mode speed switching
1..........disables answer mode speed switching

0............disables unsolicited status messages
1............enables unsolicited status messages

0..............disables echoing of commands
1..............enables echoing of commands

0................must always be zero

If the command is entered without an argument, the current program configuration is returned as status:

<com>H_1H_2<LF><CR>

If an argument is specified, the internal option changes are made and the command complete status is returned to the host:

<com><LF><CR>

The power-up default status is 00.

H_1 position 2, enables unsolicited status messages if programmed to be a 1. Messages are considered to be unsolicited if they result from an occurrence other than a host command. Such messages occur as a result of auto-answer, incoming telephone line ringing, and disconnect resulting from loss of carrier or received long space.

EXAMPLE: <com>P<CR>

Displays the current configuration

<com>P<SP>04<CR>

Programs the CH1770 to not disconnect on long received spaces and not send long spaces on disconnect. All other parameters are at their power-up default state.

QUERY

SYNTAX: <com>Q[UERY]<CR>

FUNCTION: Commands the CH1770 to return its current status.

The following message is returned:

<com>OCHSAUX./X_1 X_2<LF><CR>

Where:

O	OFF HOOK asserted
C	carrier detected
H,M,L	high speed (H:1200,M:300,L:110 bps)
S	self test enabled
A	analog loop test enabled
U	unlisten mode enabled
X	modem echoes commands
/X_1 X_2	current hex TEST error count

If a period (.) replaces any of the status characters it implies the negative or opposite status condition.

EXAMPLE: <com>Q<CR>

The CH1770's status is queried during a normal data transmission and the following is returned:

<com>OCH.....X./00<CR>

This indicates that the modem is OFF HOOK, detecting high speed carrier and set to echo all transmitted commands back to the host.

R E S E T

SYNTAX: <com>R[ESET]<CR>

FUNCTION: Resets the CH1770 to its power-up default condition (see individual commands for default condition).

EXAMPLE: <com>R<CR>

The modem resets to its power up default state then waits for a speed and parity training sequence. A minimum 300 ms pause after this command response must proceed the new training sequence.

T E S T

SYNTAX: <com>T[EST][<SP><n>]<CR>

FUNCTION: Commands the entry and exit of the test mode. To enter the test mode, the TEST command is issued with the argument <n> which indicates any of 4 different test modes:

<n>	Test Mode
0	Analog Loop, originate mode
1	Analog Loop, answer mode
2	Analog Loop Self Test, originate mode
3	Analog Loop Self Test, answer mode
(absent)	Exit the test mode.

To exit any of the TEST modes the TEST command should be given again without an argument. The resulting status sent back to the host is exactly as described in the QUERY command.

TESTS	DESCRIPTION
0,1: Analog Loop	modulates the transmit data asserted on TXD and loops it back through its own demodulator and outputs the resultant receive data on RXD. This allows the host to test the entire CH1770 by checking to see if the receive data matches the transmit data.
2,3: Analog Loop Self Test	same as Analog Loop except the CH1770 transmits a pseudo random data pattern in place of the host data. It then checks to see that the correct data pattern is received. Detected errors cause the CH1770 to increment its self test error register. Since the self test register is cleared upon the start of the test mode, the error count status returned at the end of the test command represents the number of data errors encountered during the test. The register maximum count is FF hex.

EXAMPLE: <com>T<SP><n><CR>

Enter test mode specified by <n>

<com>T<CR>

Exit test mode

UNLISTEN

SYNTAX: <com>U[NLISTEN][<SP><n>]<CR>

Where <n> is 0 or 1 or absent

FUNCTION: Enables or disables the CH1770 command interpreter in the data transmission mode. The command interpreter is always enabled when the CH1770 is not in the middle of a data call. Once a call is set up, three different UNLISTEN modes are possible. At power-up, the default state of the command interpreter is in the LISTEN mode.

<n>	Description
argument absent	listens for commands in the data mode
0	does not listen for commands in the data mode until the host transmits a break (start bit, data bits all zero, and a zero stop bit). Thereafter, the CH1770 listens for host commands until it is commanded to again UNLISTEN. The host's break signal is absorbed by the command interpreter and does not pass on through to the telephone line.
1	does not listen for commands in the data mode. The only way the host can command the CH1770 to disconnect at the end of the call in this mode is to assert $\overline{\text{DTR}}$ OFF (high).

The command assures that inadvertent embedded commands in blocks of data sent through the CH1770 will not inappropriately affect the data transmission. Both binary and ASCII files can therefore be passed through the modem with complete data transparency assured.

The CH1770 returns the following status to show command completion:

<com><LF><CR>

EXAMPLE: <com>U<CR>

Sets the CH1770 to listen to commands in the data mode.

<com>U<SP>0<CR>

Sets the CH1770 to not listen (or unlisten) to commands during the data mode until it receives a break.

<com>U<SP>1<CR>

Sets the CH1770 to not listen to commands during the data mode.

Z Z Z Z

SYNTAX: <com>Z[ZZZ]<CR>

FUNCTION: Makes the modem quiet. When the command is given, the modem squelches its transmitter and stays OFF HOOK. To escape this command, either the ORIGINATE, ANSWER or END commands should be issued.

If both modems on either end are optioned to not disconnect on carrier loss, this command can enable the telephone connection to be used for voice and data switching under host control.

The following status is returned on completion of the ZZZZ command:

<com><LF><CR>

8.0 FCC GENERAL INFORMATION

FCC rules and regulations under part 68, requires the following information be provided to the user of FCC-registered terminal equipment such as the Cermetek CH1770.

Section 68.100 GENERAL

Terminal equipment may be directly connected to the telephone network in accordance with the rules and regulations...of this part.

Section 68.104 STANDARD PLUGS AND JACKS

(a) General

"Except for telephone company-provided ringers, all connections to the telephone network shall be made through standard (USOC) plugs and standard telephone company-provided jacks, in such a manner as to allow for easy and immediate disconnection of the terminal equipment. Standard jacks shall be so arranged that if the plug connected thereto is withdrawn, no interference to the operation of equipment at the customer's premises which remains connected to the telephone network shall occur by reason of such withdrawal."

Section 68.106 NOTIFICATION TO TELEPHONE COMPANY

"Customers connecting terminal equipment or protective circuitry to the telephone network shall, before such connection is made, give notice to the telephone company of the particular line(s) to which such connection is to be made, and shall provide to the telephone company the FCC Registration Number and Ringer Equivalence of the registered terminal equipment or protective circuitry. The customer shall give notice to the telephone company upon final disconnection of such equipment or circuitry from the particular lines(s)."

Section 68.108 INCIDENCE OF HARM

"Should terminal equipment or protective circuitry cause harm to the telephone network, the telephone company shall, where practicable, notify the customer that temporary discontinuance of service may be required; however, where prior notice is not practicable, the telephone company may temporarily disconnect service forthwith, if such action is reasonable in the circumstances. In case of such temporary discontinuance, the telephone company shall (1) promptly notify the customer of such temporary discontinuance, (2) afford the customer the opportunity to correct the situation which gave rise to the temporary discontinuance, and (3) inform the customer of the right to bring a complaint to the Commission pursuant to the procedures set forth in Subpart E of this Part."

Section 68.216 REPAIR OF REGISTERED TERMINAL EQUIPMENT AND REGISTERED PROTECTIVE CIRCUITRY

"Repair of registered terminal equipment and registered protective circuitry shall be accomplished only by the manufacturer or assembler thereof or by their authorized agent; however, routine repairs may be performed by a user, in accordance with the instruction manual if the applicant certifies that such routine repairs will not result in noncompliance with the rules in Subpart D of this Part." (This applies anytime during and after the factory warranty period. Faulty equipment should be returned to the Cermetek Distributor or Cermetek Factory for repair.)

Section 68.218(b) ADDITIONAL INSTRUCTIONS TO USER

1. "...registered terminal equipment or protective circuitry may not be used with coin lines."

2. "...when trouble is experienced, the customer shall disconnect the registered equipment from the telephone line to determine if the registered equipment is malfunctioning, and...if the registered equipment is malfunctioning, the use of such equipment shall be discontinued until the problem has been corrected."

3. "...the user must give notice to the telephone company in accordance with the requirements of Section 68.106..." for connecting the CH1770 to the telephone line.

9.0 ELECTRICAL SPECIFICATIONS

Power: +5V +/-5%, -5V +/-5%

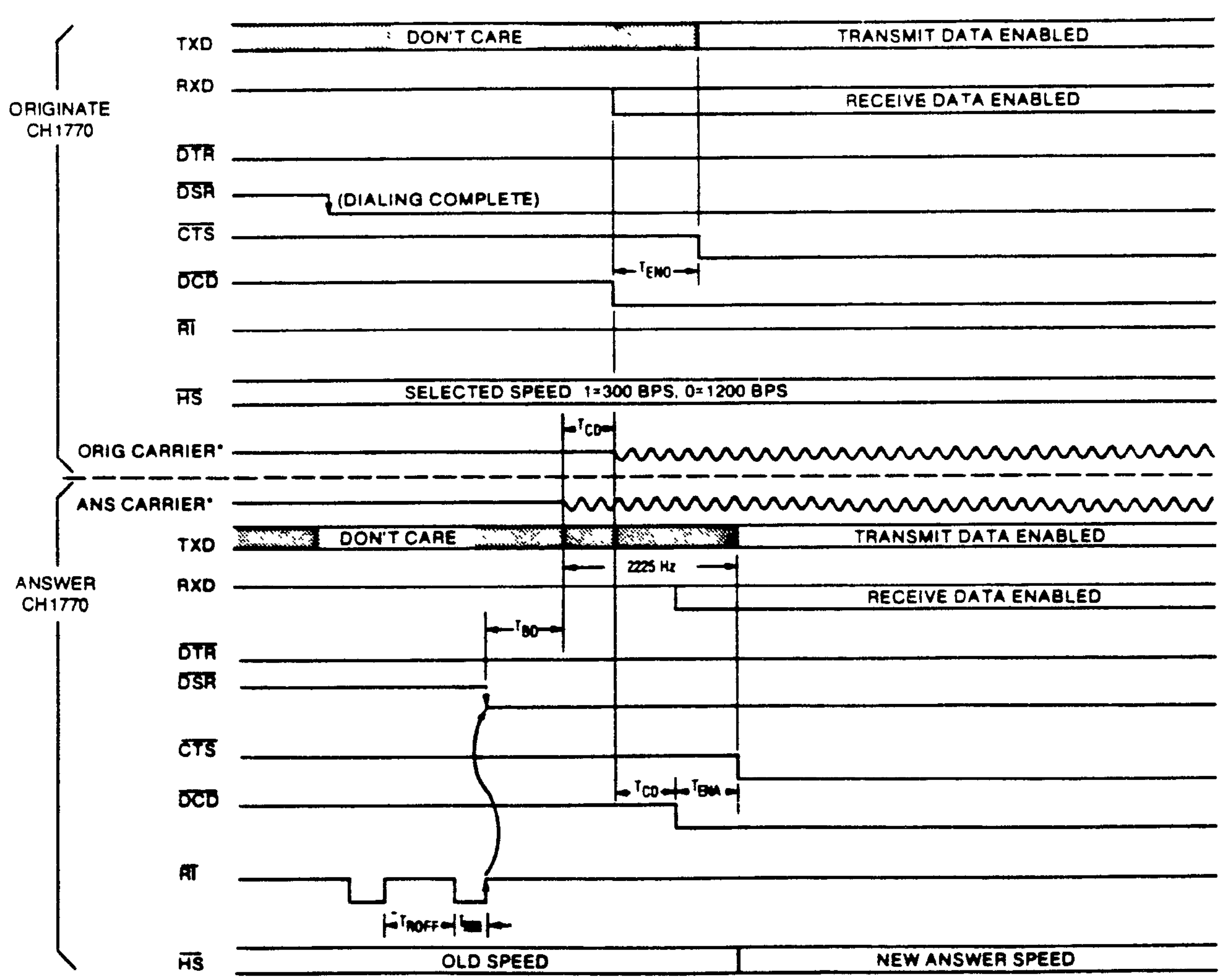
TA: 0-60 degrees C

PARAMETERS	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
<u>LOGIC I/O LINES</u>						
Input high	Vih		2.0			V
Input Low	Vil		-.3		.8	v
Input current high	Iih				500	µA
Input current low	Iil				-500	µA
Output high	Voh	OH = .2mA	2.4	3.5		V
Output low	Vol	OL = .2mA	0.0	.2	.45	V
<u>TELEPHONE LINE INTERFACE</u>						
AC Impedance	Zline			600		ohm
Surge Protection		Conforms to all FCC Part 68 surge, hazardous voltage, and leakage				
Carrier Transmit Level	Ptx	600 ohm line termination	-11	-10	-9	dbm
Carrier Receive Sensitivity	Rcar	OFF to ON detection		-43		dB
		ON to OFF drop out		-48		dB
ON HOOK Impedance	Zonhk		20M			ohm
LOOP CURRENT	Iloop			20	100	mA
FCC Registration Number		B468NR-68618-DM-E				
RINGER EQUIVALENCE				0.4B		

PARAMETERS	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
DIALING						
DTMF LEVEL			-8	-6	-4	dBm
DTMF FREQ ACCURACY			-1.0		+1.0	%
DTMF ON TIME			90	100	110	ms
DTMF OFF TIME			90	100		ms
PULSE SPEED			9	10	11	PPS
PULSE RATIO		MAKE/BREAK RATIO		40/60		%
PULSE INTERDIGIT			650	700	750	ms
DTE INTERFACE TIMING						
Carrier Detect	Tcd		150	--	300	ms
Clear to Send Delay (Answer)	Tena	110 or 300 BPS	175	200	225	ms
		1200 BPS	750	775	800	ms
Clear to Send Delay (Originate)	Teno	110 or 300 BPS	125	150	175	ms
		1200 BPS	1400	1475	1500	ms
Billing Delay	Tbd		2.0		2.2	Sec
Ring Cycle ON	Tron		0.1	2.0	3.0	Sec
Ring Cycle OFF	Troff		0.5	4.C	6.0	Sec
DISCONNECT TIMING						
DTR Forced	Tdtr	DTR asserted OFF (high)	50			ms
		DTR asserted ON (low)		10		
Long Received Space	Tlrs	Optional	1.6			Sec
Loss of Carrier	Tlc	Carrier drop out, Optional	77			ms
Send Long Space	Tsls	Optional	4		4.5	Sec
POWER						
	Icc	Current at +5 volts		300	350	mA
	Ihmv	Current at -5 volts		30	45	mA

PARAMETERS	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
WEIGHT					.3	lb
SIGNALING RATE		1200 BPS PSK Asynchronous	-2.5		+1.0	%
		110 or 300 BPS, FSK	-2.5		+2.5	%
BIT Error Rate		Average Line, 9dB S/N 1200 BPS			10^{-5}	Errors/Sec
		Average Line 5dB S/N 110 or 300 BPS			10^{-5}	Errors/Sec

CH1770 HANDSHAKE TIMING DIAGRAM



Status Transceiver Theory of Operation

Status Transceiver Board 1449 Theory of Operation

The 1449 status transceiver provides status input and output interface between the DRC19Ø and external equipment. Standard applications call for 32 inputs and outputs, although the system can be expanded to 96 inputs and outputs per site.

The 1449 board mounts on the front panel of the DRC19Ø. It is held in place by the LED clips of the display LEDs. To remove the board, the LEDs should be firmly pressed from the outside of the panel with a wide flat blade screwdriver. Once all the LEDs are snapped out, the board can be lifted out. The individual LEDs plug in to the 1449 board, allowing easy replacement. Some stations use this feature to have green LEDs for the ones that should be on, red for the ones that should be off.

The 1449 board is driven by the 1443 Power, Disk and Status interface board. Additional information on the serial communications between the interface and the status transceiver appears in the 1443 theory of operation section.

page 1. Status Input Shift Registers

UØ1, UØ2, UØ3 and UØ4 are parallel load, serial output shift registers. A low pulse on LOAD-IN (load the input shift registers, as opposed to the output shift registers, which are on pages 3 and 4), captures the parallel data presented to the inputs. RØ1, RØ2, RØ3 and RØ4 provide input pull-up resistors for these chips. These allow the status inputs to be driven by TTL level signals, or open collector drivers or contact closures to ground.

The input shift registers are cascaded. Data in UØ4 is shifted into UØ3, then UØ2, then UØ1. From UØ1, it is shifted through one more bit of shift register, formed by U1ØA, then to the shift register in the VIA on the 1443 interface board. On each positive edge of SCLK, one bit of status data is shifted towards the VIA on the 1443 board. On the positive edge of SCLK, SCLK* will have a negative edge, causing U1ØA to "shift" the input bit to the output prior to the change of the bit at UØ1 pin 9. It typically takes 27 nS for the output of the 74LS165 to change state following the positive edge of the clock. By this time, the next shift register (or the D flip flop) has captured the previous data. On each positive edge of SCLK, status data is moved through U1ØA towards the VIA on the 1443 board. U1ØA is necessary because the VIA sends one clock pulse before capturing the serial data. UØ1-Q7 has the first data bit (status line Ø7) prior to any clock pulses. After the first clock pulse, UØ1-Q7 has status line Ø6. After the first clock pulse, status line 7 is available (inverted) at the Q* output of U1ØA (Status-In).

The firmware, working with the VIA on the 1443 board, shift in 12 bytes of status (96 lines, numbered ØØ to 95). Status lines beyond 31 may be shifted into the system using PØ4, the expand connector. This connector allows additional shift registers to be cascaded.

page 2, Site Compatator

When the VIA on the 1443 board shifts out status, it shifts out 12 bytes of status (with the last byte holding received status ØØ to Ø7). These 12 bytes are then followed by a byte indicating the site that this status came from. If the site the status was received from is the same as the site currently displayed on the front panel LCD, the status information is also sent to the status transceiver with a "site number" of 1ØØ (decimal).

This last byte (the status address) is captured by UØ7 and presented as parallel data to UØ6. UØ6 compares the received address (from UØ7) to the programmmed address (set by jumpers on PØ5). If the address matches, the LOAD-OUT* pulse is allowed to pass through UØ6. If there is no match. the LOAD-OUT* pulse is not passed on to the rest of the system (as RX-SEL*). If the address on PØ5 matches the received address, the data is latched in the remaining shift registers. If the address does not match, the data remains in the shift registers, but is not latched, and does not drive the output drivers.

page 3, Display Driver

UØ8 shifts in the received status data from the VIA on the 1443 board. The last status received by this chip is status ØØ. Status 95 through status 32 have been shifted through this chip to the expand jack on page 1. Once all the status has been received, RX-SEL is pulsed high if the address sent matches the address selected on PØ5. Note that RX-SEL is active high. It is RS-SEL* from UØ6 inverted by UØ5B.

Once the data has been latched by UØ8, an incoming data 1 causes the appropriate output to be pulled low, lighting the appropriate LED. RØ6, RØ7, RØ8 and RØ9 provide pull-up current limiting for the LEDs.

page 4, User Port Driver

UØ9, shown on page 4, duplicates the action of UØ8 on the previous page. If the address matches, the data is latched and provided to the output lines. These outputs drive PØ6, which drives a cable to the rear panel. This chip can drive user devices, such as remote status indicators, beepers, etc.

Status Expander Theory of Operation

Theory of Operation

The STX191 consists of two 1449 Status Transceiver boards. The status transceiver uses parallel in, serial out shift registers to read status. It uses serial in, parallel out shift registers to output status. When status is expanded with the STX191, the each shift register (input and output) are lengthened to the required number of bits. A few changes are made on the 1449 board when it is used in the STX191. These are outlined below.

U1Ø Bypassed

The status input shift register drives a 6522 on the disk/status interface board. Due to timing differences between the 6522 shift register and the 74HC165 shift registers on the status transceiver board, it was necessary to add one bit to the shift register between the 74HC165s and the 6522. This bit is thrown out by the 6522 when it reads the first byte of incoming status data.

When the status transceiver is used as an expander, the 74HC165s are cascaded, and the "extra bit" is not required. Therefore, U1Ø is bypassed on each 1449 board that is used as an expander, which is each board except the one that connects to the disk/status interface board of the DRC19Ø. The board that inputs status ØØ to 31 has U1Ø. All other boards get a jumper between pins 2 and 6 of U1Ø instead of U1Ø.

In addition, a 13Ø ohm pullup resistor is added between pins 3 and 14 of U1Ø. This allows the serial clock driver on the disk/status interface board to drive the increased capacity due to longer cables in linking the status transceiver boards. In addition, this acts as a terminating resistor for the transmission line formed by the interconnecting cable, preventing signal reflections, which can cause multiple clocks being seen by the shift registers.

UØ5C, UØ6 and UØ7 Bypassed

UØ7 of the 1449 status transceiver board normally holds the last byte sent by the 6522 on the disk/status interface board of the DRC19Ø. This byte holds the site number that the status was sent from. When status transceivers are cascaded to output more status channels from the same site, this "site decoding" is not required. Also, due to the differences in timing between the 5832 shift register and the 74HC164, it is not possible to further cascade the shift registers. For these reasons, UØ6 is removed and replaced with a jumper between pins 1 and 19, passing through the shift register "load-out*" pulse that was qualified with the site number on the first 1449 board. In addition, UØ7 is removed and replaced with a jumper between pins 2 and 13. Pins 5 and 6 of UØ5 are flared and replaced with a jumper, preventing inversion of the data presented to the status expander output. These changes result in the 5832 shift registers in the 1449 expander boards being driven directly by the 5832 on the original 1449 status transceiver board. The result is a longer shift register. Note: these changes are made only on the "expander" boards. The board that connects to the disk/status interface of the DRC19Ø remains unchanged, since it must still do site decoding.

The changes mentioned in the above paragraph are required only if the output shift registers are being cascaed to get more status outputs. If, instead, the additional outputs are to provide a continuous display of a different site (each status transceiver programmed to receive from a different site), these changes are not required.

Summary

The STX191 consists of 1449 status transceiver boards. Boards that are used to expand the system beyond 32 channels of status are modified, resulting in the input and output shift registers being cascaded directly. For theory of operation on the 1449 board (and schematics and parts placement), refer to the 1449 section of the DRC190 manual.

Bibliography

The following books, articles, and data sheets can be used as reference material with the DRC19Ø.

Commodore Disk Interface

The Anatomy Of The 1541. Authors: Lothar Englisch, Norbert Szczepanowski, Edited by: Greg Dykema, Arnie Lee. Abacus Software, P.O. Box 7211, Grand Rapids, MI 4951Ø (612) 241-551Ø

Inside Commodore DOS, Richard Immers, Ph.D., Gerald G. Neufeld, Ph.D.. Datamost, 2Ø66Ø Nordhoff Street, Chatsworth, CA 91311-6152, (818) 7Ø9-12Ø2

VIC Revealed, Nick Hampshire, Hayden Book Company, Inc., Rochelle Park, NJ

Personal Computing on the VIC-2Ø, Commodore Electronics, Inc.

Commodore 1541 Disk Drive User's Guide, Commodore Business Machines, Inc., 12ØØ Wilson Drive. West Chester, PA 1938Ø

How The VIC/64 Serial Bus Works, Jim Butterfield, Compute magazine, July 1983

IEEE488 Interface

IEEE Standard Digital Interface for Programmable Instrumentation, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 1ØØ17

ZT 7488 IEEE488 Interface for STD Bus Operating Manual, Ziatech Corporation, 3433 Roberto Court, San Luis Obispo, CA 934Ø1 (8Ø5) 541-Ø488

FCC Rules & Regulations

Code of Federal Regulations, 47CFR parts 15 (radiation by computing devices). 68 (connection of registed devices to the public switched telephone network), 73 (rules regarding operation of broadcast stations), part 74 (rules regarding auxiliary broadcast services), available from Superintendent of Documents, U.S. Government Printing Office, Washington, DC 2Ø4Ø2

Semiconductor Devices

A large number of semiconductors are used in the DRC19Ø. Most of these are standard semiconductos available from several manufacturers. The newer or somewhat unusual devices are listed below.

Exar Corporation, 75Ø Palomar Avenue, P.O. Box 62229, Sunnyvale, CA 94Ø88, (4Ø8) 732-797Ø. XR22Ø6 FSK Generator, XR2211 FSK Demodulator, Exar Modem

Handbook.

Hitachi America, Ltd., 2210 O'Toole Ave., San Jose, CA 95131, (408) 942-1500. HM6264LP-15 8 Kbyte static RAM

Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051, (408) 987-8080. 27256 32 Kbyte EPROM

Intersil, Inc., 10710 N. Tantau Avenue, Cupertino, CA 95014, (408) 996-5000. ICL7135CPI Analog to digital converter

Monolithic Memories, 2175 Mission College Blvd., Santa Clara, CA 95054, (408) 970-9700. MM6331-1 32x8 Tri-State Bipolar PROM (used as memory map decoder).

Motorola Semiconductor Products, 5005 East McDowell Road, Phoenix, AZ 85008. MC6802 Processor Data Sheet, M6800 Programming Reference Manual

RCA Solid State Division, Box 3200, Somerville, NJ 08876, (201) 685-6000. CD74HC166 Parallel In, Serial Out Shift Register, used in status panel

Rockwell International, Semiconductor Products Division, 4311 Jamboree Rd., P.O. Box C, Newport Beach, CA 92658-8902, (714) 833-4700. R6522 VIA used on processor and A/D boards.

Signetics Corporation, 811 East Arques Avenue, P.O. Box 409, Sunnyvale, CA 94086, (408) 739-7700. SCN2681AC1N40 DUART used on processor and direct connect modem boards.

Sprague Semiconductor Division, 115 Northeast Cutoff, Worcester, MA 01606. UCN-4821A serial input latched peripheral drivers, used in status panel

Texas Instruments, Semiconductor Group, P.O. Box 401560, Dallas, TX 75240, (214) 995-2011. SN7538E quad NAND peripheral driver, used on A/D board

Xicor, Inc., 851 Buckeye Court, Milpitas, CA 95035, (408) 946-6920. X2816A EEPROM used on processor board.

Index

Index

Index

Index

Index

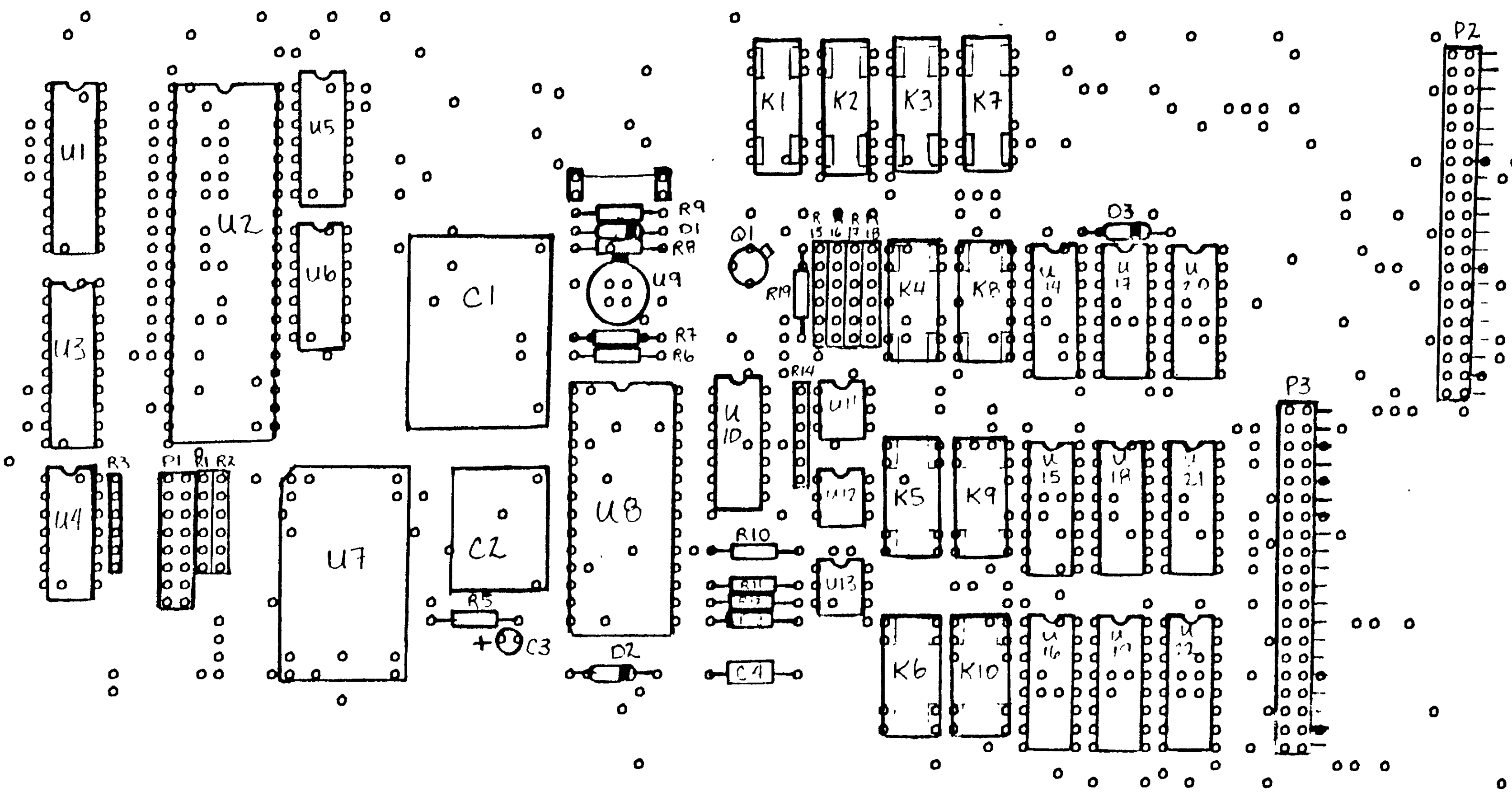
Index

Index

Index

BE 10/9/85

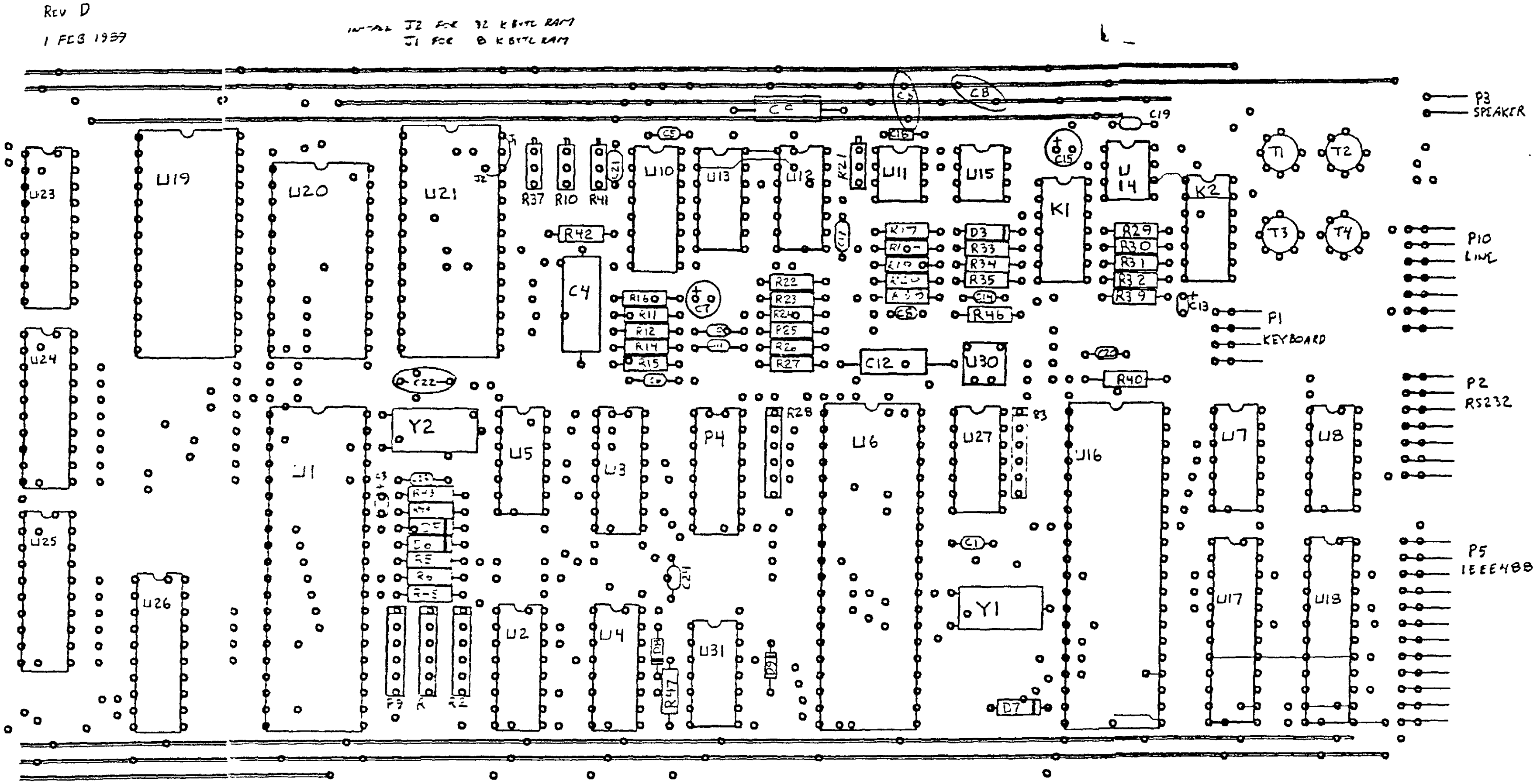
H&F #0010-1441 Component Placement
DRC 190 A/D Sub-assy A-190-

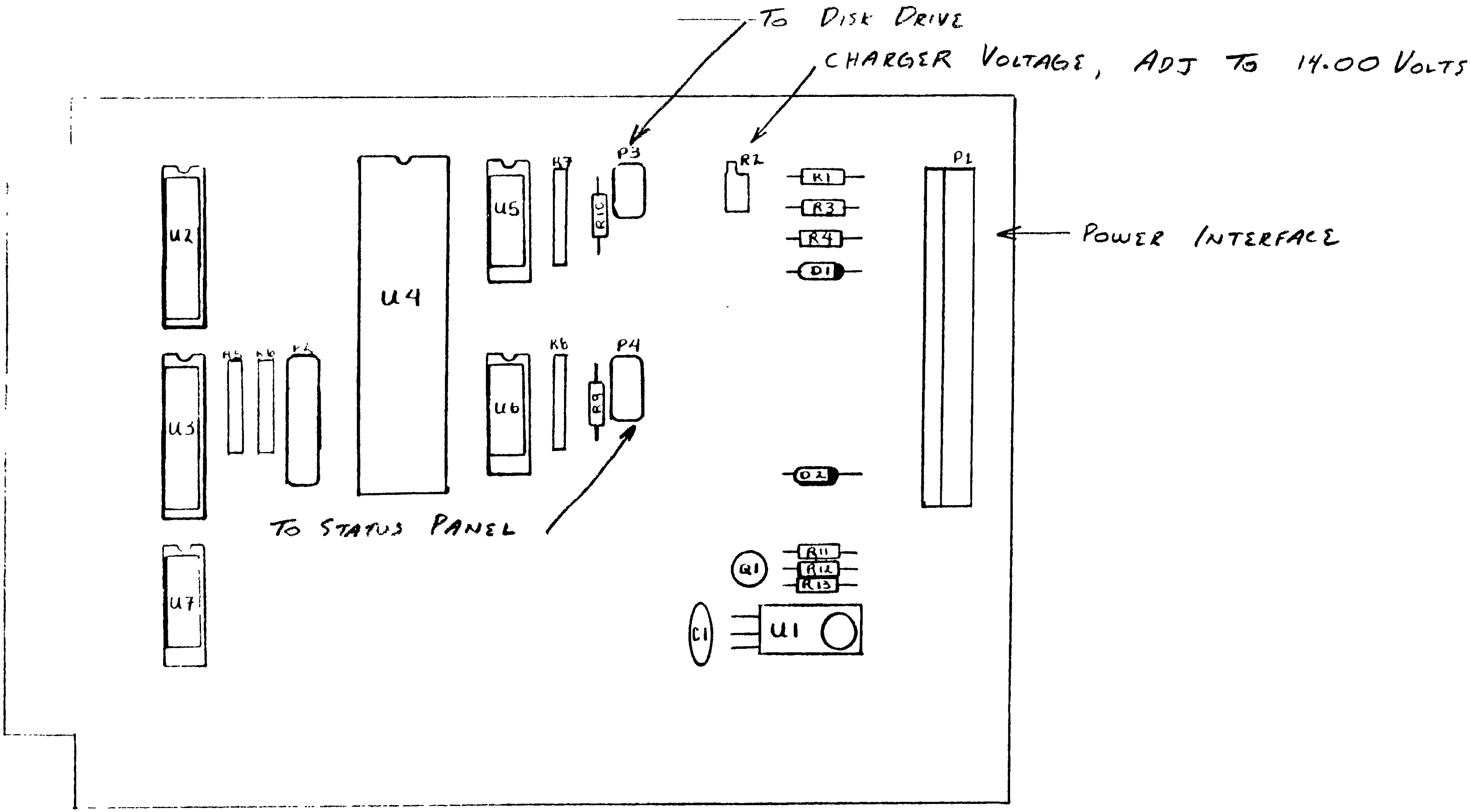


H & F 1442 Processor Board

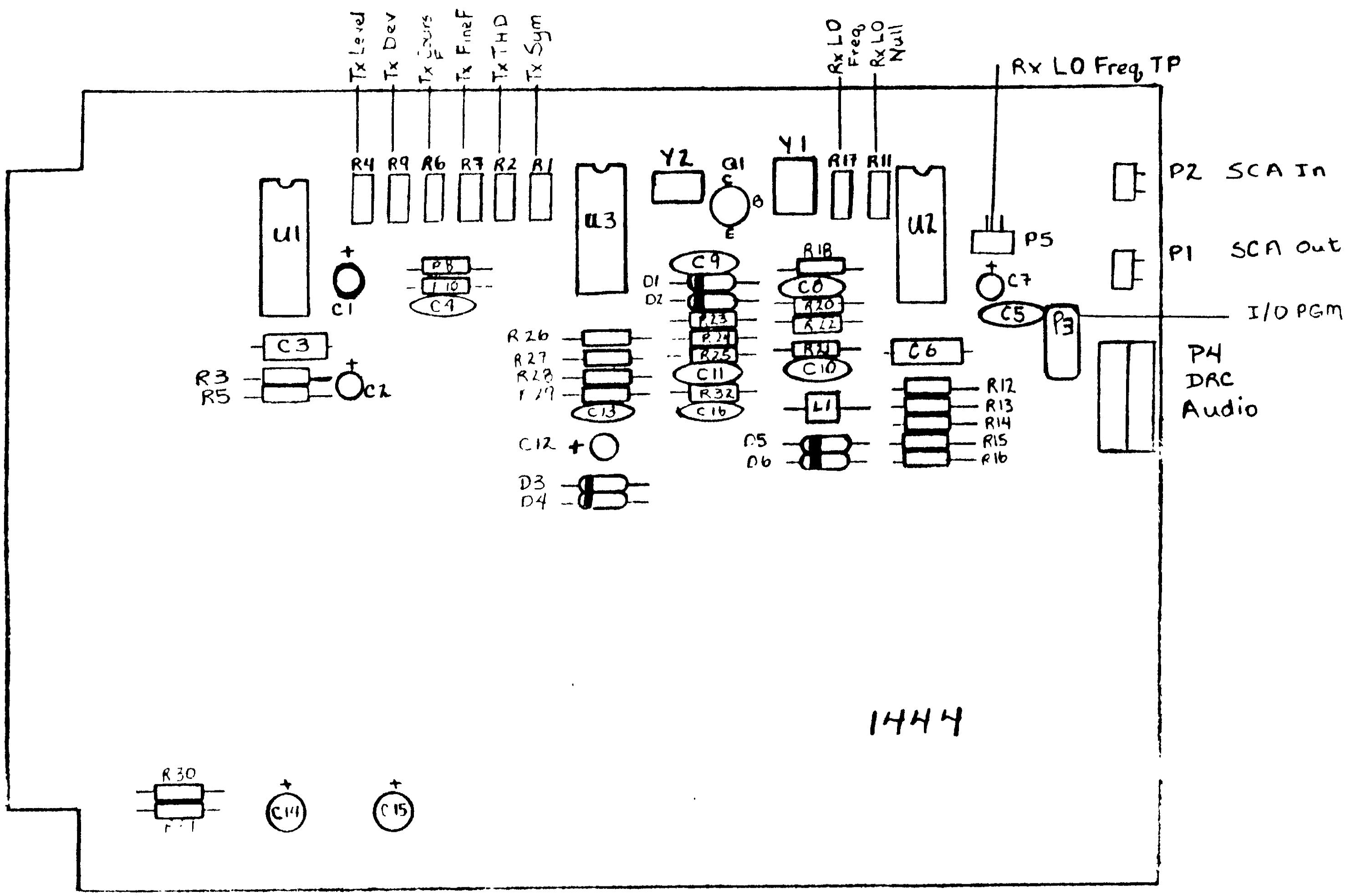
Rev D

1 FEB 1987

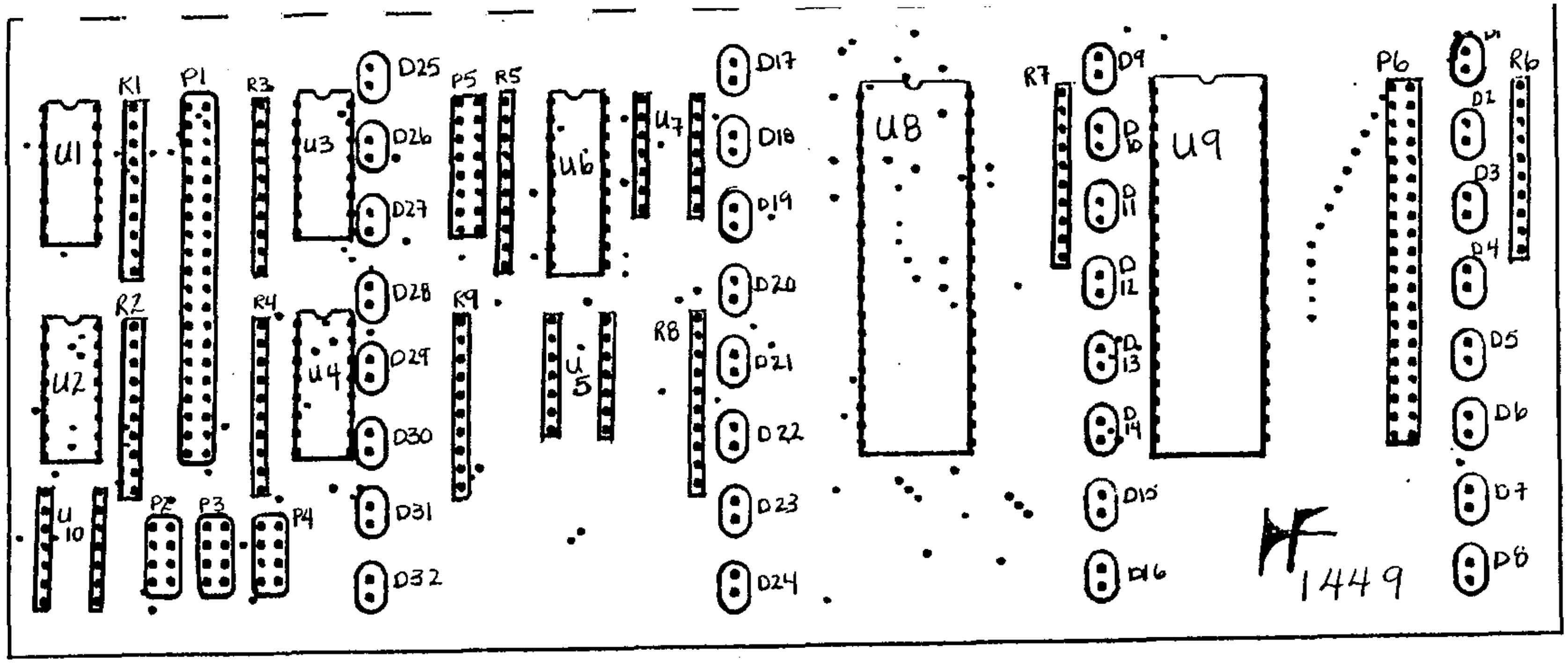




Disk, Status Board 1443

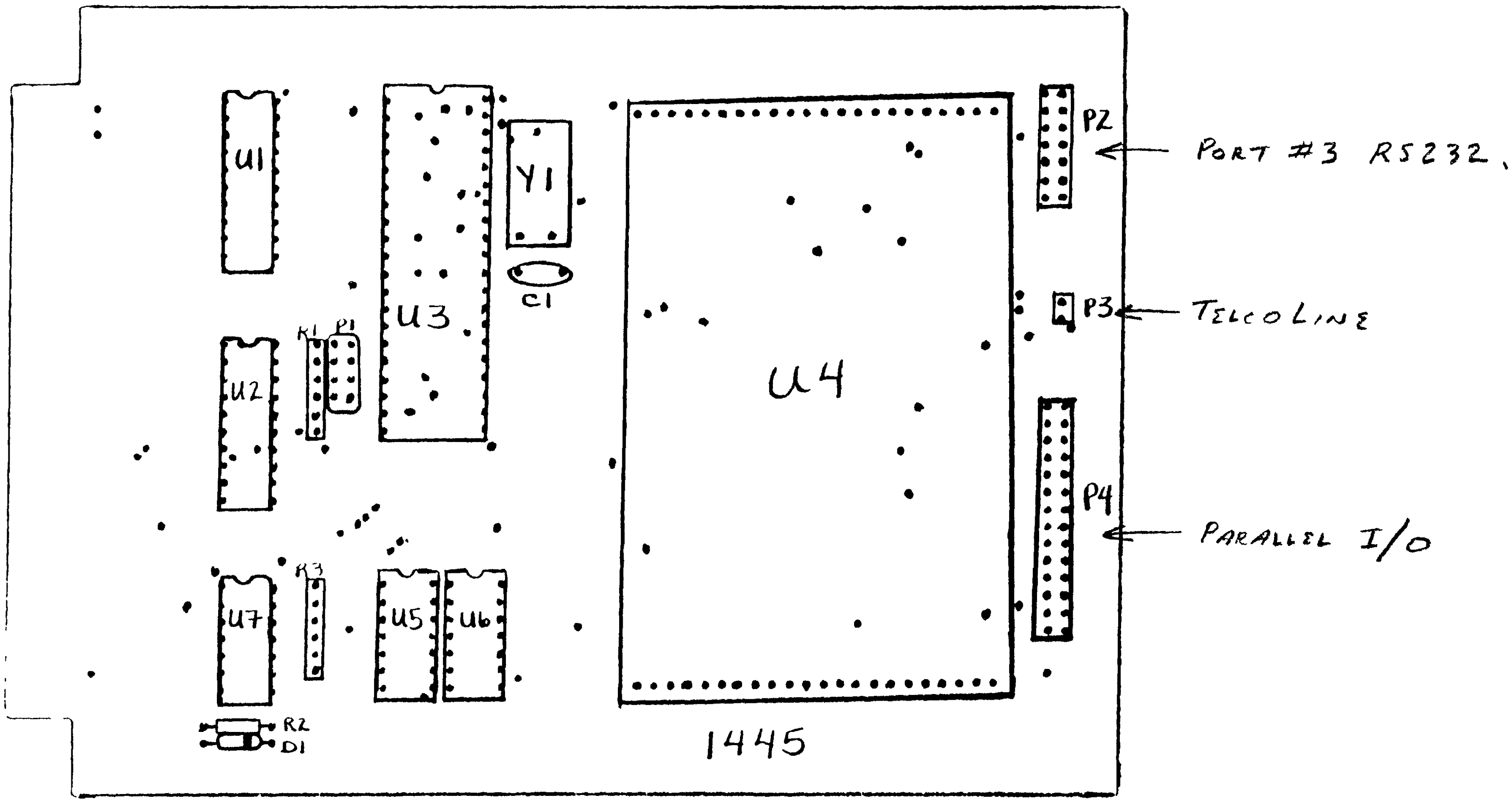


COMPONENT PLACEMENT: SUBCARRIER TRANSCEIVER 1444
H&F #0010-1444



Hallikainen & Friends
0010-1449 Status Panel

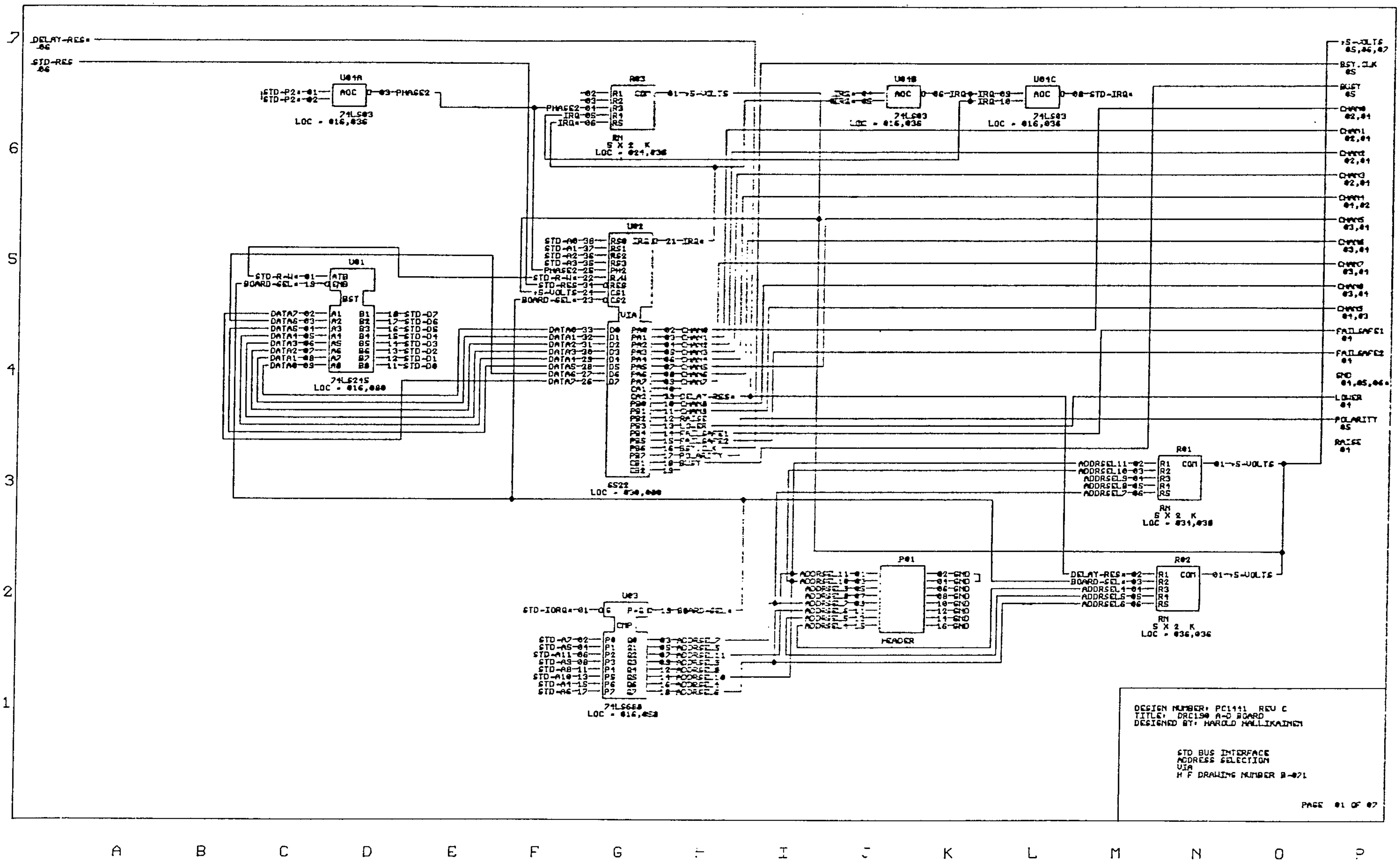
Page

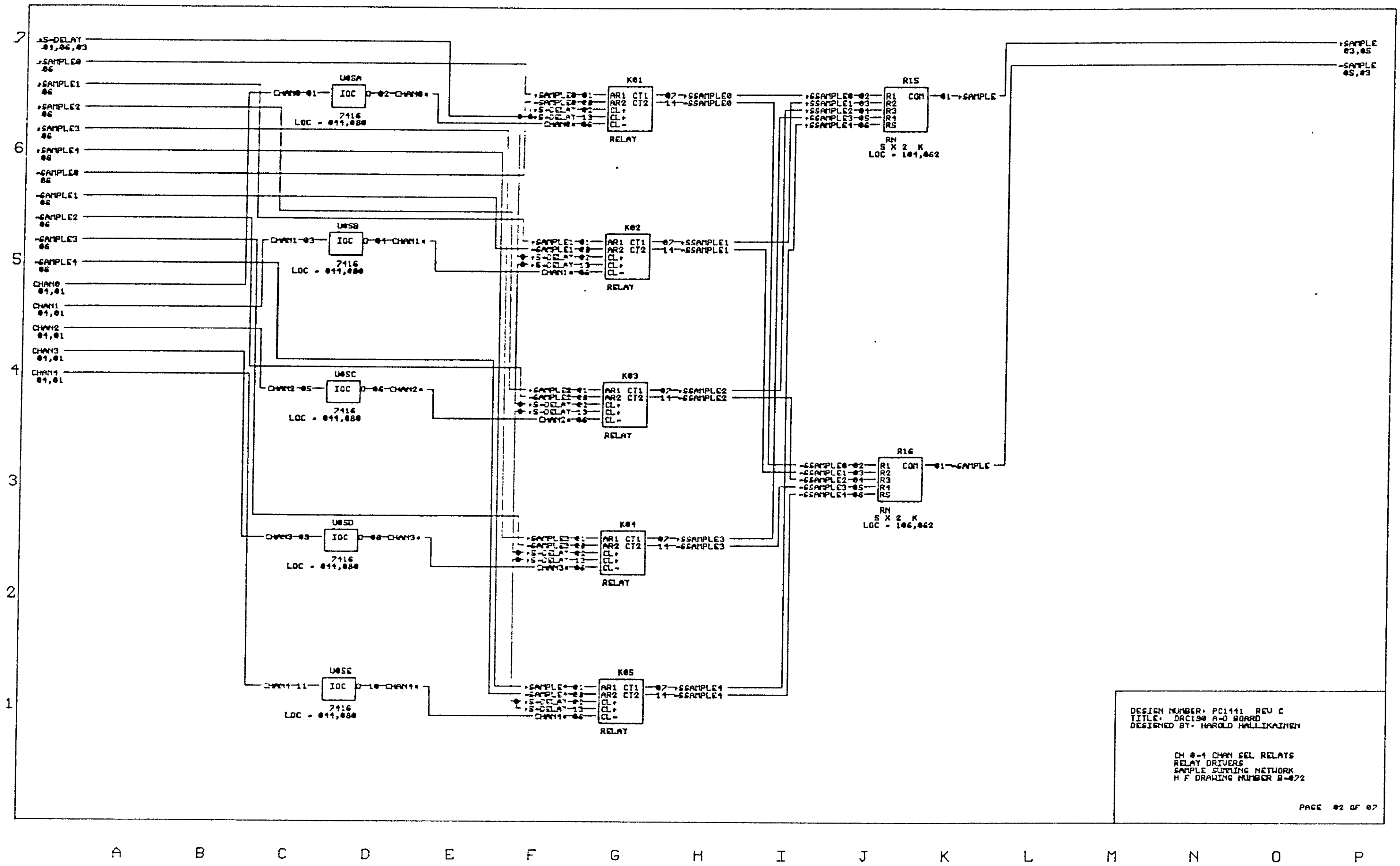


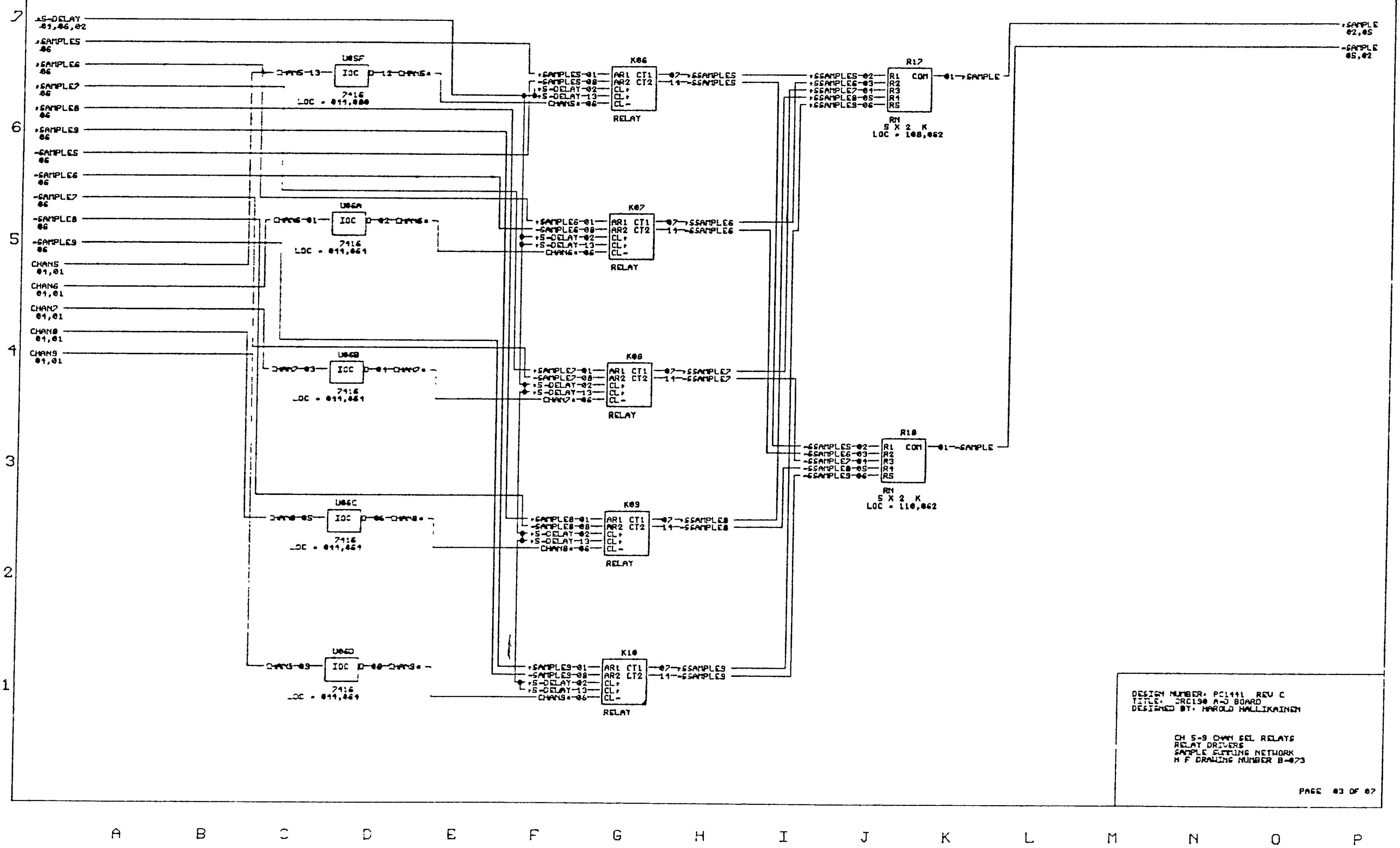
BE 10/16/85

DRC 190
H & F #0010-1445

Direct Connect
Modem


DESIGN NUMBER: PC1441 REV C
TITLE: DRC190 A-D BOARD
DESIGNED BY: HAROLD HALLIKAINEN
STD BUS INTERFACE
ADDRESS SELECTION
VIA
H F DRAWING NUMBER B-071
PAGE 01 OF 07
74LS03
74LS245
6522
74LS688
HEADER

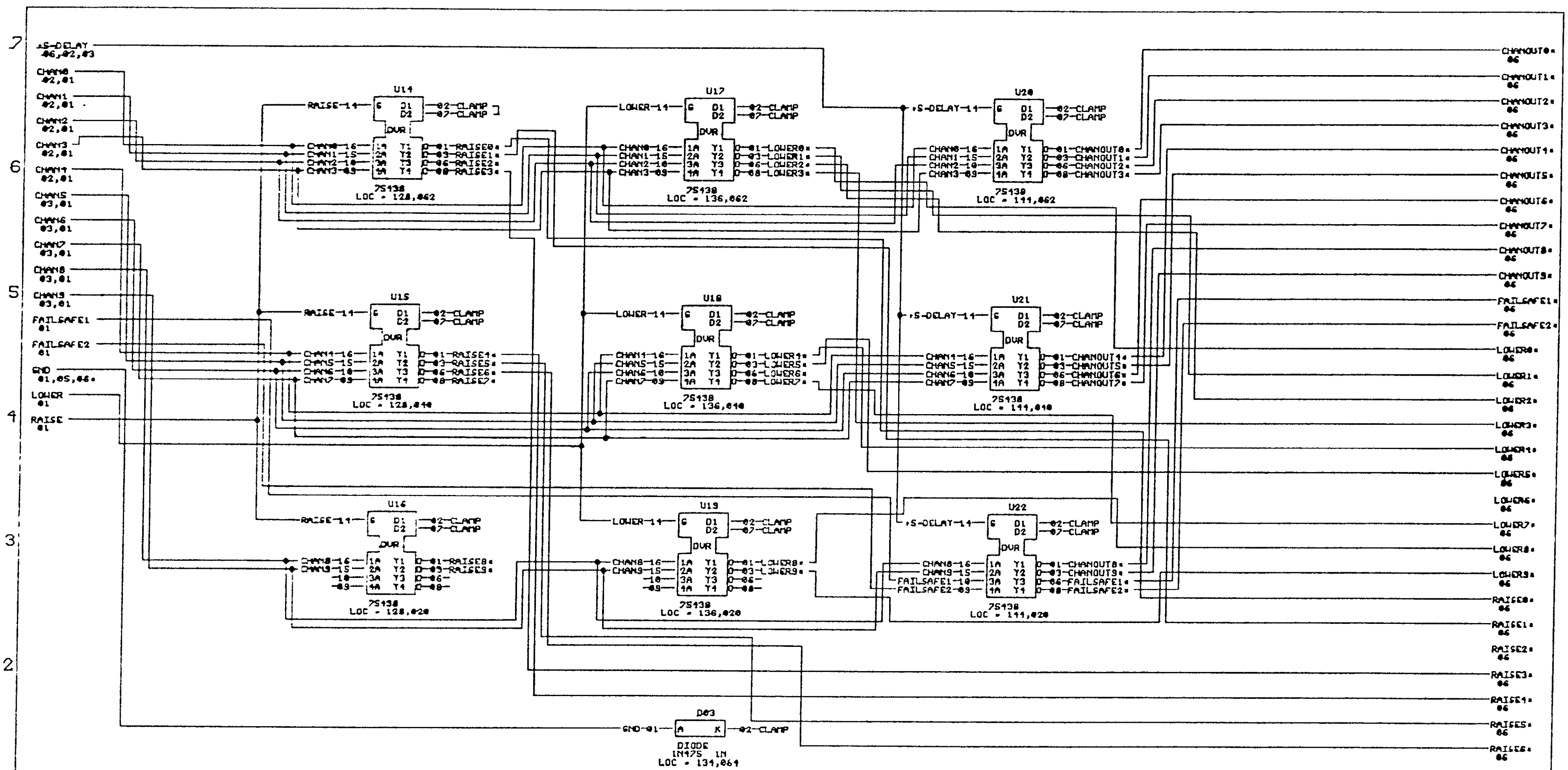

+S-DELAY
+SAMPLE0
+SAMPLE1
+SAMPLE2
+SAMPLE3
+SAMPLE4
-SAMPLE0
-SAMPLE1
-SAMPLE2
-SAMPLE3
-SAMPLE4
CHAN0
CHAN1
CHAN2
CHAN3
CHAN4
U05A
U05B
U05C
U05D
U05E
IOC
7416
LOC = 011,080
K01
K02
K03
K04
K05
AR1 CT1
AR2 CT2
CL+
CL-
RELAY
R15
R16
R1 COM
R2
R3
R4
R5
RN
5 X 2 K
LOC = 101,062
LOC = 106,062
+SAMPLE
-SAMPLE
DESIGN NUMBER: PC1441 REV C
TITLE: DRC190 A-D BOARD
DESIGNED BY: HAROLD HALLIKAINEN
CH 0-4 CHAN SEL RELAYS
RELAY DRIVERS
SAMPLE SUMMING NETWORK
H F DRAWING NUMBER B-072
PAGE 02 OF 07
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
1
2
3
4
5
6
7



DESIGN NUMBER: PC1441 REV C
TITLE: CRC190 A-D BOARD
DESIGNED BY: HAROLD HALLIKAINEN

CH 5-9 CHAN SEL RELAYS
RELAY DRIVERS
SAMPLE SUMMING NETWORK
H F DRAWING NUMBER B-073

PAGE 03 OF 07

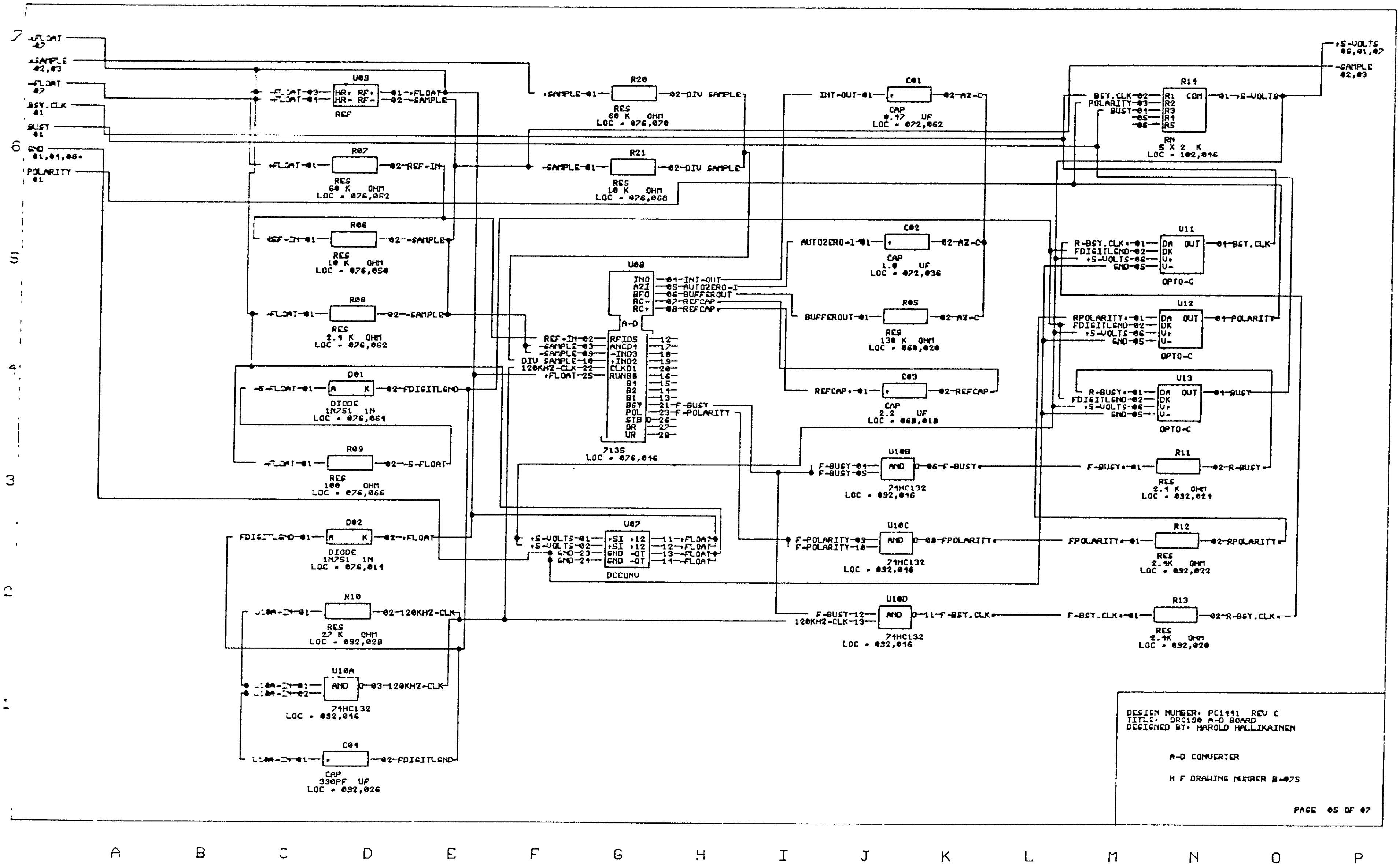


DESIGN NUMBER: PC1441 REV C
TITLE: DRC190 A-D BOARD
DESIGNED BY: HAROLD HALLIKAINEN

CONTROL LINE DRIVERS

H F DRAWING NUMBER B-074

PAGE 04 OF 07

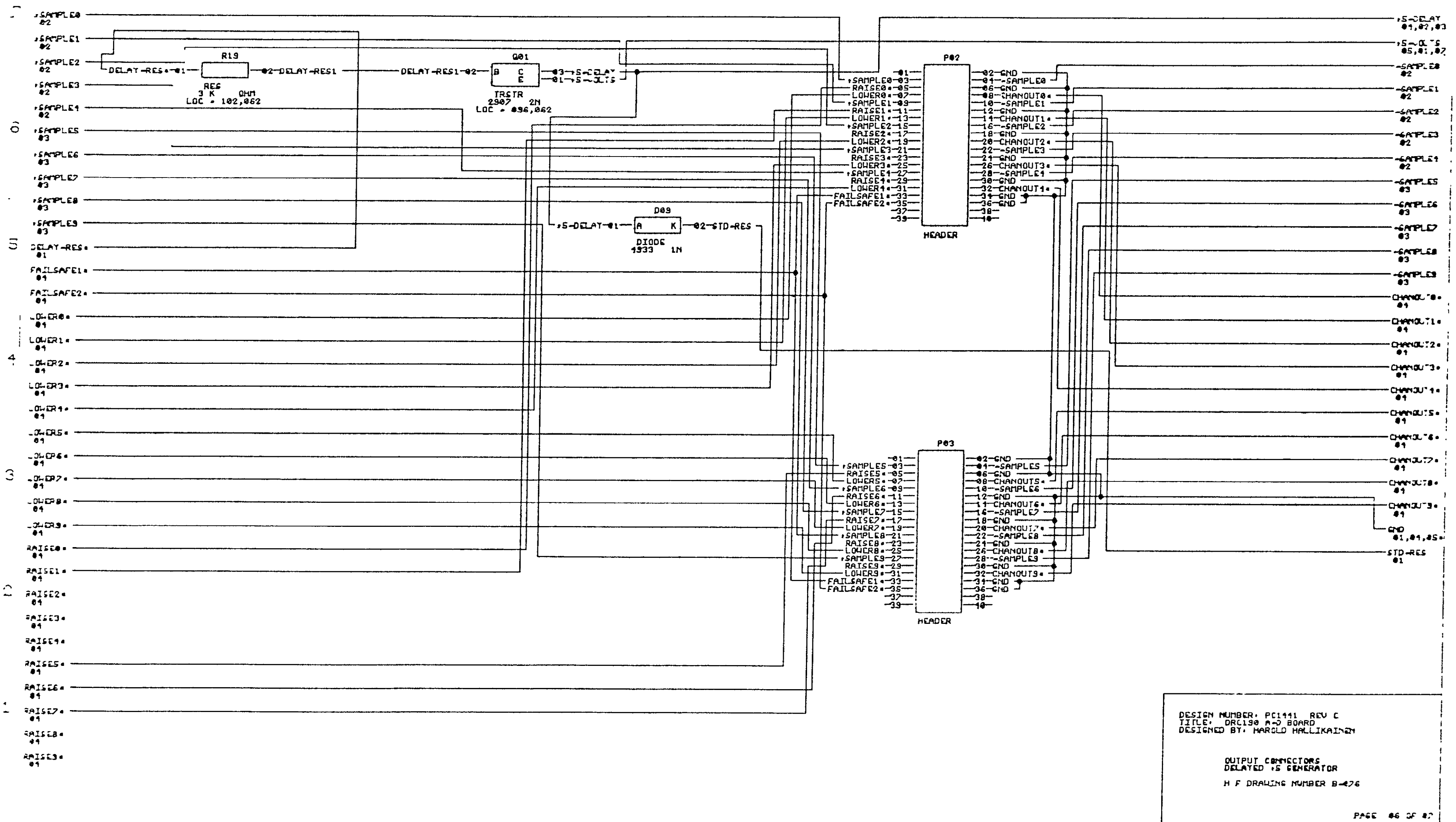


DESIGN NUMBER: PC1441 REV C
TITLE: DRC190 A-D BOARD
DESIGNED BY: HAROLD HALLIKAINEN

A-D CONVERTER

H F DRAWING NUMBER B-075

PAGE 05 OF 07

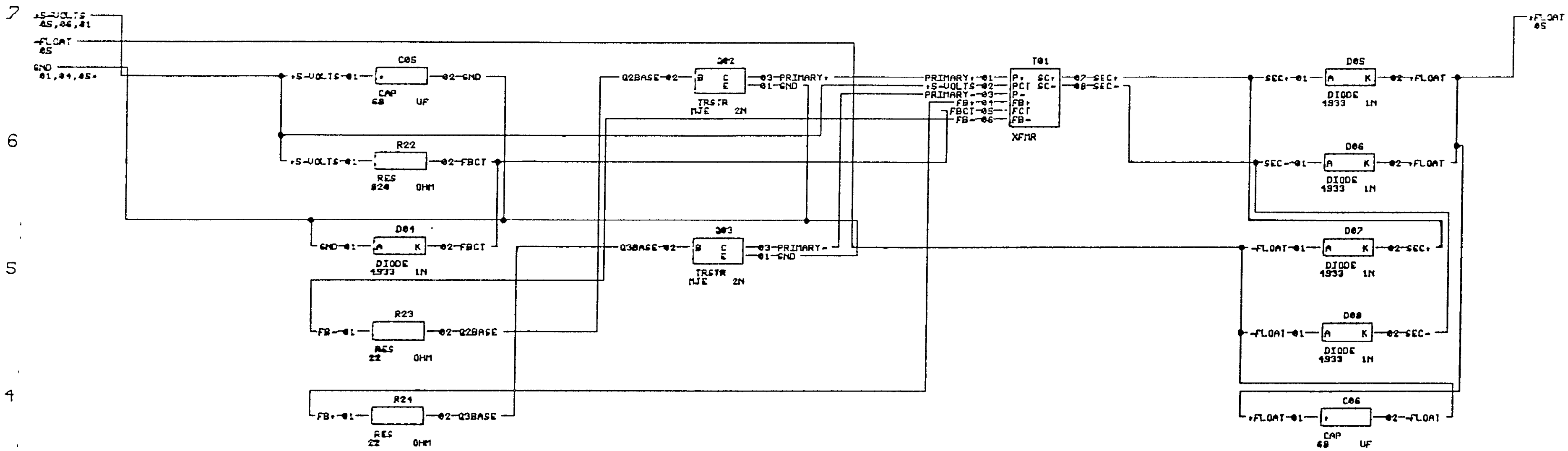


DESIGN NUMBER: PC1441 REV C
TITLE: DRC190 A-D BOARD
DESIGNED BY: HAROLD HALLIKAINEN

OUTPUT CONNECTORS
DELATED +5 GENERATOR

H F DRAWING NUMBER B-476

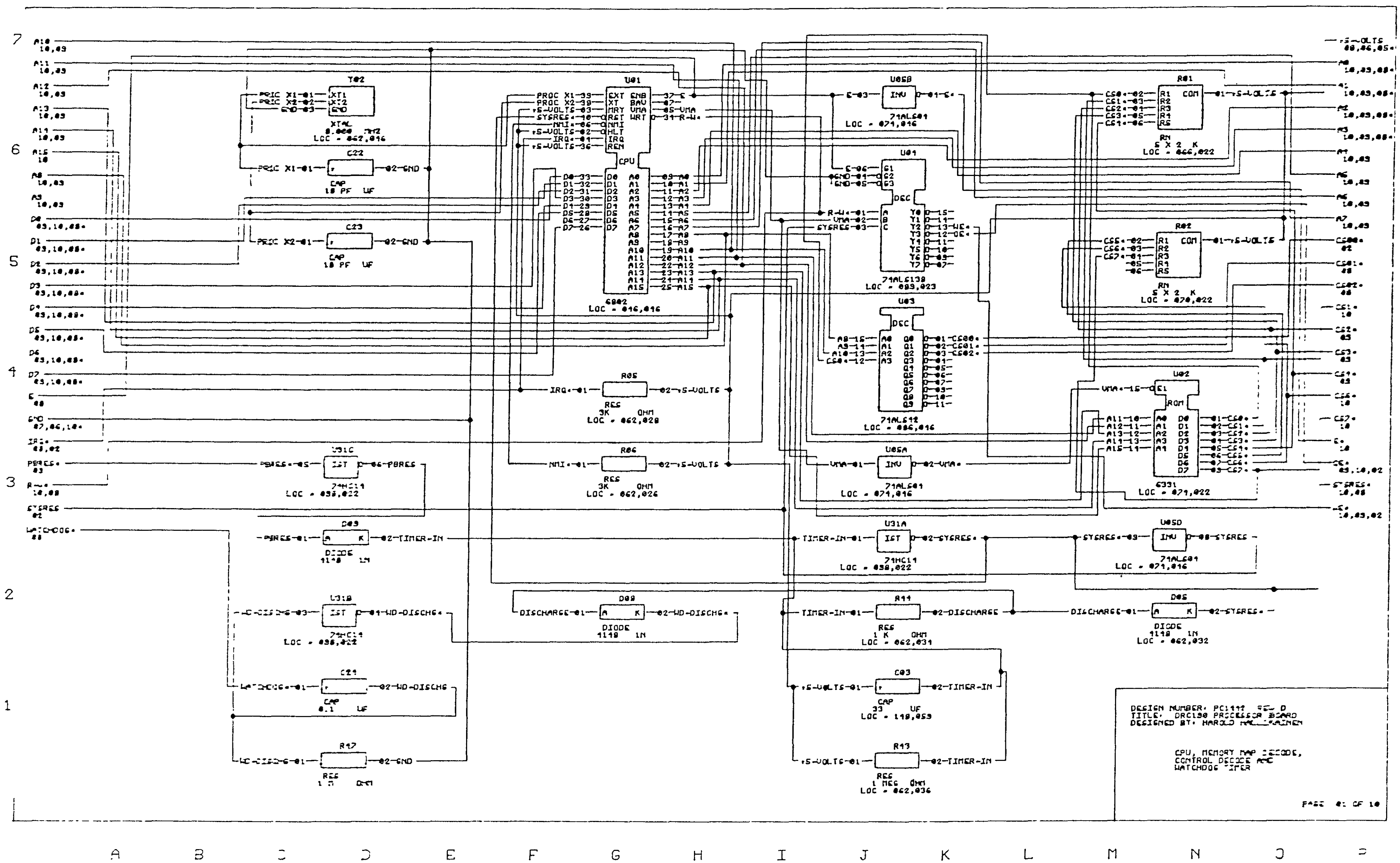
PAGE 06 OF 07

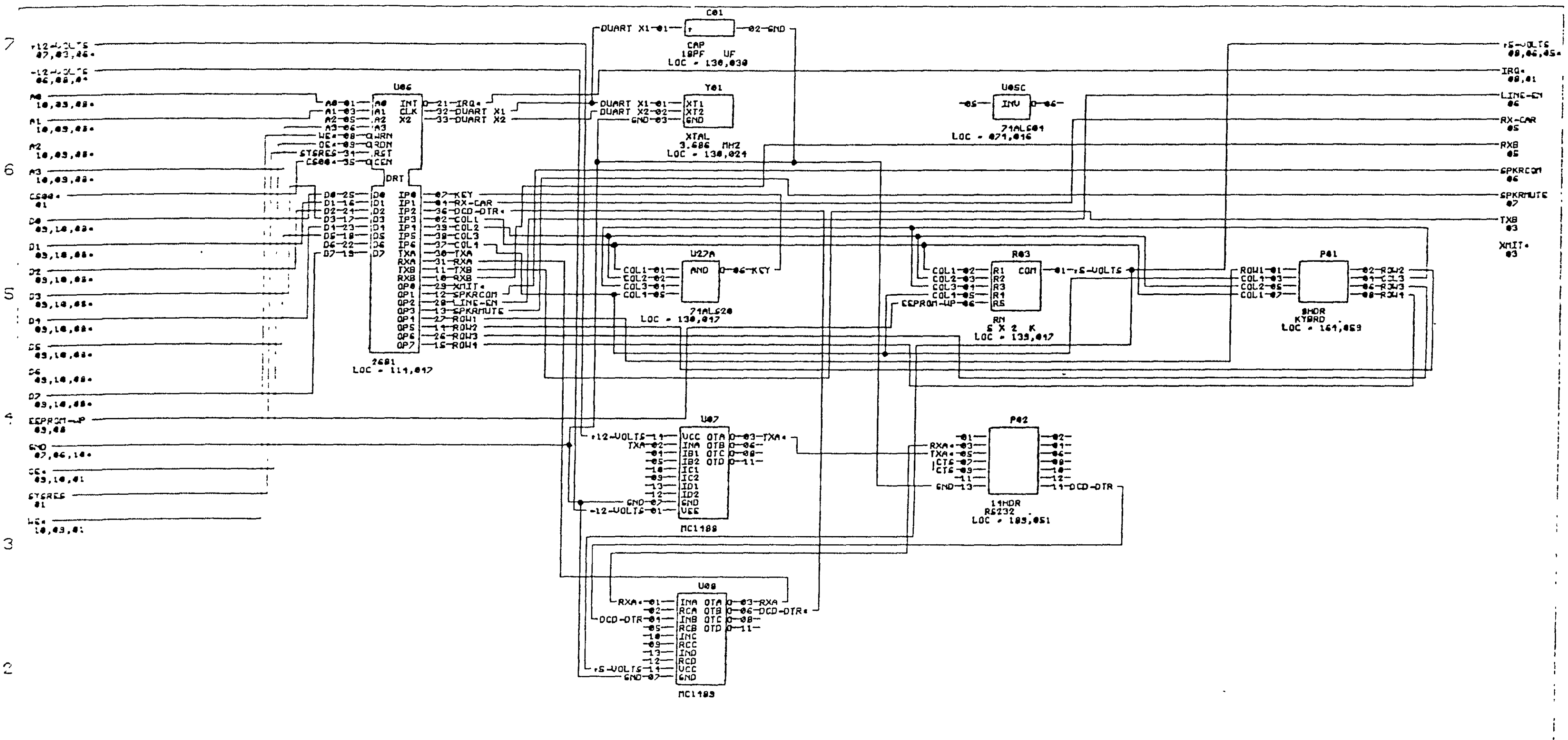


DESIGN NUMBER: PC1441 REV C
TITLE: DRC190 A-C BOARD
DESIGNED BY: HAROLD HALLIKAINEN

+5 VDC TO FLOATING 12 VDC CONVERTER. BOARD CAN USE EITHER THIS CIRCUIT OR WALL XB1119 CONVERTER

PAGE 07 OF 07

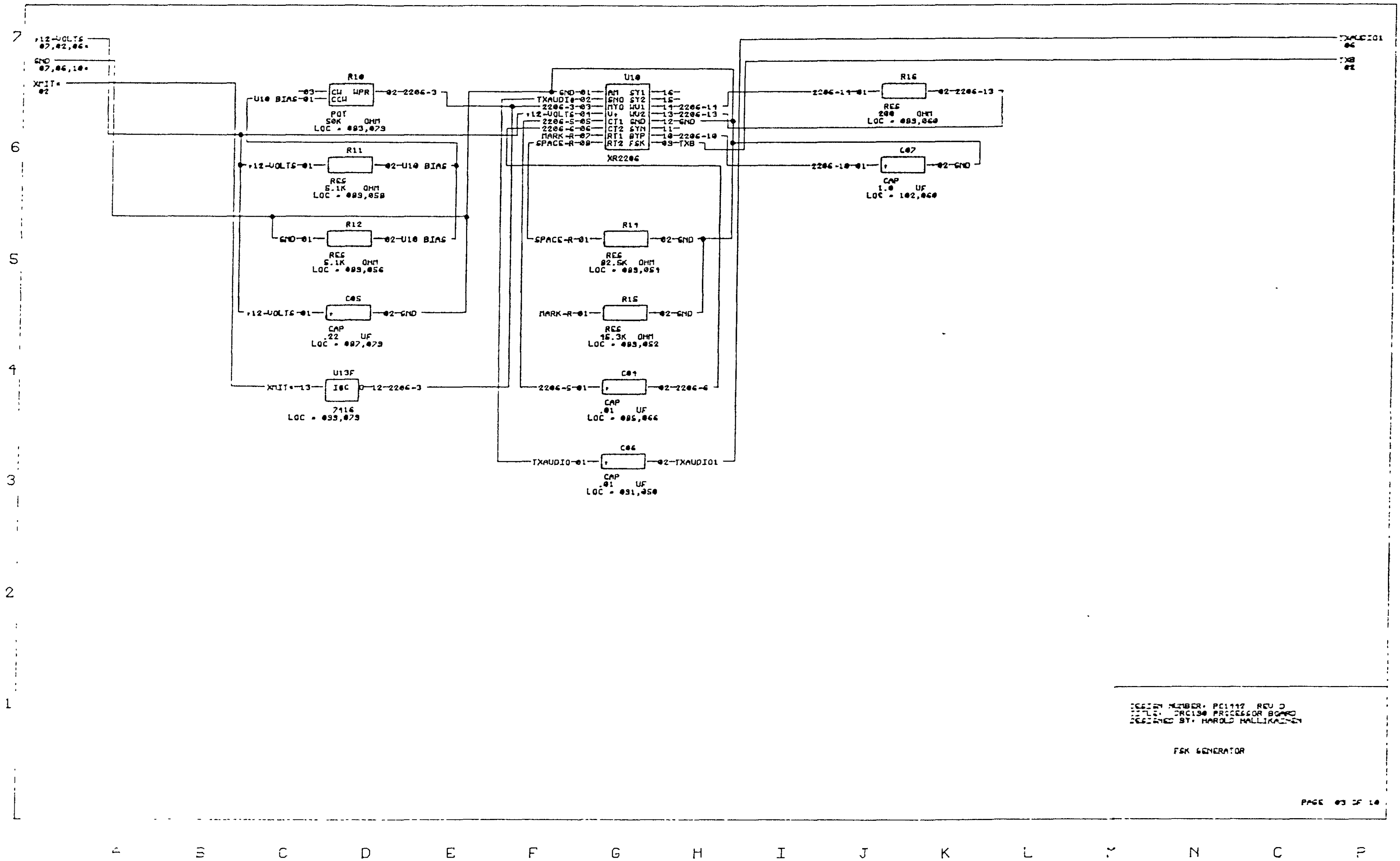

U01
CPU
6802
LOC = 016,016
U04
DEC
74ALS138
LOC = 083,023
U03
DEC
74ALS12
LOC = 086,016
U02
ROM
6331
LOC = 071,022
R01
RN
5 X 2 K
LOC = 066,022
R02
RN
5 X 2 K
LOC = 070,022
Y02
XTAL
8.000 MHZ
LOC = 062,016
C22
CAP
18 PF UF
C23
CAP
18 PF UF
R05
RES
3K OHM
LOC = 062,028
R06
RES
3K OHM
LOC = 062,026
U31C
IST
74HC14
LOC = 058,022
U31A
IST
74HC14
LOC = 058,022
U31B
IST
74HC14
LOC = 058,022
U05B
INV
74ALS04
LOC = 071,016
U05A
INV
74ALS04
LOC = 071,016
U05D
INV
74ALS04
LOC = 071,016
D03
DIODE
4148 1N
D09
DIODE
4148 1N
D05
DIODE
4148 1N
LOC = 062,032
R44
RES
1 K OHM
LOC = 062,034
C03
CAP
33 UF
LOC = 119,053
R43
RES
1 MEG OHM
LOC = 062,036
C21
CAP
0.1 UF
R47
RES
DESIGN NUMBER: PC1497 REV D
TITLE: DRC190 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN
CPU, MEMORY MAP DECODE,
CONTROL DECODE AND
WATCHDOG TIMER
PAGE 01 OF 10



DESIGN NUMBER: PC1442 REV D
TITLE: DRC190 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN

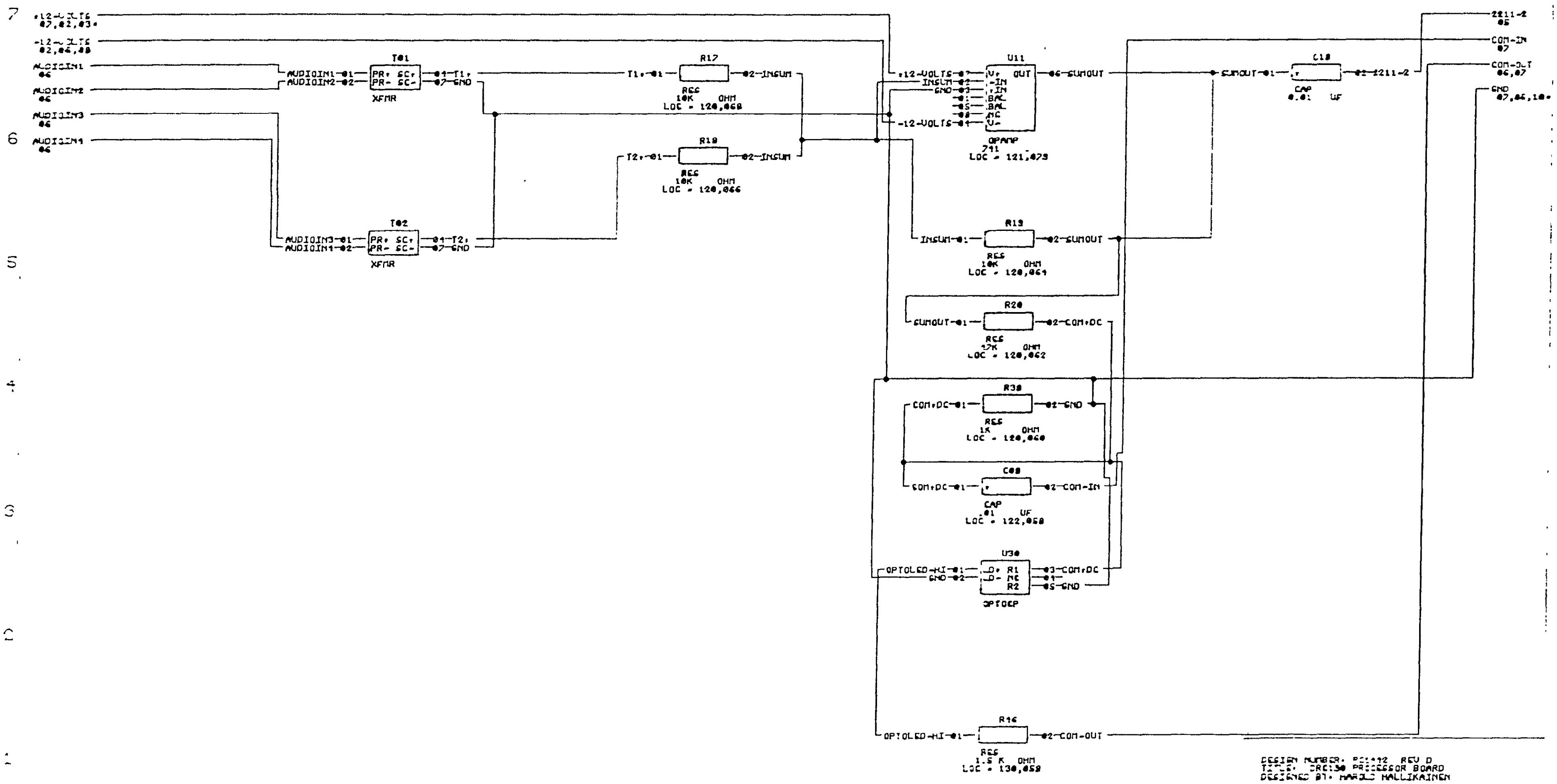
DUART, SYS CLOCK, KBD
AND RS232 INTERFACE,
REAL TIME CLOCK.

PAGE 02 OF 10



DESIGN NUMBER: PC1[illegible] REV D
TITLE: CRC190 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN

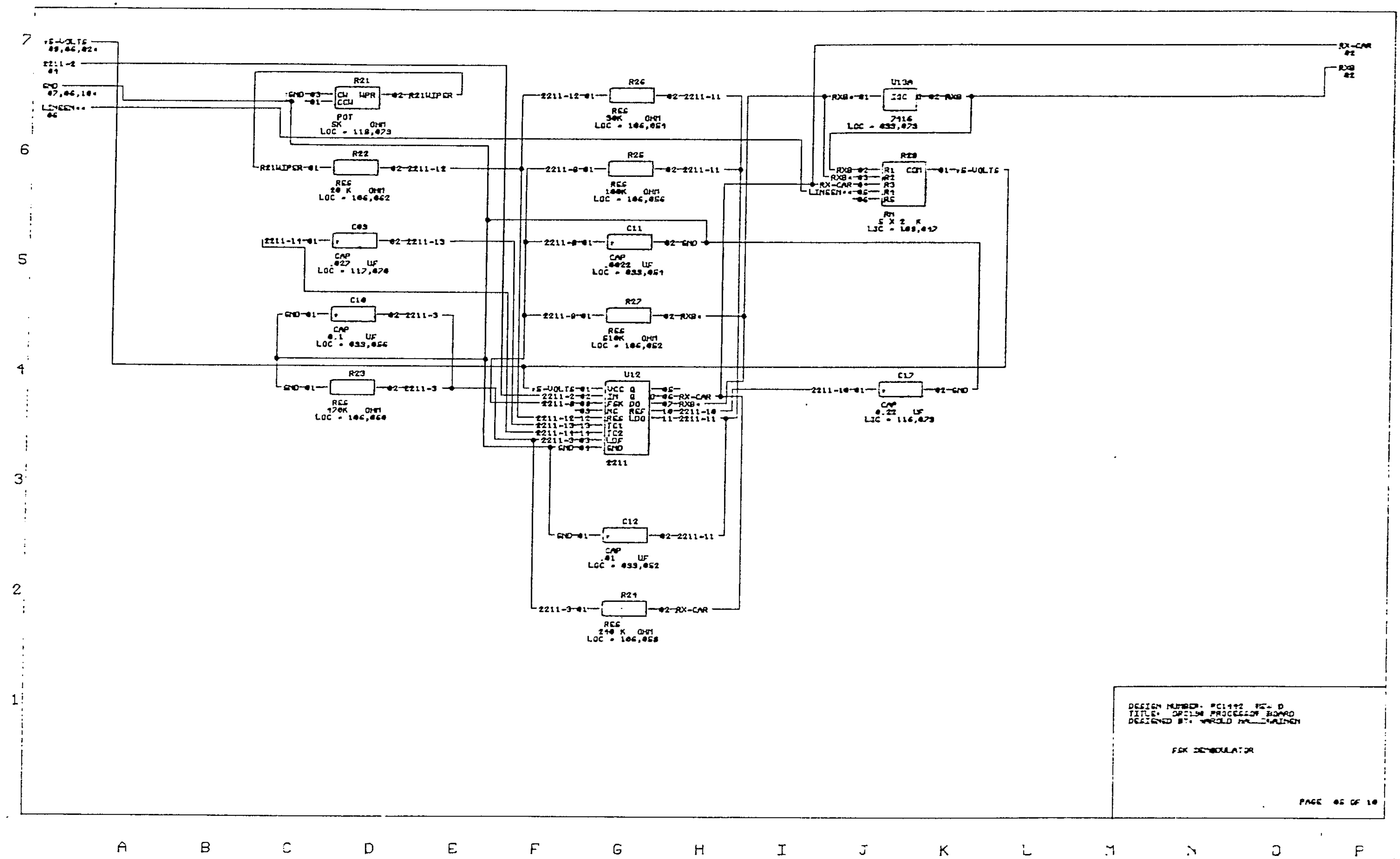
FSK GENERATOR

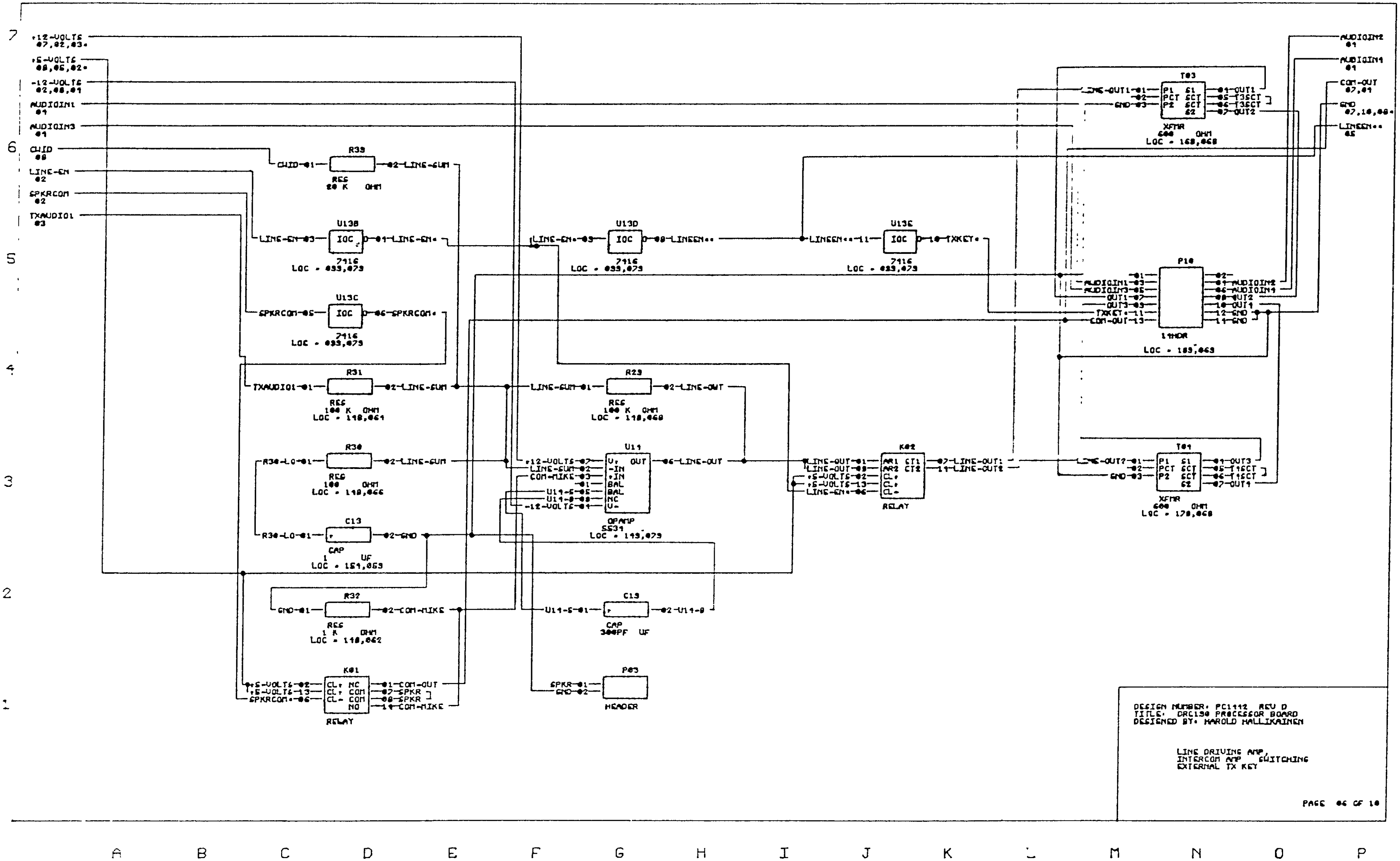


DESIGN NUMBER: P21492 REV D
TITLE: CRC150 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN

INPUT SUMMING AMPLIFIER

A B C D E F G H I J K L M N O P

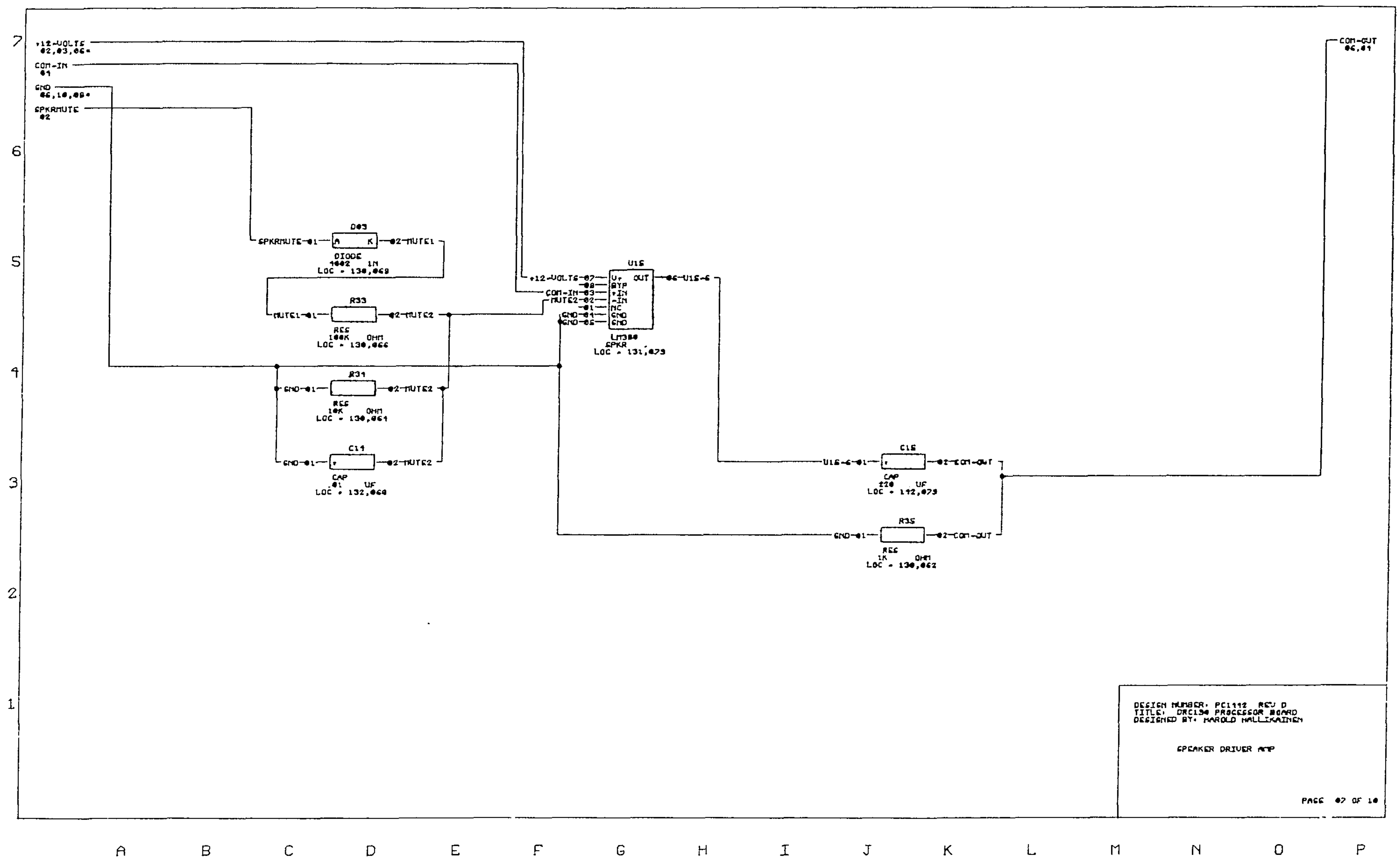

R21
R22
C03
C10
R23
R26
R25
C11
R27
U12
2211
C12
R24
U13A
R28
C17
FSK DEMODULATOR
PAGE 05 OF 10



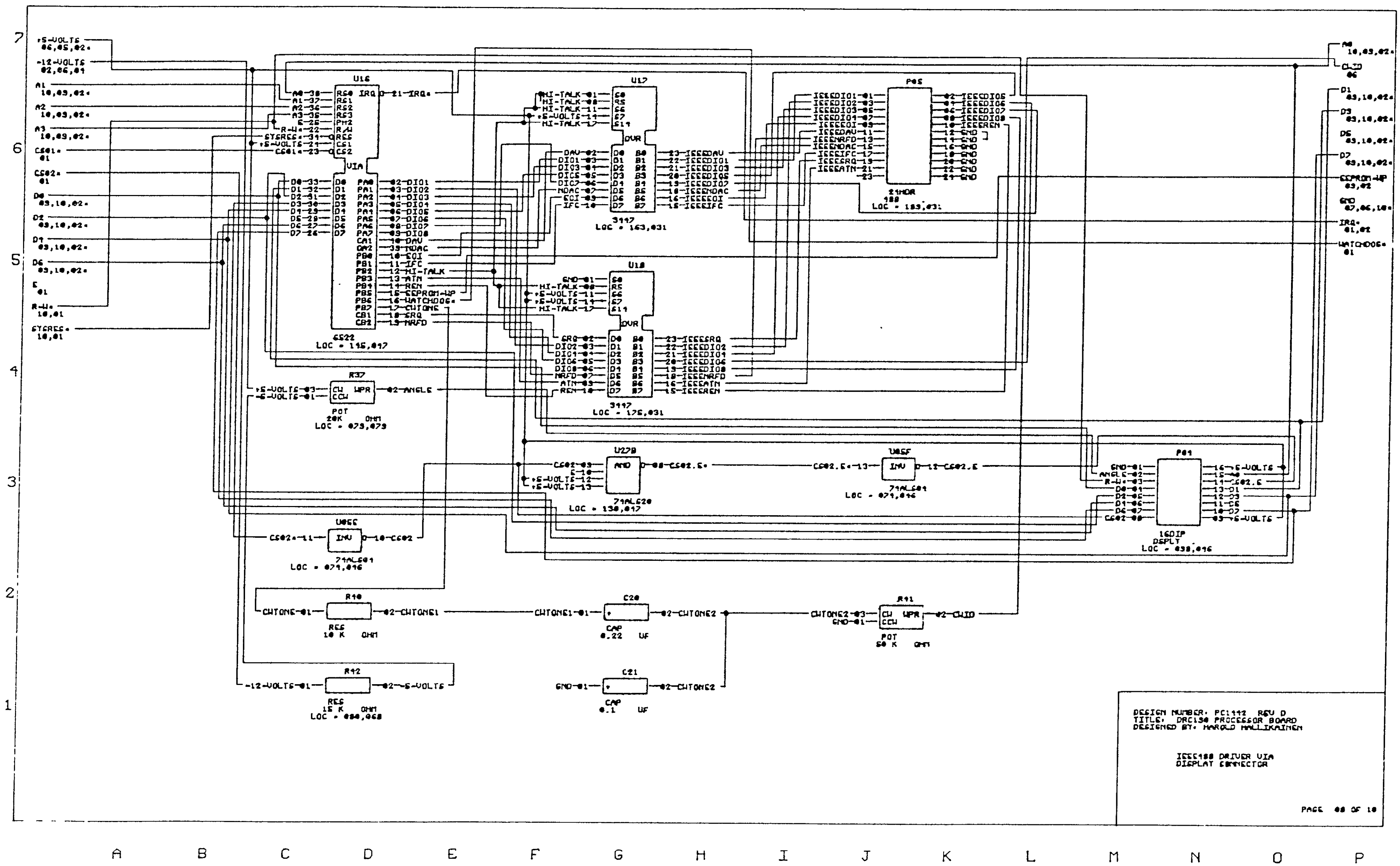
DESIGN NUMBER: PC1442 REV D
TITLE: DRC190 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN

LINE DRIVING AMP,
INTERCOM AMP SWITCHING
EXTERNAL TX KEY

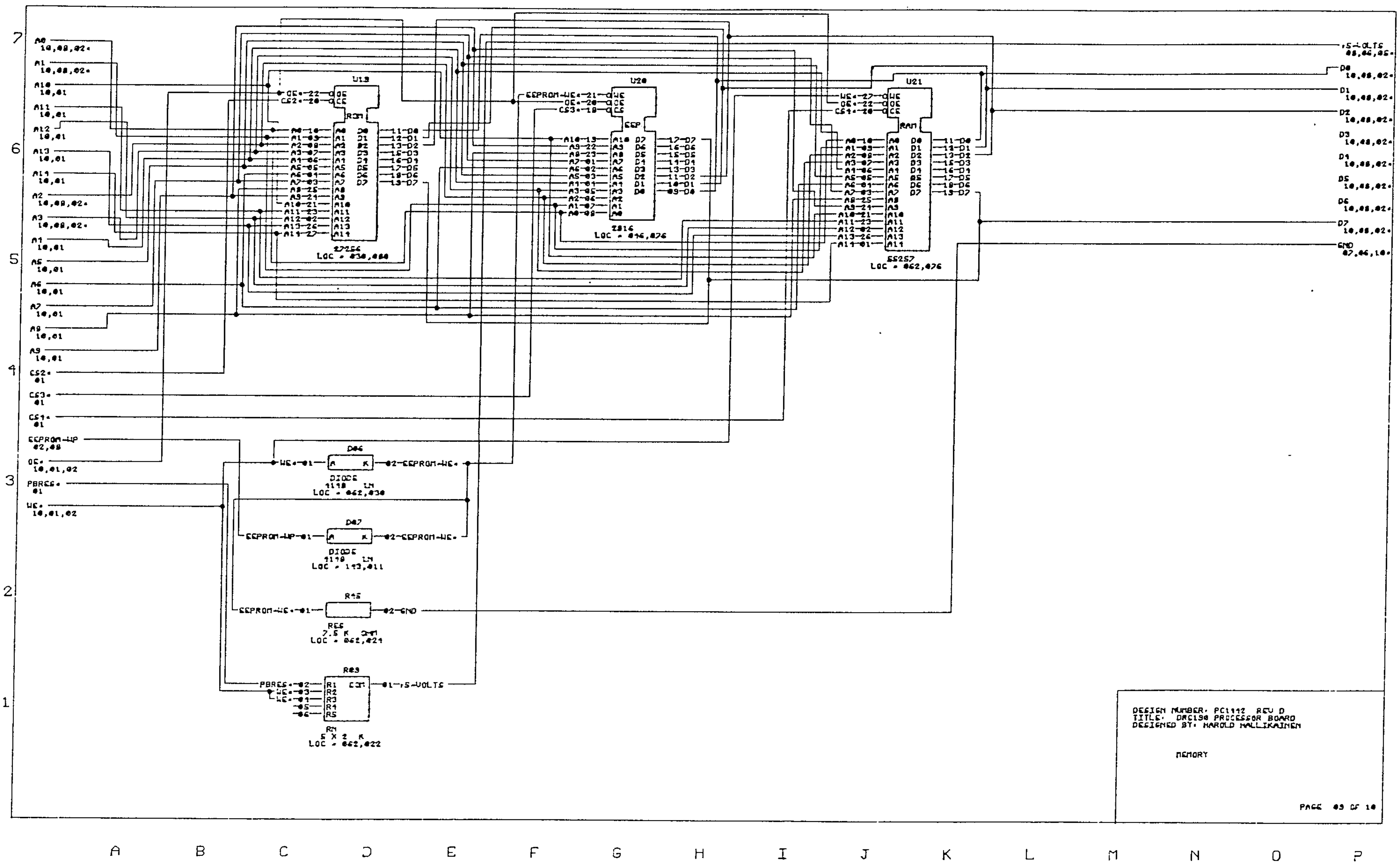
PAGE 06 OF 10

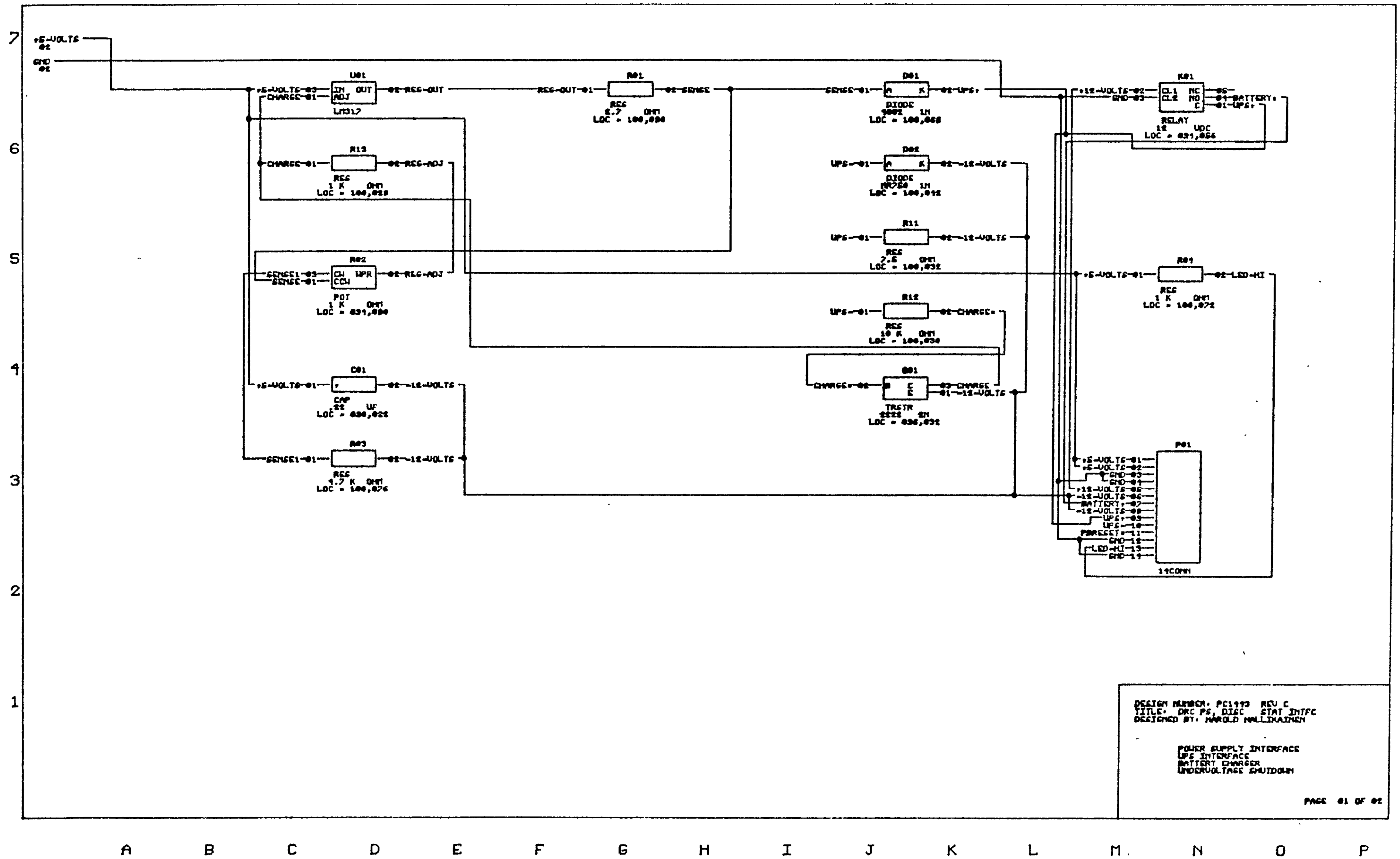


+12-VOLTS 02,03,06*
COM-IN 04
GND 06,10,09*
SPKRMUTE 02
D03
SPKRMUTE-01 A K 02-MUTE1
DIODE
4002 1N
LOC = 130,069
R33
MUTE1-01 02-MUTE2
RES
100K OHM
LOC = 130,066
R34
GND-01 02-MUTE2
RES
10K OHM
LOC = 130,064
C14
GND-01 02-MUTE2
CAP
.01 UF
LOC = 132,060
U15
+12-VOLTS-07 V+
08 BYP
COM-IN-03 +IN
MUTE2-02 -IN
01 NC
GND-04 GND
GND-05 GND
OUT 06-U15-6
LM380
SPKR
LOC = 131,073
C15
U15-6-01 02-COM-OUT
CAP
220 UF
LOC = 142,073
R35
GND-01 02-COM-OUT
RES
1K OHM
LOC = 130,062
COM-OUT 06,04
DESIGN NUMBER: PC1492 REV D
TITLE: DRC190 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN
SPEAKER DRIVER AMP
PAGE 07 OF 10
7
6
5
4
3
2
1
A B C D E F G H I J K L M N O P

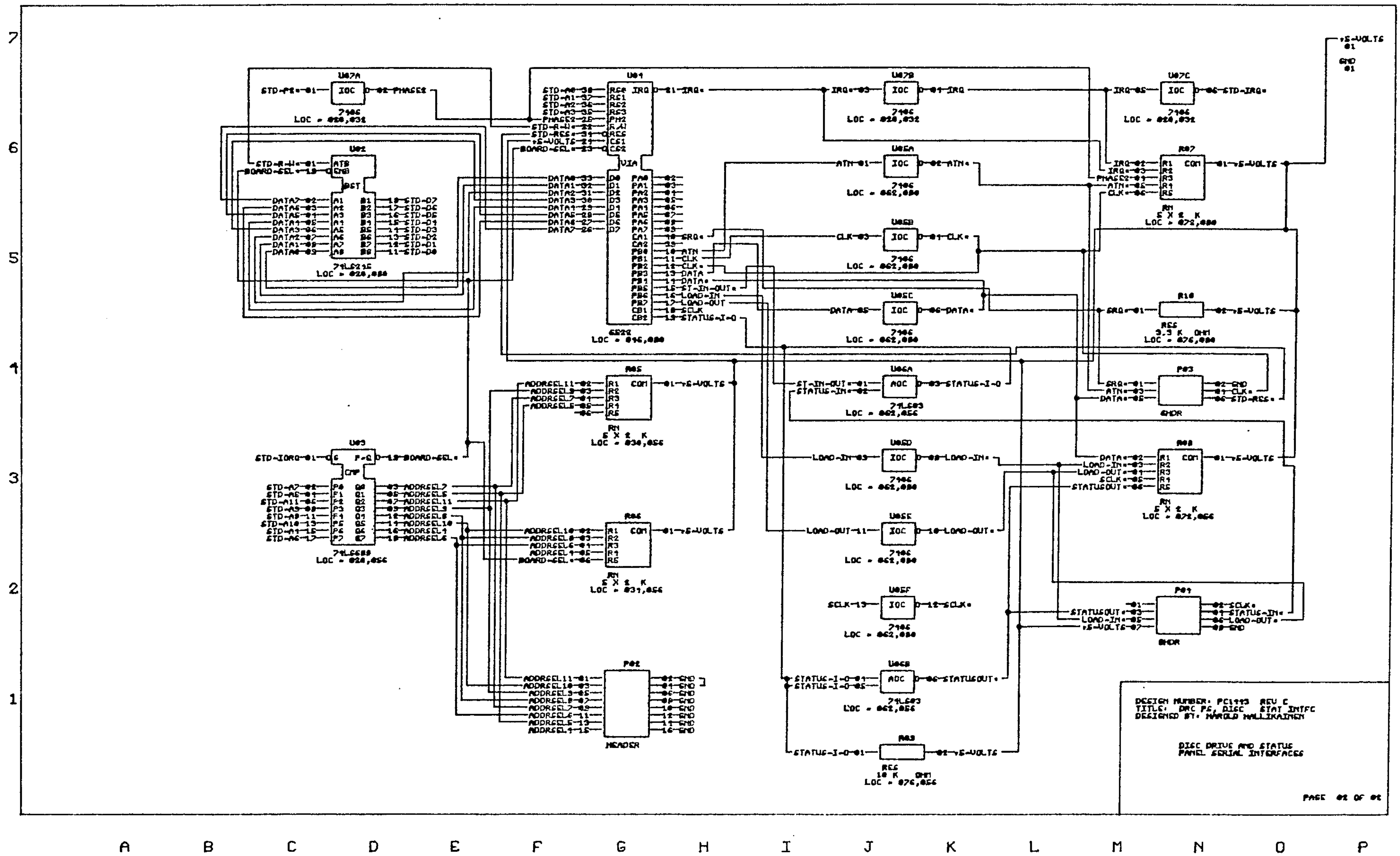


U16
6522
LOC = 115,017
U17
3447
LOC = 163,031
U18
3447
LOC = 175,031
P05
24HDR
488
LOC = 183,031
U27B
74ALS20
LOC = 130,017
U05F
74ALS04
LOC = 074,016
U05E
74ALS04
LOC = 074,016
P04
16DIP
DSPLY
LOC = 038,016
R37
POT
20K OHM
LOC = 073,073
R10
RES
10 K OHM
C20
CAP
0.22 UF
R41
POT
50 K OHM
R42
RES
15 K OHM
LOC = 050,068
C21
CAP
0.1 UF
DESIGN NUMBER: PC1442 REV D
TITLE: DRC130 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN
IEEE488 DRIVER VIA
DISPLAY CONNECTOR
PAGE 08 OF 10


U13
U20
U21
DIODE
DESIGN NUMBER: PC1142 REV D
TITLE: DRC190 PROCESSOR BOARD
DESIGNED BY: HAROLD HALLIKAINEN
MEMORY
PAGE 03 OF 10



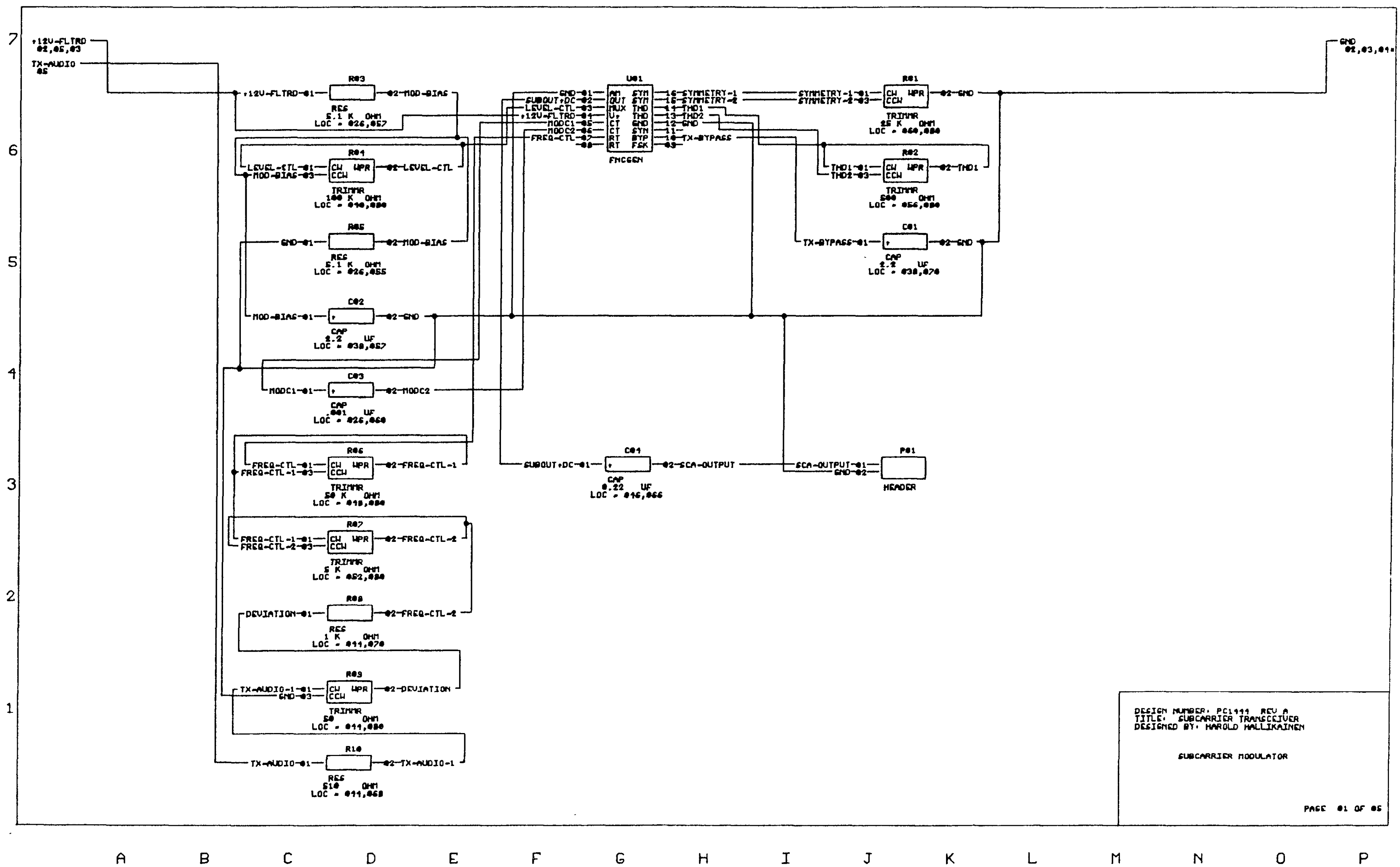
U01
LM317
R01
RES
2.7 OHM
D01
DIODE
D02
DIODE
R13
RES
1 K OHM
R02
POT
1 K OHM
C01
CAP
R03
RES
4.7 K OHM
R11
RES
R12
RES
10 K OHM
Q01
TRSTR
K01
RELAY
12 VDC
R04
RES
1 K OHM
P01
14CONN
DESIGN NUMBER: PC1993 REV C
TITLE: DRC PS, DISC STAT INTFC
DESIGNED BY: HAROLD HALLIKAINEN
POWER SUPPLY INTERFACE
UPS INTERFACE
BATTERY CHARGER
UNDERVOLTAGE SHUTDOWN
PAGE 01 OF 02



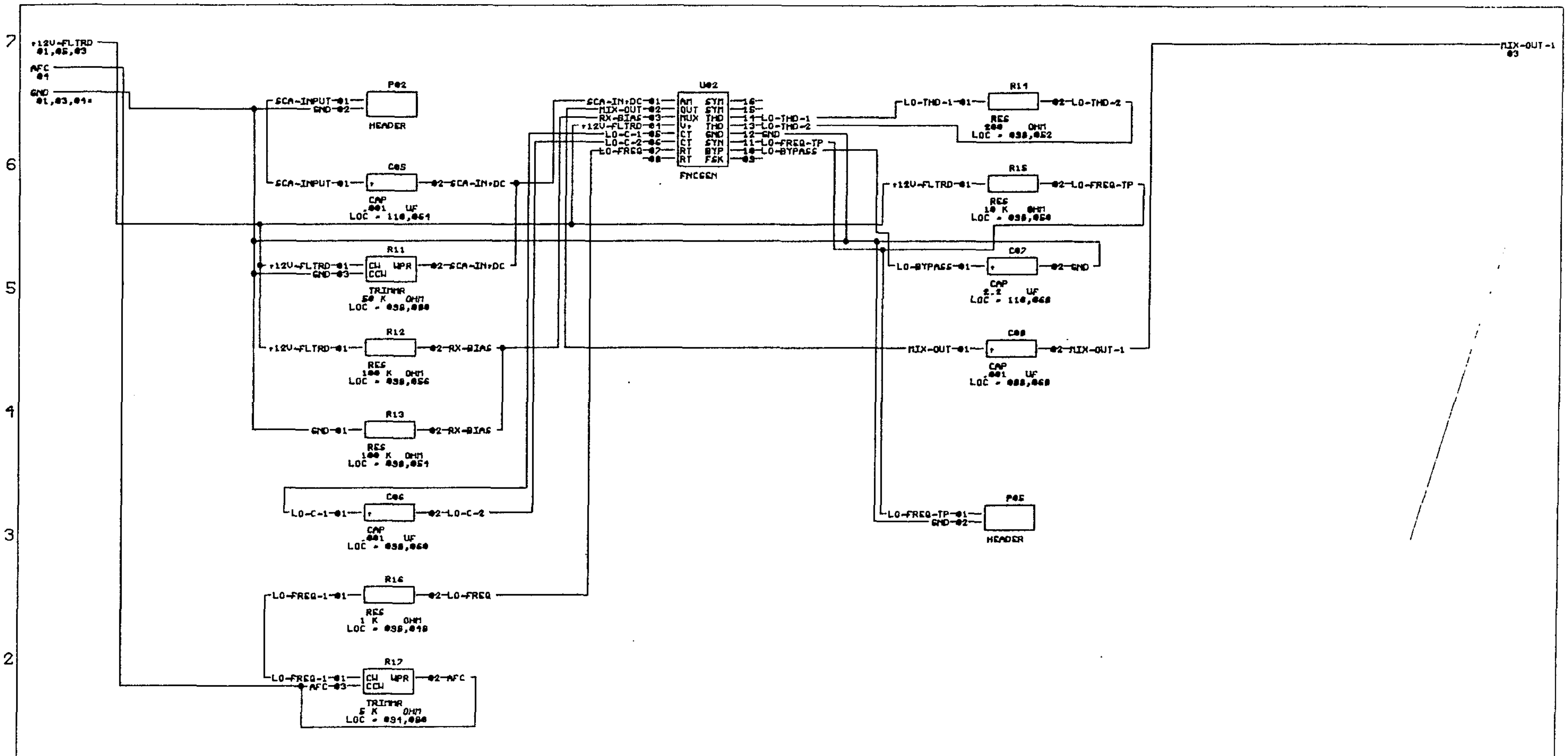
DESIGN NUMBER: PC1993 REV C
TITLE: DRC PS, DISC STAT INTFC
DESIGNED BY: HAROLD HALLIKAINEN

DISC DRIVE AND STATUS
PANEL SERIAL INTERFACES

PAGE 02 OF 02



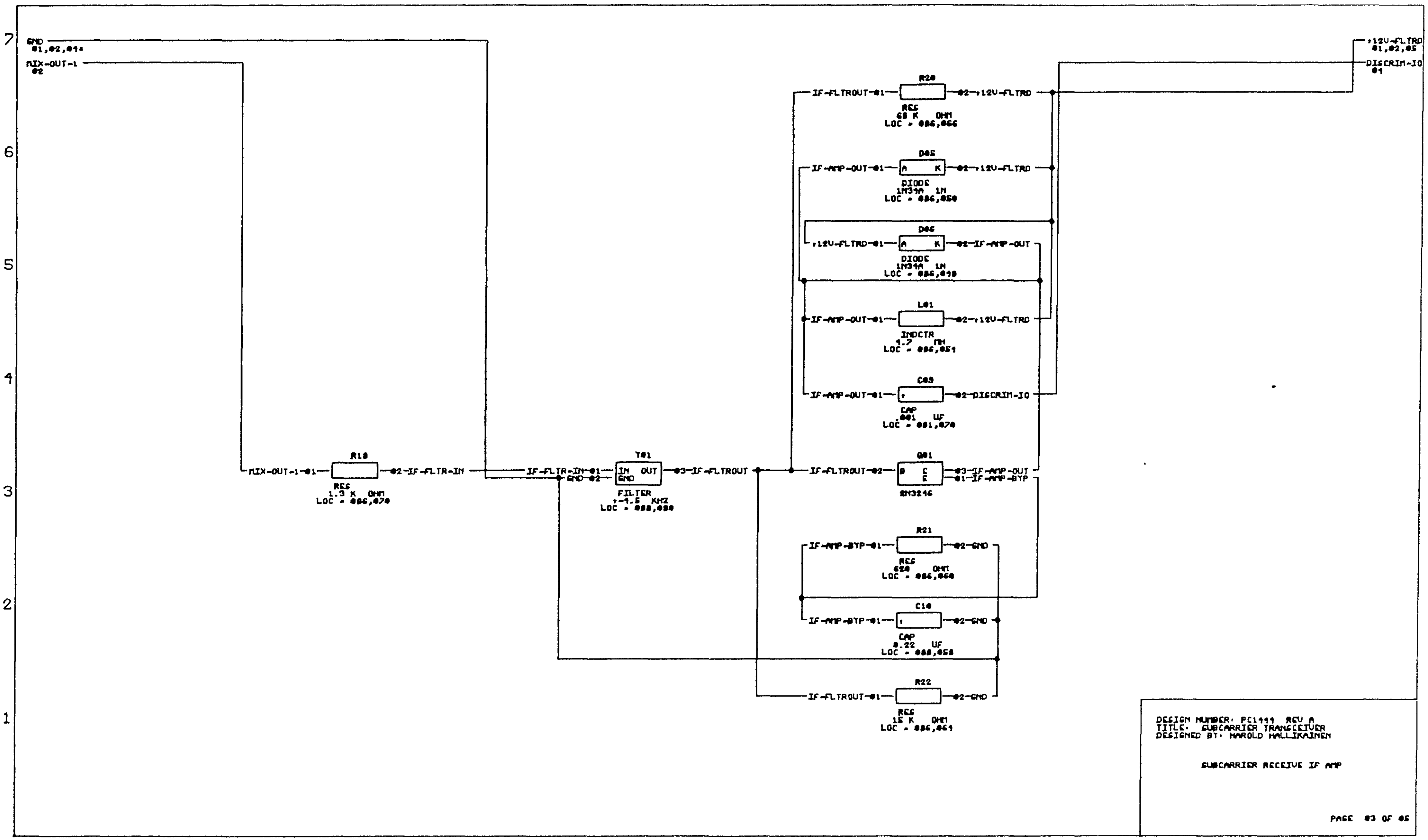
+12V-FLTRD 02,05,03
TX-AUDIO 05
GND 02,03,04
U01 FMGEN
R03 RES 5.1 K OHM
R04 TRIMMR 100 K OHM
R05 RES 5.1 K OHM
C02 CAP 2.2 UF
C03 CAP .001 UF
R06 TRIMMR 50 K OHM
R07 TRIMMR 5 K OHM
R08 RES 1 K OHM
R09 TRIMMR 50 OHM
R10 RES 510 OHM
C04 CAP 0.22 UF
P01 HEADER
R01 TRIMMR 25 K OHM
R02 TRIMMR 500 OHM
C01 CAP 2.2 UF
DESIGN NUMBER: PC1444 REV A
TITLE: SUBCARRIER TRANSCEIVER
DESIGNED BY: HAROLD HALLIKAINEN
SUBCARRIER MODULATOR
PAGE 01 OF 05

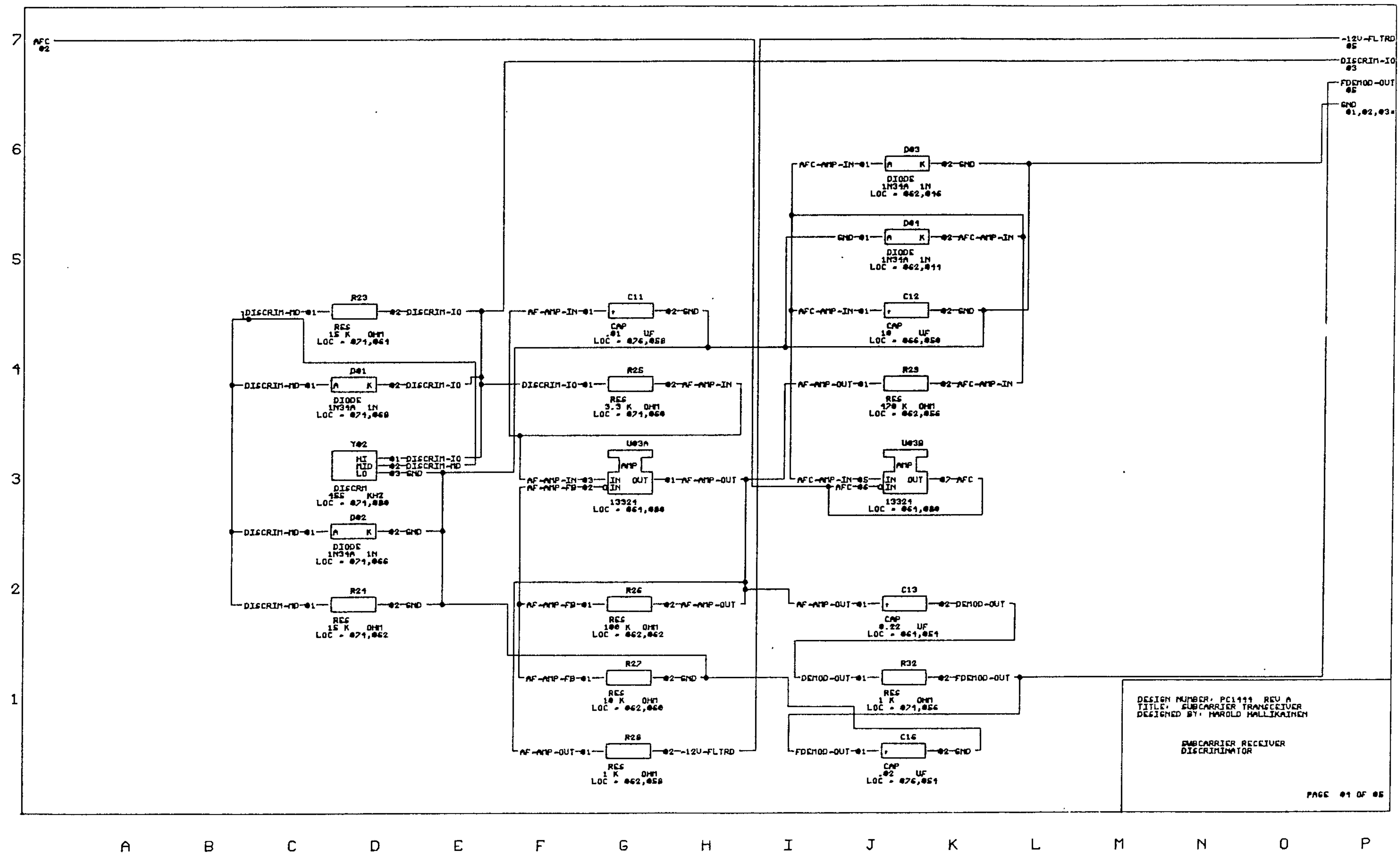


DESIGN NUMBER: PC1444 REV A
TITLE: SUBCARRIER TRANSCEIVER
DESIGNED BY: HAROLD HALLIKAINEN

SUBCARRIER RECEIVE MIXER

PAGE 02 OF 05

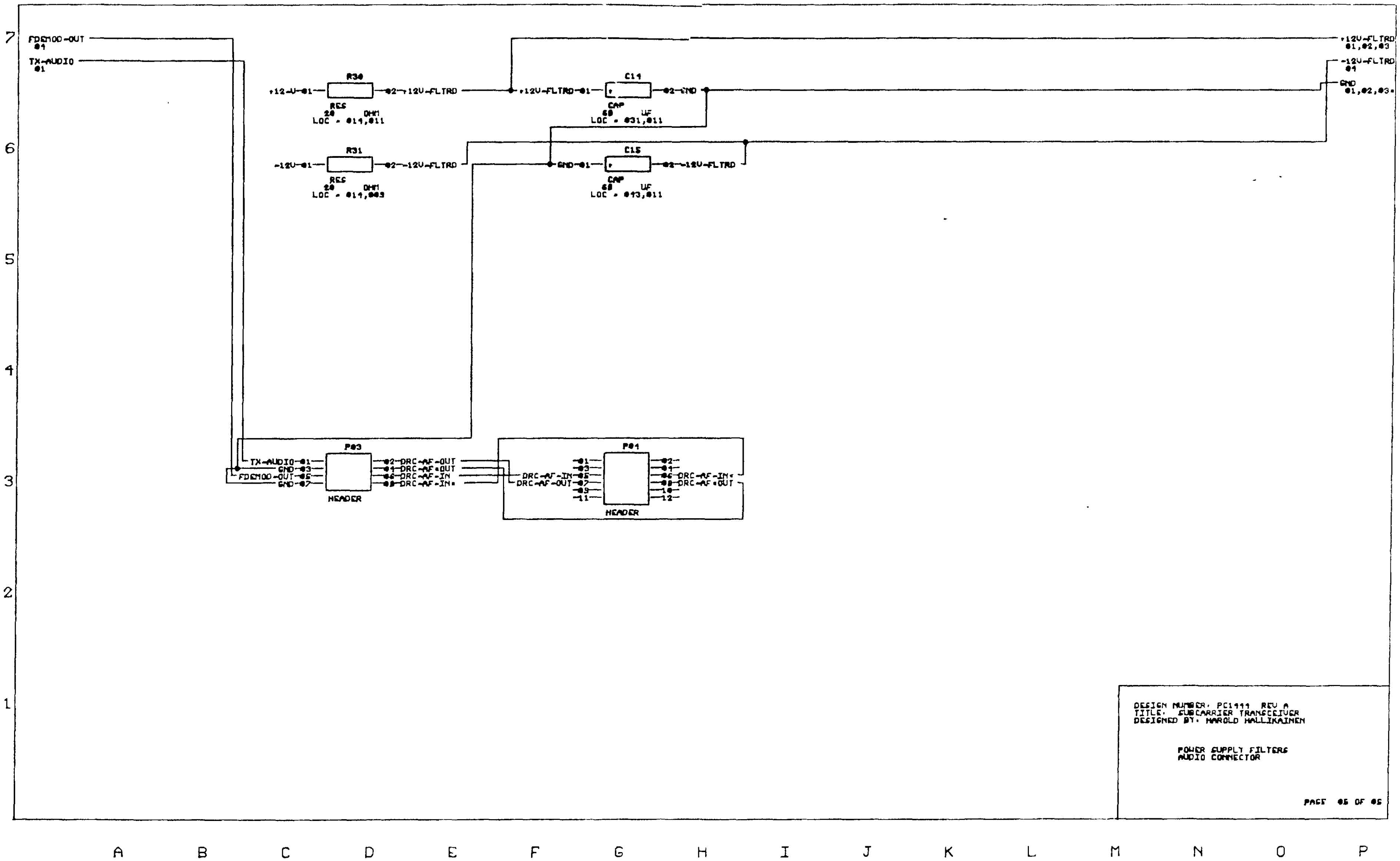

GND
01,02,04*
MIX-OUT-1
02
R18
MIX-OUT-1-01
02-IF-FLTR-IN
RES
1.3 K OHM
LOC = 086,070
T01
IF-FLTR-IN-01
GND-02
IN OUT
GND
03-IF-FLTROUT
FILTER
+-4.5 KHZ
LOC = 088,080
R20
IF-FLTROUT-01
02-+12V-FLTRD
RES
68 K OHM
LOC = 086,066
D05
IF-AMP-OUT-01
A K
02-+12V-FLTRD
DIODE
1N34A 1N
LOC = 086,050
D06
+12V-FLTRD-01
A K
02-IF-AMP-OUT
DIODE
1N34A 1N
LOC = 086,048
L01
IF-AMP-OUT-01
02-+12V-FLTRD
INDCTR
4.7 MH
LOC = 086,054
C09
IF-AMP-OUT-01
02-DISCRIM-IO
CAP
.001 UF
LOC = 081,070
Q01
IF-FLTROUT-02
B C
E
03-IF-AMP-OUT
01-IF-AMP-BYP
2N3246
R21
IF-AMP-BYP-01
02-GND
RES
620 OHM
LOC = 086,060
C10
IF-AMP-BYP-01
02-GND
CAP
0.22 UF
LOC = 088,058
R22
IF-FLTROUT-01
02-GND
RES
15 K OHM
LOC = 086,064
+12V-FLTRD
01,02,05
DISCRIM-IO
04
DESIGN NUMBER: PC1444 REV A
TITLE: SUBCARRIER TRANSCEIVER
DESIGNED BY: HAROLD HALLIKAINEN
SUBCARRIER RECEIVE IF AMP
PAGE 03 OF 05

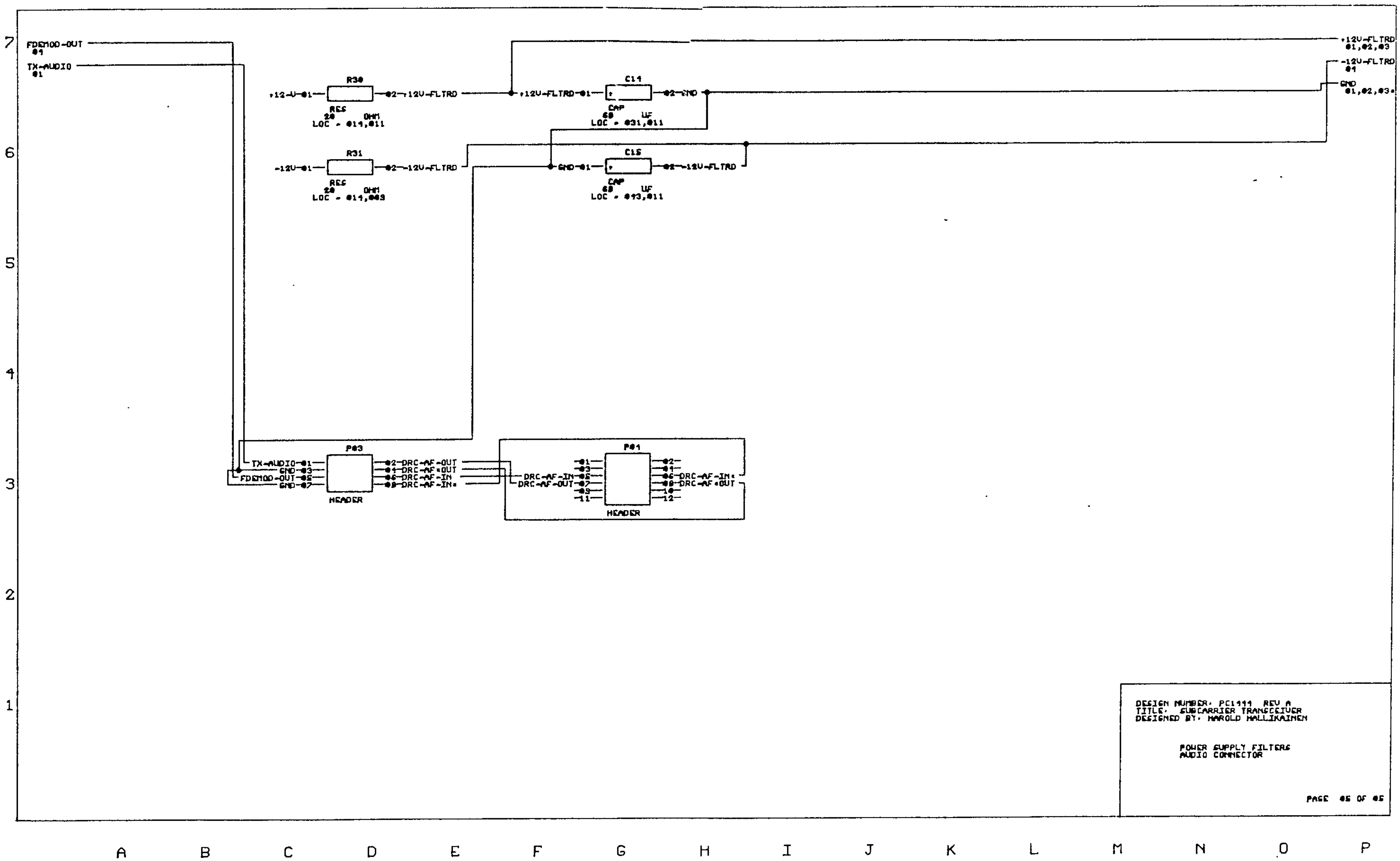


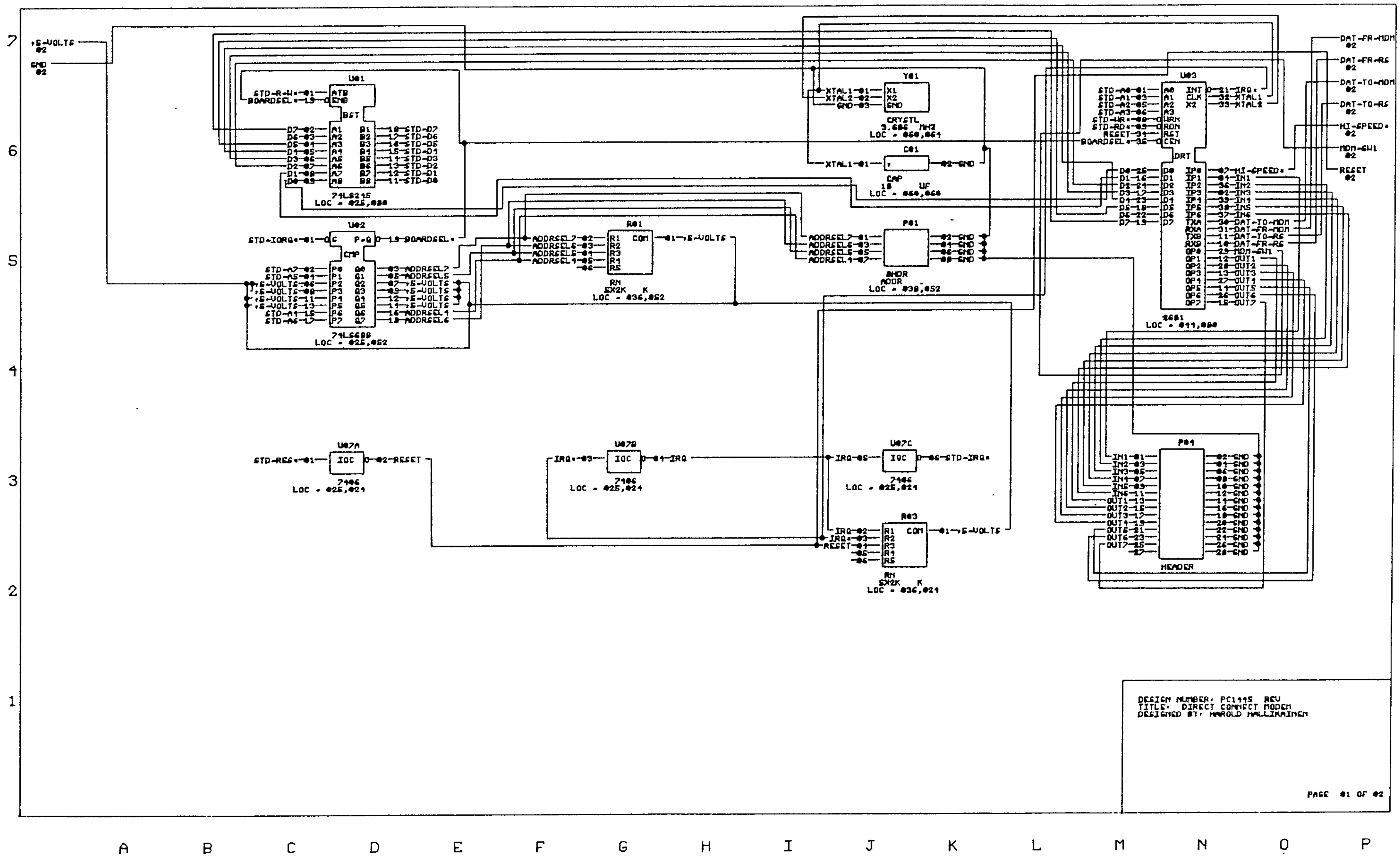
DESIGN NUMBER: PC1444 REV A
TITLE: SUBCARRIER TRANSCEIVER
DESIGNED BY: HAROLD HALLIKAINEN

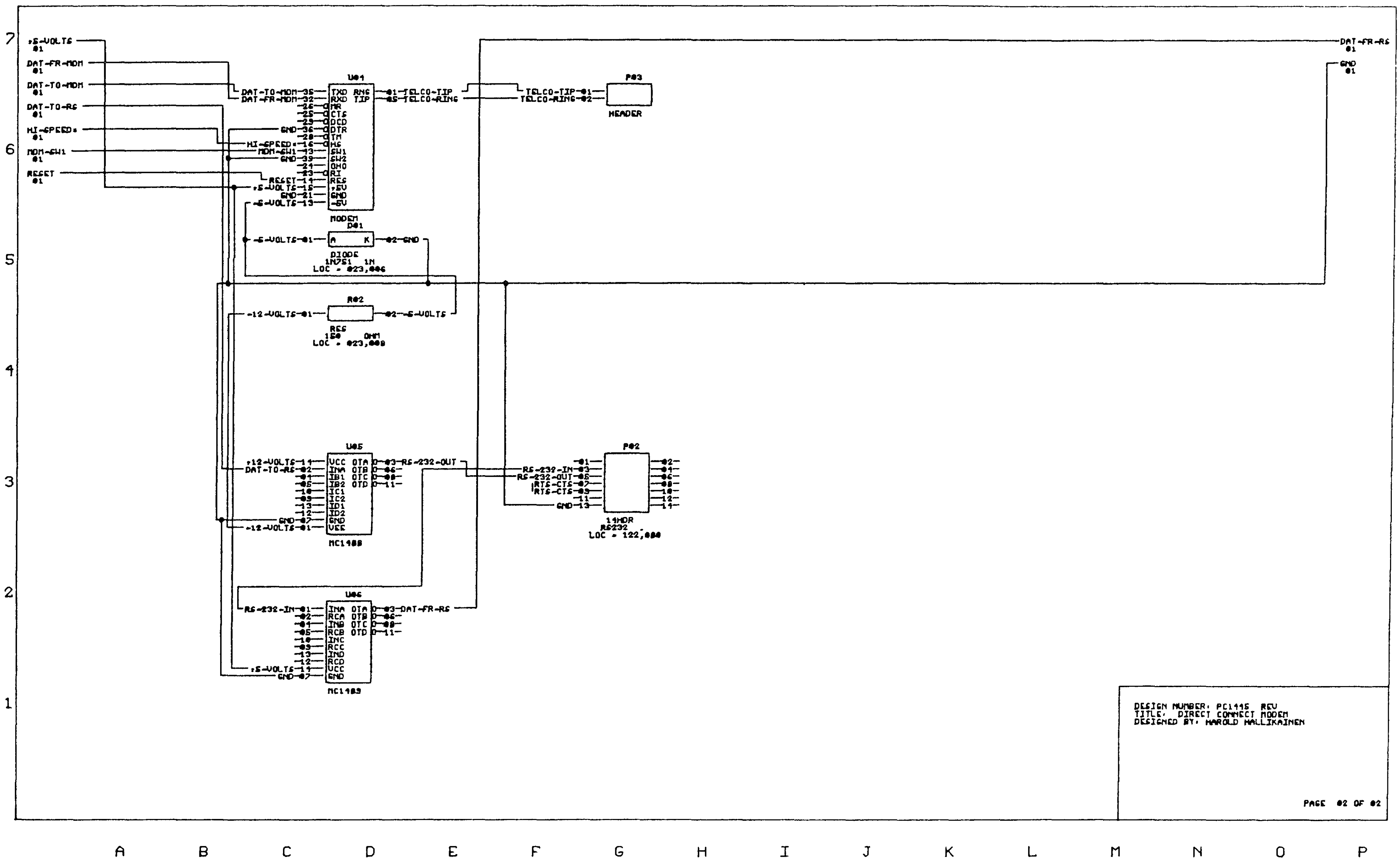
SUBCARRIER RECEIVER
DISCRIMINATOR

PAGE 04 OF 05

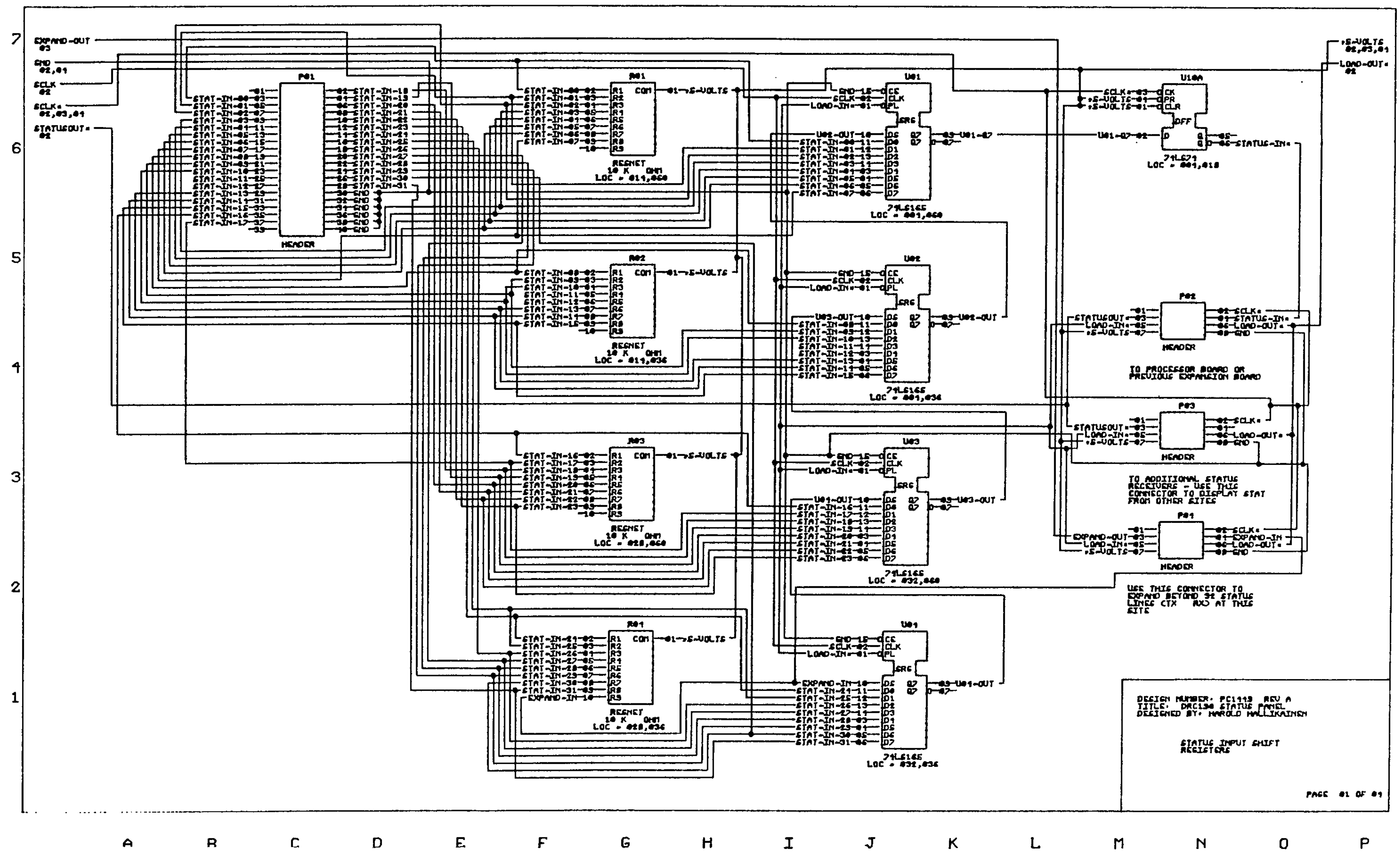

FDEMOD-OUT
TX-AUDIO
R30
+12-V-01
02-+12V-FLTRD
RES
20 OHM
LOC = 014,011
R31
-12V-01
02--12V-FLTRD
RES
20 OHM
LOC = 014,003
C14
+12V-FLTRD-01
02-GND
CAP
68 UF
LOC = 031,011
C15
GND-01
02--12V-FLTRD
CAP
68 UF
LOC = 013,011
+12V-FLTRD
01,02,03
-12V-FLTRD
01
GND
01,02,03
P03
TX-AUDIO-01
GND-03
FDEMOD-OUT-05
GND-07
02-DRC-AF-OUT
04-DRC-AF+OUT
06-DRC-AF-IN
08-DRC-AF-IN+
HEADER
P04
DRC-AF-IN-05
DRC-AF-OUT-07
06-DRC-AF-IN+
08-DRC-AF+OUT
HEADER
DESIGN NUMBER: PC1444 REV A
TITLE: SUBCARRIER TRANSCEIVER
DESIGNED BY: HAROLD HALLIKAINEN
POWER SUPPLY FILTERS
AUDIO CONNECTOR
PAGE 05 OF 05

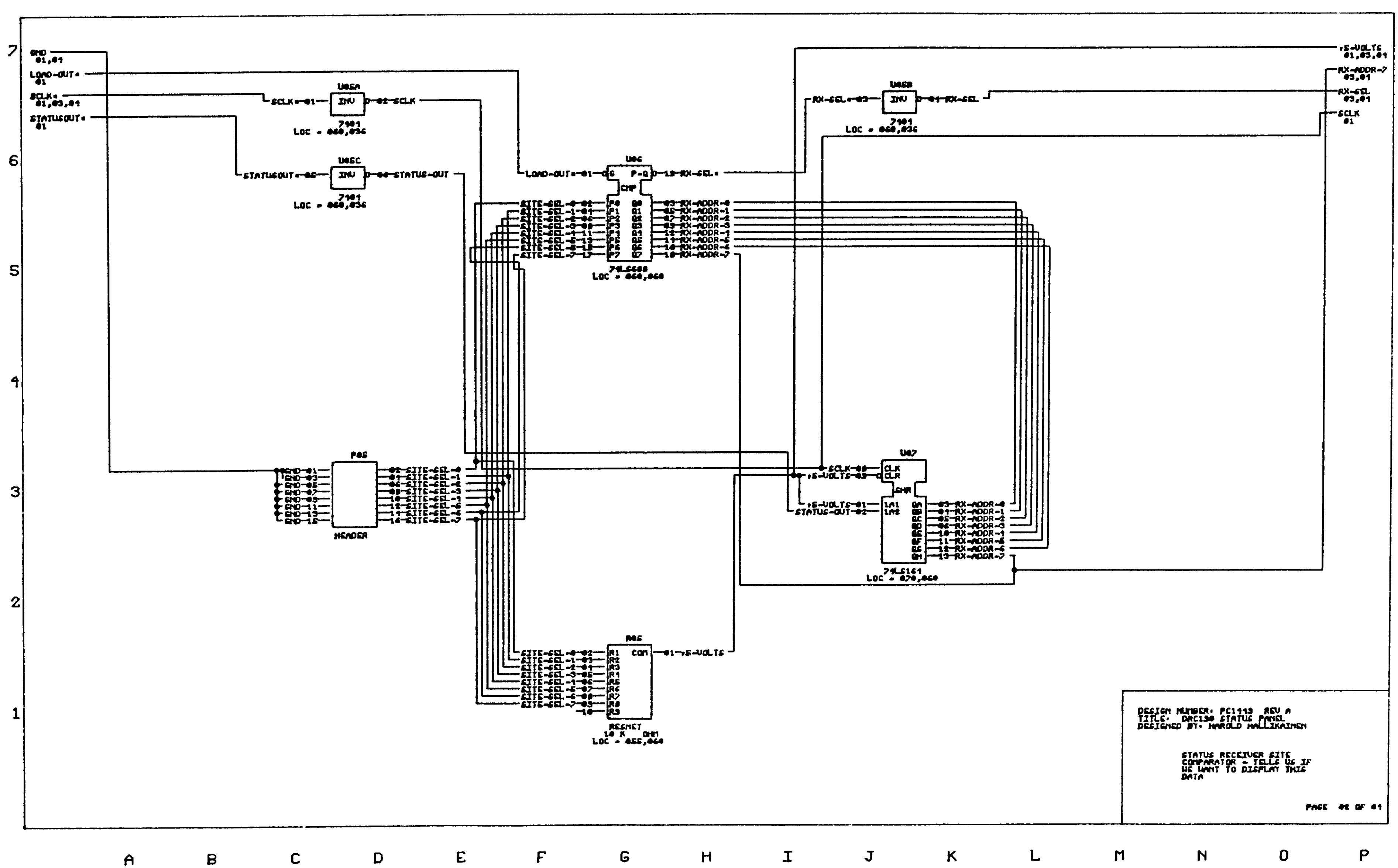

FDEMOD-OUT
01
TX-AUDIO
01
R30
+12-V-01
02-+12V-FLTRD
RES
20 OHM
LOC - 014,011
R31
-12V-01
02--12V-FLTRD
RES
20 OHM
LOC - 014,003
C14
+12V-FLTRD-01
02-GND
CAP
68 UF
LOC - 031,011
C15
GND-01
02--12V-FLTRD
CAP
68 UF
LOC - 043,011
+12V-FLTRD
01,02,03
-12V-FLTRD
01
GND
01,02,03
P03
TX-AUDIO-01
GND-03
FDEMOD-OUT-05
GND-07
02-DRC-AF-OUT
04-DRC-AF*OUT
06-DRC-AF-IN
08-DRC-AF-IN*
HEADER
P04
DRC-AF-IN-05
DRC-AF-OUT-07
06-DRC-AF-IN*
08-DRC-AF*OUT
HEADER
DESIGN NUMBER: PC1444 REV A
TITLE: SUBCARRIER TRANSCEIVER
DESIGNED BY: HAROLD HALLIKAINEN
POWER SUPPLY FILTERS
AUDIO CONNECTOR
PAGE 05 OF 05


U01
74LS245
LOC = 025,080
U02
74LS688
LOC = 025,052
R01
RN
5X2K K
LOC = 036,052
Y01
CRYSTL
3.686 MHZ
C01
CAP
10 UF
P01
ADDR
LOC = 039,052
U03
2681
LOC = 011,080
P04
HEADER
U07A
7406
LOC = 025,024
U07B
7406
LOC = 025,024
U07C
7406
LOC = 025,024
R03
RN
5X2K K
LOC = 036,024
+5-VOLTS
GND
DAT-FR-MDM
DAT-FR-RS
DAT-TO-MDM
DAT-TO-RS
HI-SPEED*
MDM-SW1
RESET
STD-RES*
STD-IRQ*
DESIGN NUMBER: PC1445 REV
TITLE: DIRECT CONNECT MODEM
DESIGNED BY: HAROLD HALLIKAINEN
PAGE 01 OF 02

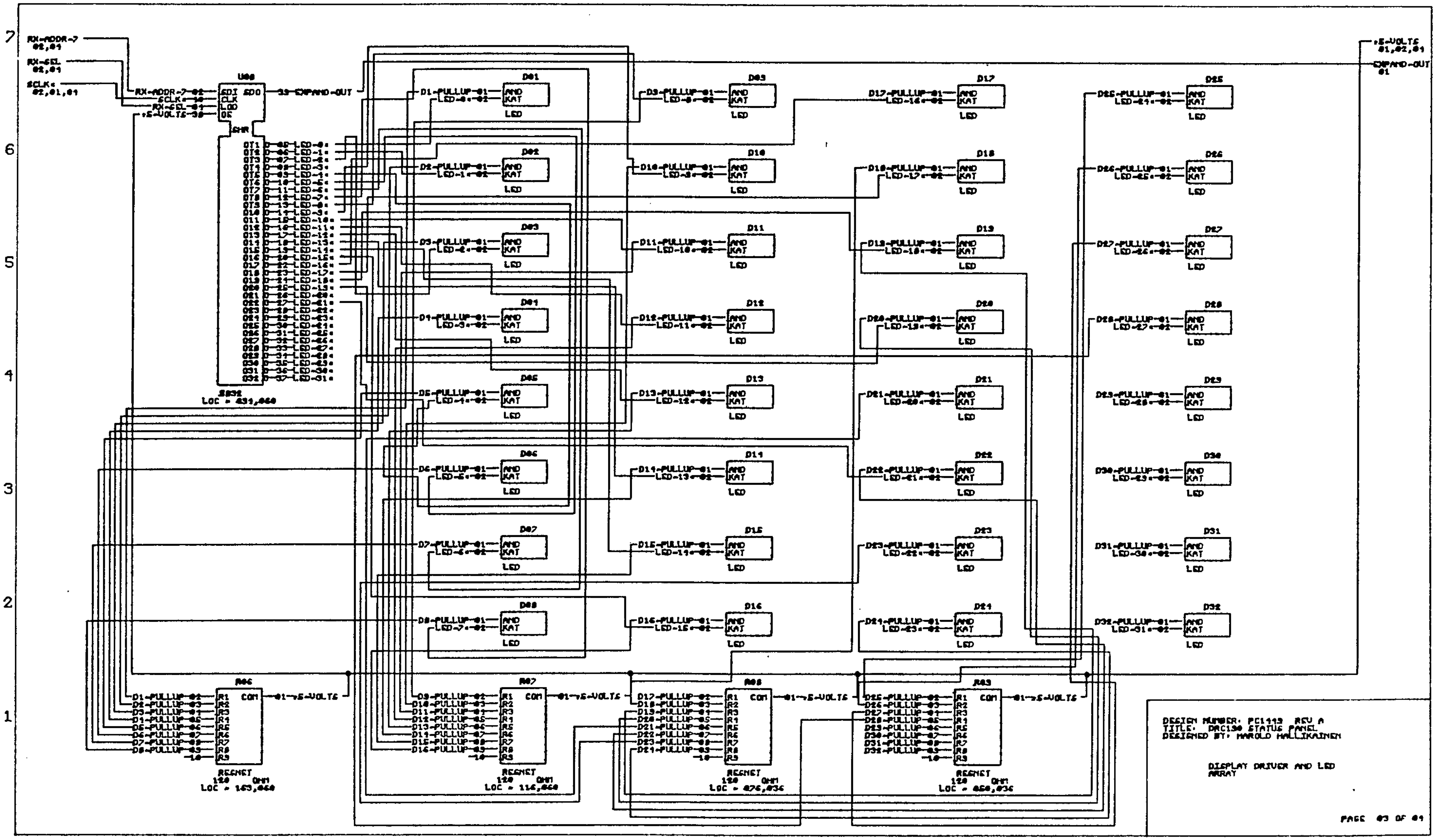


+5-VOLTS
DAT-FR-MDM
DAT-TO-MDM
DAT-TO-RS
HI-SPEED*
MDM-SW1
RESET
U01
MODEM
TELCO-TIP
TELCO-RING
P03
HEADER
D01
DIODE
1N751 1N
LOC = 023,006
R02
RES
150 OHM
LOC = 023,008
DAT-FR-RS
GND
U05
MC1488
P02
14HDR
RS232
LOC = 122,080
RS-232-OUT
RS-232-IN
RTS-CTS
U06
MC1489
DESIGN NUMBER: PC1445 REV
TITLE: DIRECT CONNECT MODEM
DESIGNED BY: HAROLD HALLIKAINEN
PAGE 02 OF 02

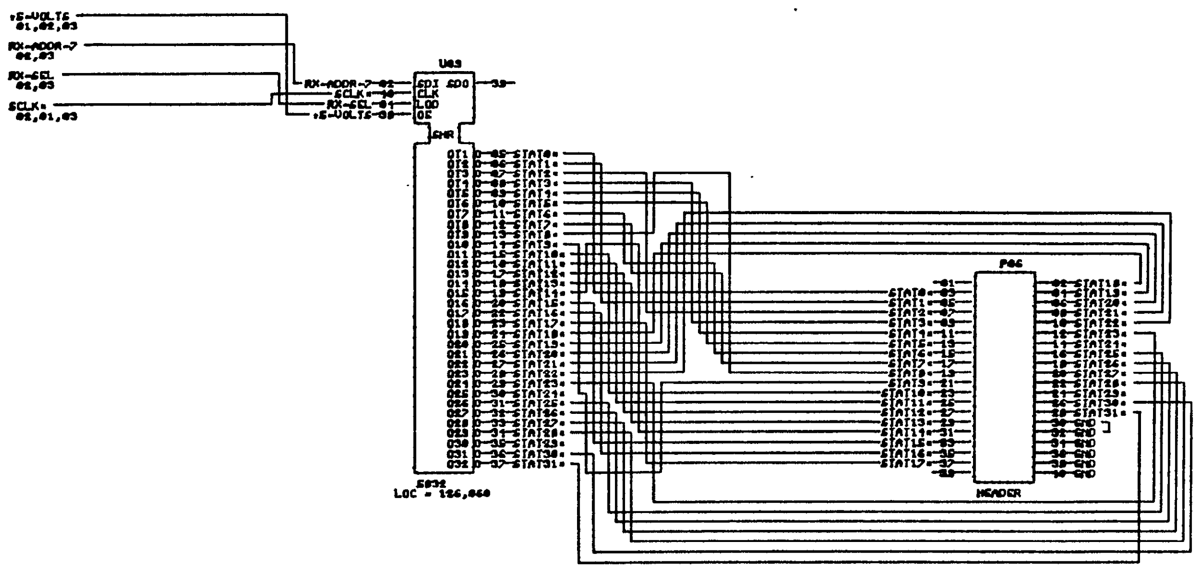

DESIGN NUMBER: PC1443 REV A
TITLE: DRC190 STATUS PANEL
DESIGNED BY: HAROLD HALLIKAINEN
STATUS INPUT SHIFT REGISTERS
PAGE 01 OF 01
TO PROCESSOR BOARD OR PREVIOUS EXPANSION BOARD
HEADER
RESNET
74LS165
74LS74



GND
LOAD-OUT*
SCLK*
STATUSOUT*
+5-VOLTS
RX-ADDR-7
RX-SEL
SCLK
INV
7404
HEADER
CMP
74LS688
SHR
74LS164
RESNET
DESIGN NUMBER: PC1443 REV A
TITLE: DRC130 STATUS PANEL
DESIGNED BY: HAROLD HALLIKAINEN
STATUS RECEIVER SITE COMPARATOR - TELLS US IF WE WANT TO DISPLAY THIS DATA



U08
SHR
S832
R06
R07
R08
R03
RESNET
DESIGN NUMBER: PC1449 REV A
TITLE: DRC130 STATUS PANEL
DESIGNED BY: HAROLD HALLIKAINEN
DISPLAY DRIVER AND LED
ARRAY
PAGE 03 OF 04

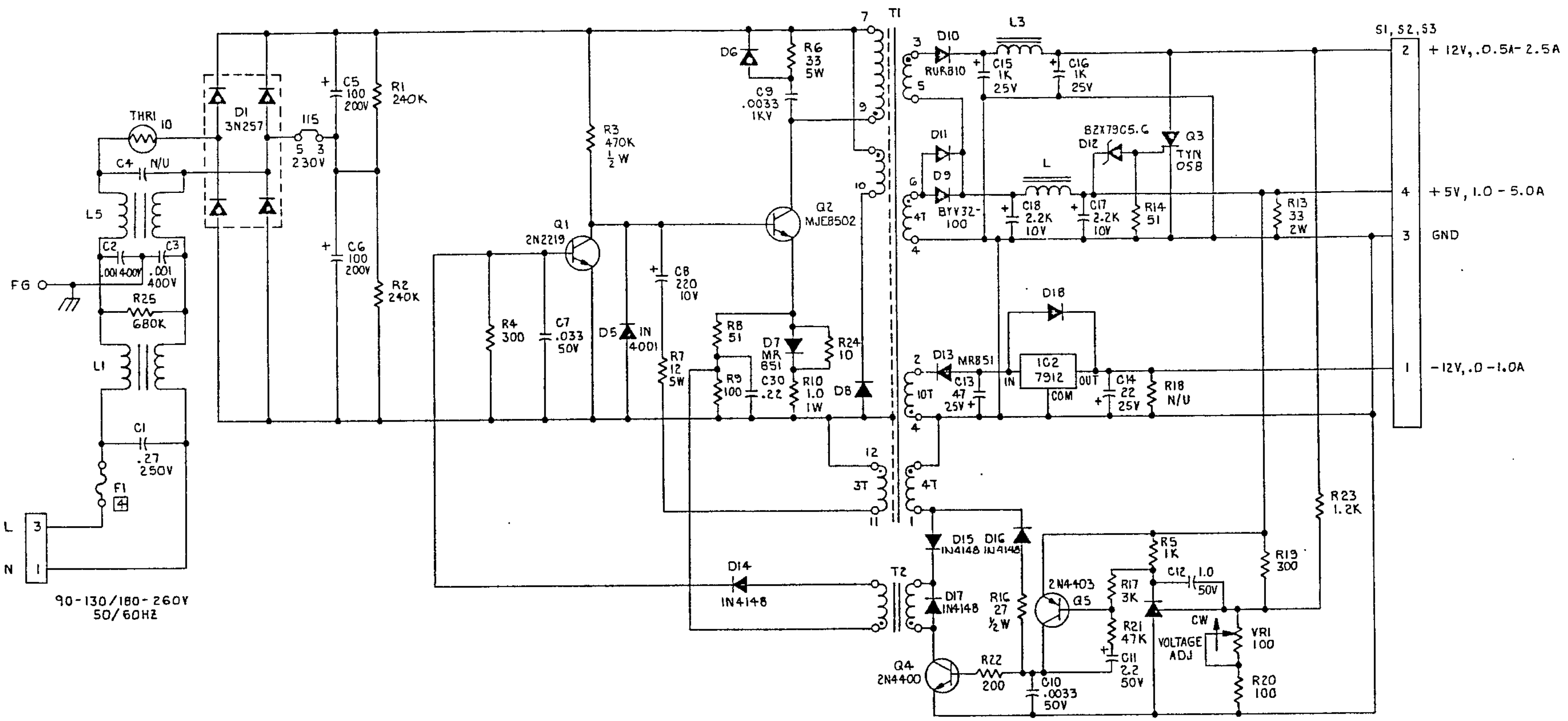


DESIGN NUMBER: PC1113 REV A
TITLE: DRC130 STATUS PANEL
DESIGNED BY: HAROLD HALLIKAINEN

USER PORT DRIVER

PAGE 01 OF 01

REV.	BY	DESCRIPTION	APPR	DATE
NC	EB	INITIAL RELEASE		



SEP 04 1987

NOTES: UNLESS OTHERWISE SPECIFIED

1. RESISTOR VALUES IN OHMS, $\frac{1}{4}$ W
2. CAPACITOR VALUES IN MICROFARADS.
3. DIODES IN4937
4. 1A / 110V
.5A / 220V

REF DESIGNATORS NOT USED	REF DESIGNATORS LAST USED
L2	R24 C18 D18 T2 Q5 F1 VR1 THR1

QTY REQ'D	DESCRIPTION	PART NO.	MATERIAL	MANUFACTURER	ITEM NO.

LIST OF MATERIAL

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES

TOLERANCES

FRAC. ±1/32 DEC. .XX ±.02 .XXX ±.010 ANGLES ±1/2°

SIGNATURES		DATES
DWN.		8-13-86
CHK.		
DSGN.		
ENGR.		
APPR.		

MATERIAL: SEE B/M

FINISH:

PROPRIETARY NOTICE

ALL PATENT OR OTHER PROPRIETARY RIGHTS PERTAINING TO THE ITEMS SOLD HEREUNDER SHALL REMAIN THE SOLE PROPERTY OF XENTEK INCORPORATED EXCEPT THAT THE PURCHASER SHALL HAVE THE RIGHT TO USE SUCH ITEMS FOR THEIR INTENDED PURPOSE.

XENTEK INCORPORATED
San Marcos, California

TITLE

SCHEMATIC
POWER SUPPLY
STANDARD XSW-60-512

CODE IDENT	SIZE	DWG. NO.	REV.
53279	D	8524-301	NC

DO NOT SCALE DWG. | SCALE: 1:1 | WEIGHT: ~ | SHEET 1 OF 1

8524-301 NC