OU/AEC/EER 53-5

TECHNICAL REPORT

COCKPIT WEATHER RADAR DISPLAY DEMONSTRATOR

AND

GROUND-TO-AIR SFERICS TELEMETRY SYSTEM

Described is a course-up cockpit weather radar display demonstrator that simulates aircraft translation and rotation of uplinked weather radar information based on aircraft speed and heading. Also described is a system to uplink to a display in an aircraft, sferic weather information, from a 3M-Ryan Stormscope. This includes a method of recording and playback of sferic data.

by

James D. Nickum, P.E.
Daryl L. McCall

Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

FINAL REPORT

December 1981 - December 1982

Prepared for

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia

Grant NAG-1-124

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D.)

LIST OF FIGURES

LIST OF FIGURES (CONT'D.)

I. INTRODUCTION

This report details the results of two methods of obtaining timely and accurate severe weather presentations in the cockpit. The first method described is a course-up display of uplinked weather radar data. This involves the construction of a demonstrator that will show the feasibility of producing a course-up display in the cockpit of the NASA simulator at Langley. A set of software algorithms have been designed that could easily be implemented, along with data tapes generated, to provide the cockpit simulation. The second method described in this report involves the uplinking of sferic data from a ground-based 3M-Ryan Stormscope. The technique involves transfer of the data on the CRT of the Stormscope to a remote CRT. This sferic uplink and display could also be included in an implementation on the NASA cockpit simulator, allowing evaluation of pilot responses based on real Stormscope data.

The combination of these two simulation capabilities should increase the ability of NASA Langley personnel to develop better man/machine interfaces for decreased workload and improved, accurate transfer of information to the pilot.

II. COCKPIT WEATHER RADAR DEMONSTRATOR DISPLAY

A. General Description

1. Initial Weather Radar Data Acquisition. The recordings of the weather radar for this demonstration were made at the Port Columbus Airport National Weather Service Facility which operates a WSR-74C weather radar that is equipped with an Enterprise Weather Radar Scan Converter. The recordings were made with a standard half-inch video tape recorder that was connected to the monitor used with the Enterprise system. All of the overlays except for the color intensity legend were turned off for the period of making the recordings. Approximately four hours of continuous weather activity was recorded on June 15, 1982 as a cold front passed through the Columbus, Ohio area.

On returning from Port Columbus, the weather radar data on video tape was re-recorded in a form that would allow better presentation to the cockpit weather radar display demonstrator (CWRDD). Figure 2.1 is a block diagram of the data acquistion and re-recording setup. The re-recording was necessary to facilitate the acquisition by the Matrox Frame Grabber of a complete updated radar display.

In the Enterprise system, a complete update of the radar display occurs every two minutes. During this time a complete radar display is available, but the display is updating radially through a complete 360-degree arc. The re-recording allows freezing a video frame when the update reaches the 360-degree position in the sweep and holding that radar display for the two-minute interval while the radar display is being updated. This then produces a video tape with clean radar display frames that will be input to the CWRDD during a simulated flight.

2. Matrox Video Display Boards. The central part of the display system is the Matrox Video display boards. This system consists of a display board and a frame grabber board which are designed LSI-11 compatible [1,2]. The display board will produce NTSC compatible video with a picture consisting of 256 x 256 pixels with a gray scale of 16 levels. This can be expanded easily to 256 levels by addition of a second display board and appropriate control strapping. The display boards are also capable of pseudo-color operation which allows colors to be displayed as a function of the gray scale value.

Video frames can be loaded into memory very easily with the Matrox frame grabber, which can digitize and store in the display memory the video information on the input during a single vertical frame interval. This is used to grab a frame of video from the video tape recorder for display on the CWRDD. A Rockwell AIM-65 microcomputer was used to perform the interface and computation necessary for the CWRDD unit. A custom interface was designed so that the LSI-11 interface of the Matrox boards would operate with the AIM-65. The Matrox boards are easy to use and provide access to the display memory on a pixel addressing basis which allows access to any pixel value either for modification or examination.

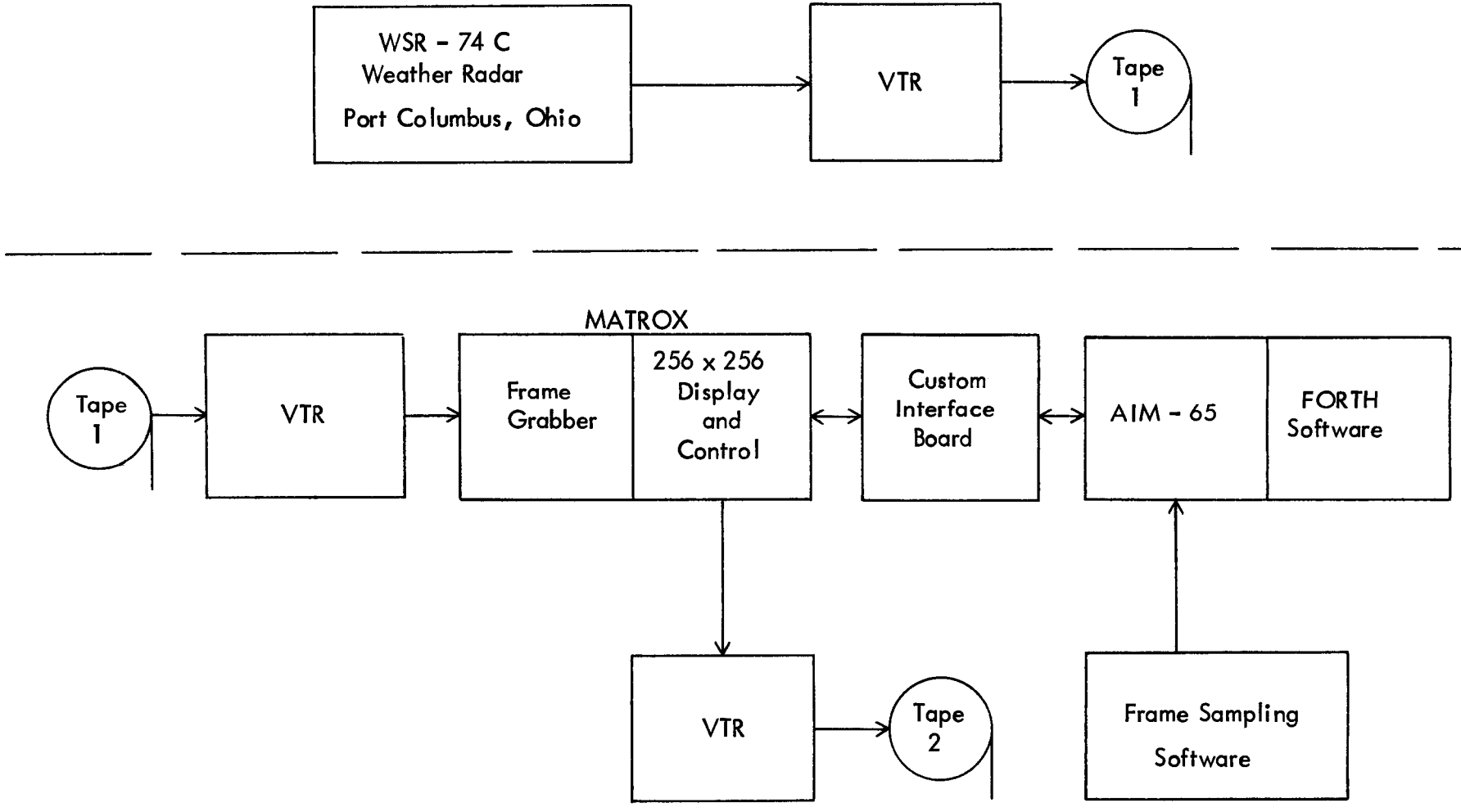


Figure 2-1. Weather Radar Data Acquisition Block Diagram.

3. Rockwell AIM-65 Microcomputer. The computer used to control the Matrox video boards is the AIM-65, which was chosen for its simplicity and access to necessary software and development tools [3]. The AIM-65 used is equipped with 4K of RAM and the standard 8K monitor ROM. In addition to this is an 8K FORTH software package and a 4K floating point software package that is used for trigonometric function availability. The FORTH software package allows the rapid development of software for the CWRDD allowing easy interface to assembly routines necessary for the interface between the AIM-65 and the Matrox boards [4]. Additionally, FORTH provides a faster run time environment than BASIC while still maintaining the higher level software development available in a structurable language. To implement sines and cosines, the AIM-65 floating point software package, which occupies 4K of ROM memory, was used [5]. The total software development design for this CWRDD task was approximately three months. Appendix A contains details of software development.

4. LSI-11 to AIM-65 Buss Interface. The hardware interface designed for the CWRDD is described in detail in Appendix B. Essentially, this interface will allow the transfer of data and addresses from the bidirectional 16-bit data and address buss of the LSI-11 to the 16-bit address and 8-bit data buss of the AIM-65. The design of this interface relies primarily on software to affect the exchange of data. Typical times for transfer of data are 135 μs using this interface. A complete hardware implementation could decrease the transfer time to about 15 to 20 μs. In order to demonstrate the CWRDD, it was decided that the slower software rate would be adequate since this system was never intended to interface directly with the cockpit simulator, but only as a way to demonstrate and develop the necessary concepts and algorithms.

Also included in Appendix B is the schematic for a joystick that is used by the CWRDD to input heading and speed information for the translation and rotation of the weather information in a course-up display mode.

B. Custom Interface Description

Figure 2.2 is a block diagram of the hardware used to produce the CWRDD. The AIM-65 computer executes a FORTH program that controls the frame grabber and display board to produce the translated and rotated course-up display on the monitor. The joystick is used to input speed and heading commands to the software for user interaction.

The custom interface provides two functions. First, it provides the necessary timing and hardware to transfer data across the Q-BUS to AIM-65 Busses. The Q-BUS is a combined 16-bit address and data buss with the AIM-65 using a 16-bit address and 8-bit data buss. Therefore, necessary timing and buffering is required to transfer data and addresses to the Matrox display and frame grabber boards. The second function of the custom interface is that the Matrox display board must have video sync to operate properly. The sync must be derived from the VTR while digitizing the video from the VTR, but using the VTR sync for display of the weather information on the monitor is not optimum as some jitter is present.

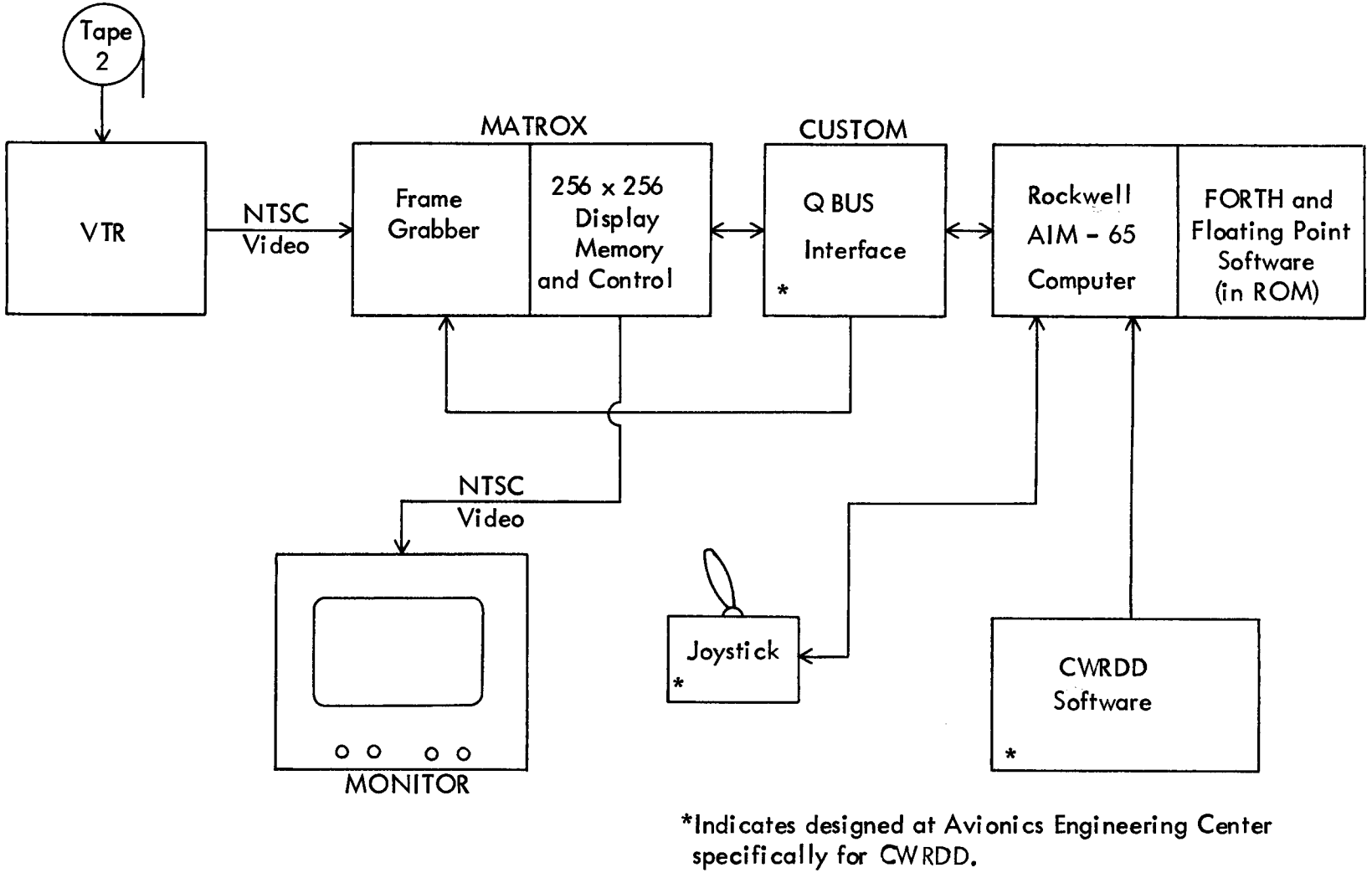


Figure 2.2. Cockpit Weather Radar Demonstrator Display Block Diagram.

It was decided that switching the sync from the VTR to an external sync generator would solve the problem. A simple sync generator is included on the custom interface card and the composite sync output is connected to one of the video inputs of the frame grabber. The VTR output is connected to another of the video inputs of the frame grabber. Under software control, when a frame is to be digitized from the VTR, the sync input is switched to the VTR. After the frame is digitized, the video input is switched back to the sync generator on the custom interface card. This provides stable sync for good monitor viewing.

Figures 2.3, 2.4, and 2.5 are photographs of the hardware used in the CWRDD described here. Because the CWRDD is a feasibility demonstrator, no special packaging is provided.

C. Software Definition of the CWRDD

The simplifications that are described here were made only to make the completion of the task progress more quickly. These simplifications do not affect the ability to produce a working CWRDD for the NASA simulator because they affect only the implementation on the AIM-65 and the interface to the Matrox boards. The basic translation and rotation algorithms remain the same and, therefore, can be transported through FORTRAN implementation on the NASA simulator.

Figure 2.6 indicates the format of the display of the CWRDD. Note that the input weather radar display and the rotated cockpit weather display are shown on the same frame. This was necessary because only one display board was available for the development of the project.

The maximum distance scale for the WSR-74C weather radar video frames provides a 230 km range. This provides, on the Matrox video display boards, a 0.97 x 0.97 nautical mile, square pixel scale for the display. The part of the display that shows the course-up weather was chosen to be 52 x 52 pixels which translates into a displayed area of approximately 50 x 50 nm. The aircraft was chosen to be placed along the vertical bisector of the course-up display and 5 pixels, or approximately 5 nm from the rear of the course-up display.

This display format was chosen as a compromise of several parameters which include the number of pixels in each course-up display and the ability to make the course-up display similar to the type of display currently used in airborne weather radars. One advantage is that there is a certain amount of rear vision with this scheme. This could be extended further by placing the aircraft at the center of the course-up display area, thereby allowing weather information display in all directions around the aircraft.

The course-up display is placed in the lower right portion of the actual north-up weather display. This allows viewing the course-up display on the same monitor as the north-up weather display, for easier software development interaction.

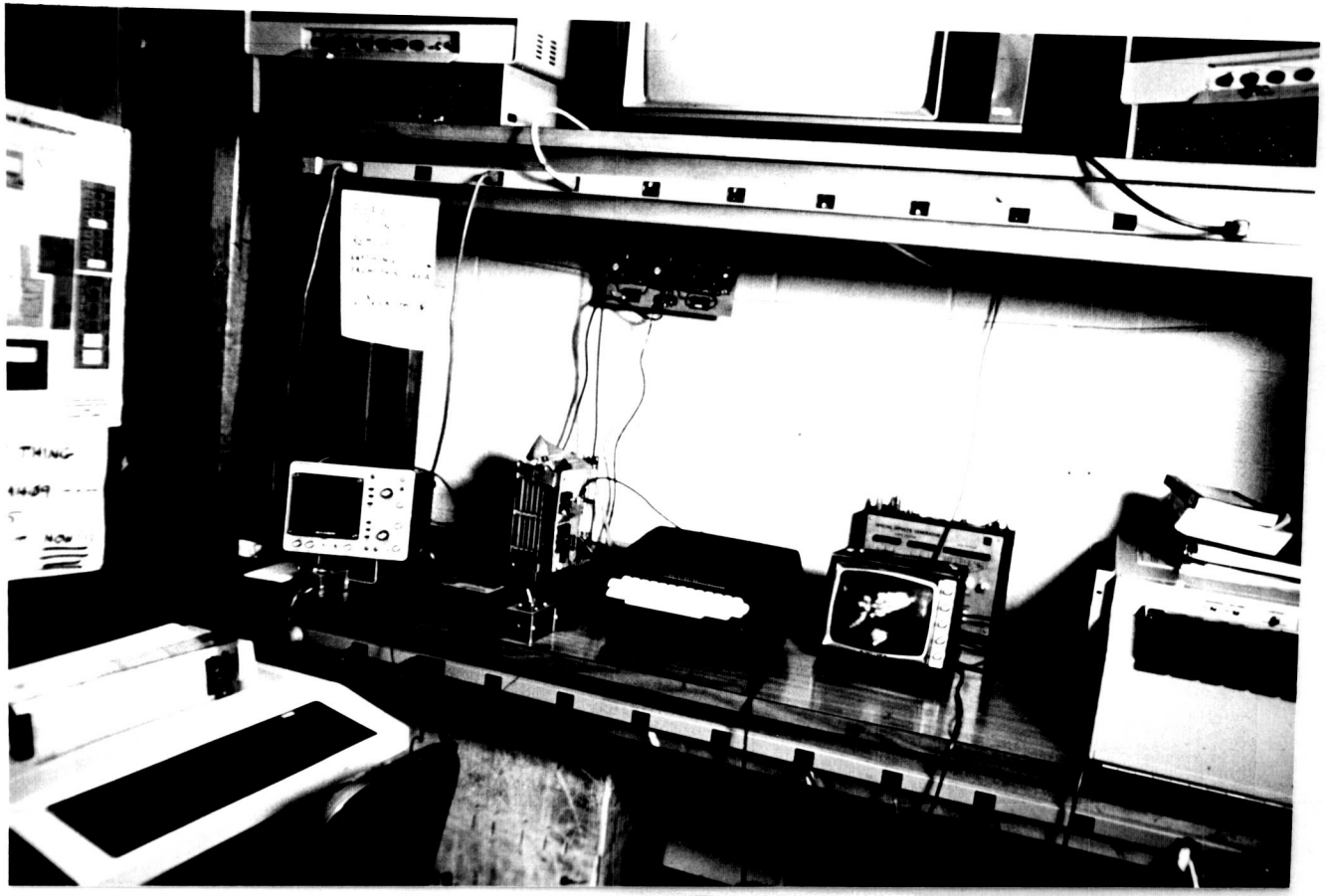


Figure 2-3. Software Bench Test Set-Up for CWRDD.

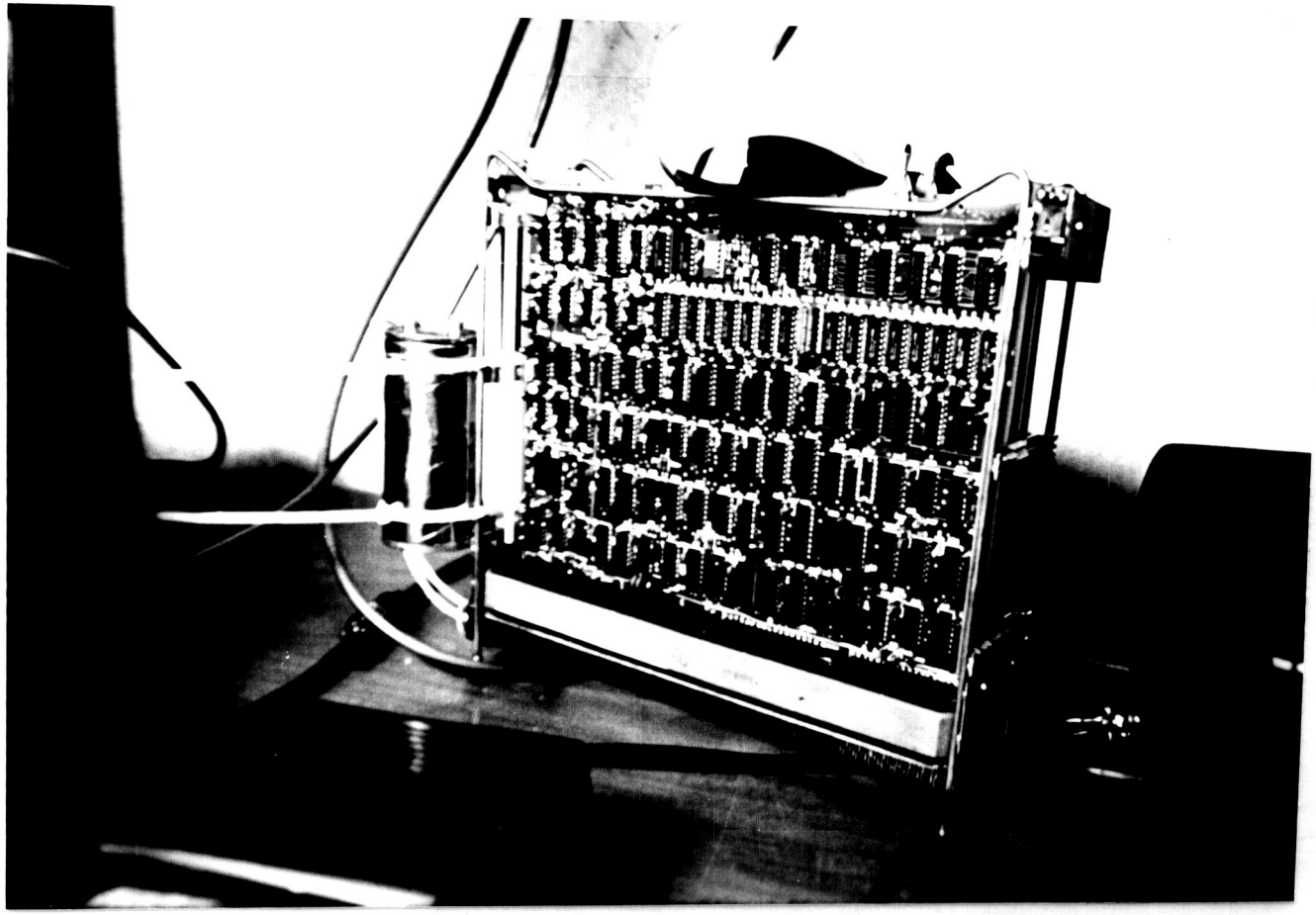


Figure 2-4. Matrox Display Board and Q-BUS Card Cage.

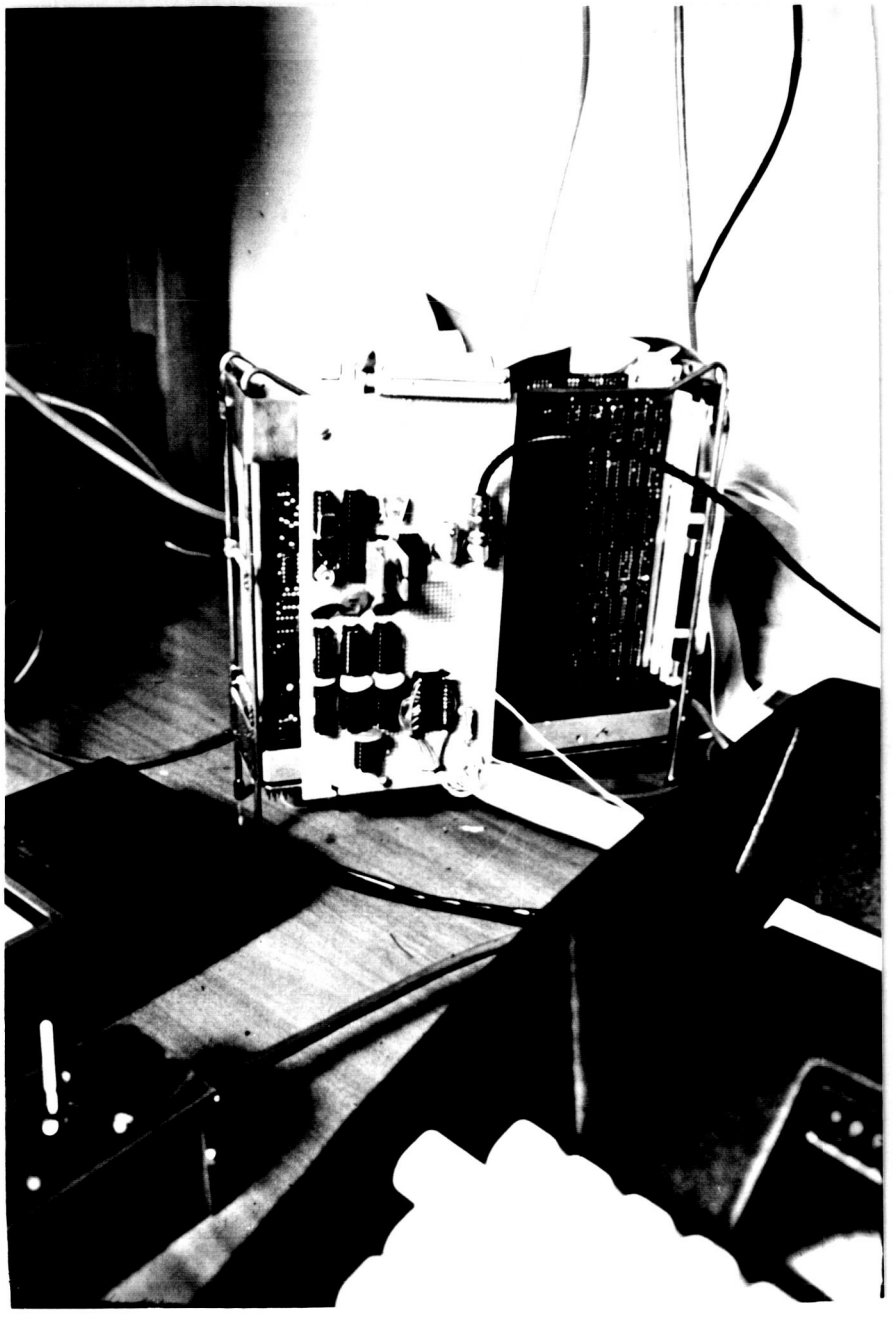
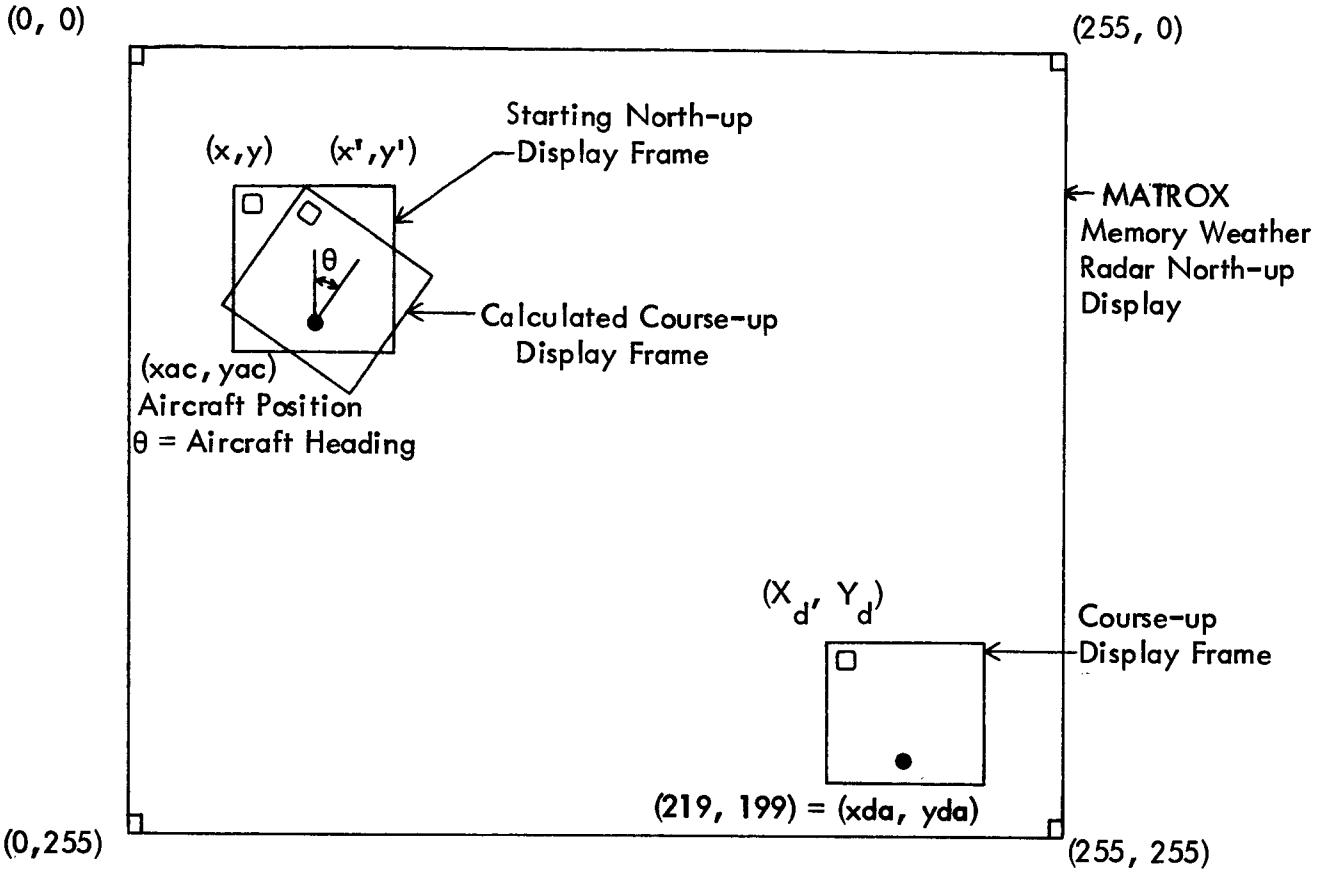


Figure 2-5. Rear View of Card Cage and Custom Interface Card.



$$\text{For each } y \begin{vmatrix} -5 \\ 47 \end{vmatrix}; \quad x \begin{vmatrix} 26 \\ -26 \end{vmatrix}$$

$$f(X_d,Y_d) = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x\cos(h) - y\sin(h) \\ x\sin(h) + y\cos(h) \end{bmatrix} + \begin{bmatrix} xac \\ yac \end{bmatrix}$$

$$f(X_d,Y_d) = \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} xda \\ yda \end{bmatrix}$$

$$\text{where } \begin{bmatrix} xda \\ yda \end{bmatrix} = \begin{bmatrix} 219 \\ 199 \end{bmatrix}$$

$$h = 360° - \theta$$

Figure 2-6. CWRDD Format and Calculations.

The actual calculations to perform the translation and rotation of the course-up display use simple equations to translate and rotate, in two dimensions, the cartesian coordinate system [6]. The two equations are the following:

$$x' = x \cos(h) - y \sin(h) + dx$$

$$y' = x \sin(h) + y \cos(h) + dy$$

Where (x,y) = coordinates of pixel in north up display field (52 x 52 pixel area)

(x',y') = coordinates of pixel to be deposited in course-up display

h = aircraft heading

(dx,dy) = coordinates of aircraft position in total weather display frame

For each y, which varies from 47 to -5, x varies from -26 to 26 relative to the aircraft position. This produces a top-to-bottom, left-to-right update of the course-up display frame.

The aircraft position is considered the origin of the coordinate system which, in our case, allows easy translation for "lifting" the 52 x 52 display space from the north-up display and placing it in the course-up display (CUD) space (addition of a constant to each pixel coordinate). The rotation has already been applied in selecting the correct pixel from the north-up weather display. In figure 2.6 the starting frame is located around the aircraft position in the north-up display (NUD) by assigning aircraft position as the origin of this frame then values of (x,y) are transformed into (x',y') values. The intensity value of (x',y') is read and then written into the same position as that of the NUD only translated to the CUD (Xd,Yd). In this manner the appropriate weather information is seen in course-up fashion. Since the sense of the trigonometric angles (counter clockwise) and the aircraft heading (clockwise) are opposite, the aircraft heading (h) in these calculations needs to be complemented. The difference of 90 degrees from the trigonometric definition of 0 degree and the aircraft heading of 0 degree is inconsequential since we are dealing with relative angles.

The complete equations for the displaying of course-up weather radar information used in this demonstrator are as follows:

The following calculations are made, for each value of y from 47 to -5, x is varied from -26 to 26.

$$f(x',y') = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \cos(h) - y \sin(h) \\ x \sin(h) + y \sin(h) \end{bmatrix} + \begin{bmatrix} xac \\ yac \end{bmatrix}$$

$$f(Xd,Yd) = \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} xda \\ yda \end{bmatrix} = \begin{bmatrix} Xd \\ Yd \end{bmatrix}$$

(xda, yda) = (219, 199) = coordinates of aircraft in north-up memory plane coordinates which locate origin of course-up display.

(xac, yac) = coordinates of aircraft position in weather display area.

(Xd, Yd) = coordinates in course-up display that correspond to (x,y) coordinates in the north-up display field.

h = 360 - (aircraft heading)

The north-up memory display coordinates (xda, yda) values of (219, 199) were chosen to place the course-up display frame in the bottom right side of the north-up display but to keep it far enough from the edge of the CRT screen edge to allow good viewing qualities. Changing these coordinates will move the area where the course-up display field is presented. This translation would not be necessary with the implementation on the NASA simulator because only the course-up display is required.

The above calculations are performed for each of the 2704 pixels in the 52 x 52 course-up display frame. These calculations are performed in approximately 20 seconds which provide the course-up display frame rate.

In the cockpit weather demonstrator presented here, the speed of the aircraft is controlled by a value input by the position of a joystick. The heading of the aircraft is input in the same way. The apparent motion of the aircraft through the weather is performed by modifying the coordinates of the aircraft position in the NUD. A complete display frame (52 x 52) is rotated by the heading input. Pixel intensity values relating to the NUD are stored in their relative positions in the CUD, as explained in detail above. This then provides aircraft (AC) motion in course-up sense around the north-up weather display.

For this demonstrator the speed is allowed to vary from 3 to 30 nm/min. which represent 180 to 1800 nm/hr. Due to the approximately 20-second per frame update rate, a change of 1 pixel per 20 seconds provides 180 nm/hr and 10 pixels per 20 seconds provides 1800 nm/hr. The faster AC speeds allow easier indications of AC movement around the weather. The weather movement with respect to the geographic positions are provided by the weather radar video update digitized from the recorded weather radar data which updates every 2 minutes.

Figure 2.7 is a flow diagram of the FORTH software to perform the CWRDD function. The detailed software is described in Appendix A including a complete FORTH listing.

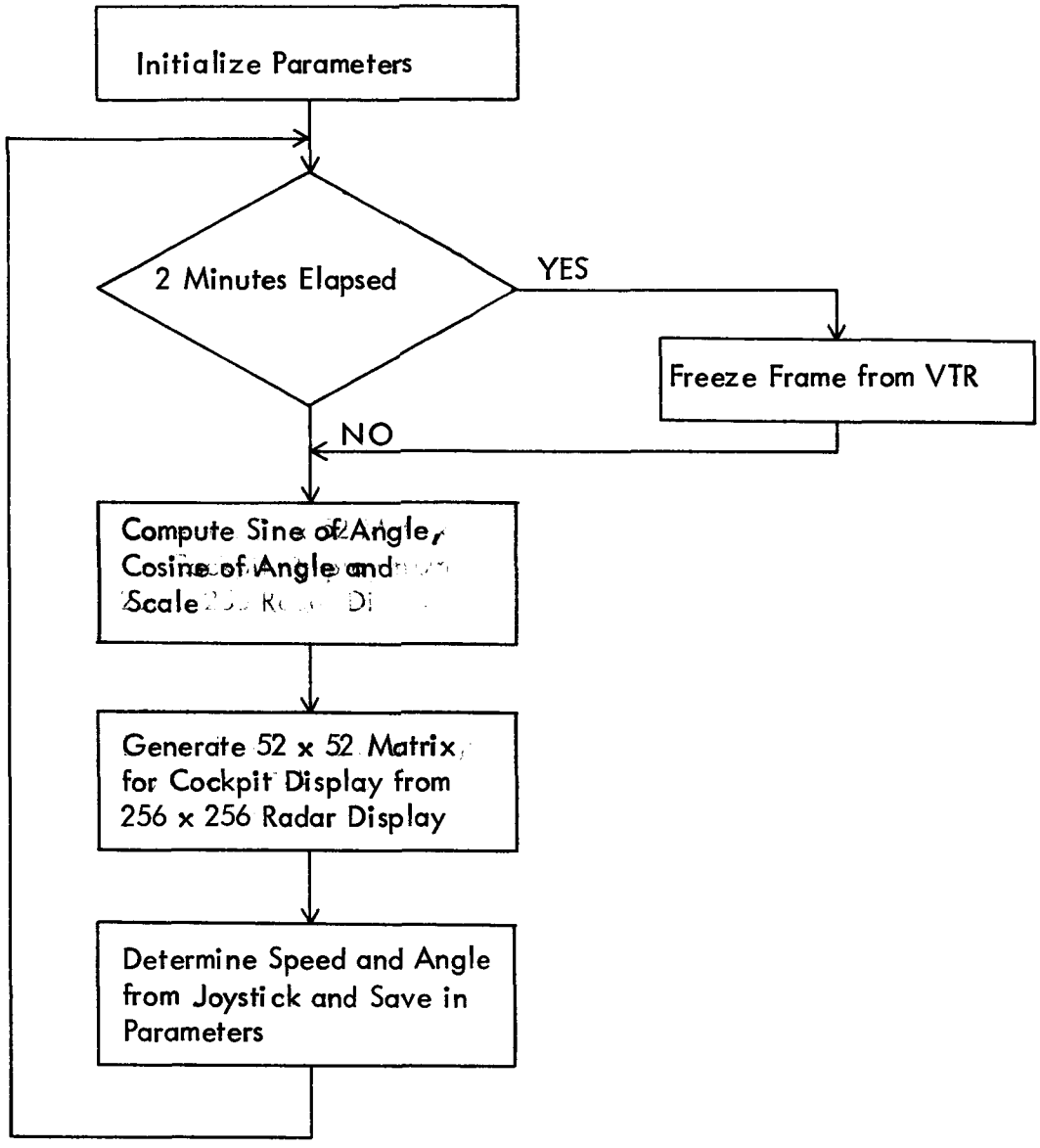


Figure 2-7. Simple Block Diagram of Software for CWRDD.

D. Cockpit Weather Radar Demonstrator Results

The cockpit weather radar demonstrator, using the calculations described above, does indicate a course-up display of aircraft motion with respect to the weather radar display. Using the joystick, the weather radar information that is in the course-up display frame translates and rotates based on the commanded heading and speed. The course-up display frame contents update at approximately a 20-second rate. This is slow, but the ability to demonstrate the algorithm is still valid. Faster and/or slower update rates will be achieved by implementation on the NASA simulator.

Figures 2.8 to 2.12 indicate a sequence of aircraft translation and rotation relative to the weather radar data display. Figure 2.8 indicates the aircraft on a heading of 360 degrees at approximately the center of the weather radar display. The next frame, figure 2.9, shows the aircraft on a 045-degree heading with the aircraft position still in the center of the weather radar display. The third frame, figure 2.10, shows the aircraft after translation along a heading of 040 degrees. Figure 2.11 now indicates the aircraft and course-up display after turning to a heading of 329 degrees. Finally, figure 2.12 shows the aircraft after translation along this 329-degree heading.

As demonstrated in these photographs of the CRT display the cockpit weather radar demonstrator is capable of producing course-up radar displays of simulated uplinked radar data. The calculations necessary to perform this course-up display are very simple and only the sheer volume of calculations and small computer implementation have resulted in slow frame update rate.

E. Recommendations

Based on the simplicity of the display algorithms and available speed of the NASA simulator computer, implementation of this concept in the NASA cockpit simulator is possible. Additionally, using a 16-bit microcomputer, hardware multiply, and look-up-table sines and cosines could produce an order of magnitude or better speed improvements of a system that could be used in operational aircraft.

Certain trade-offs of update rate vs. display field size can provide an acceptable display of course-up weather information in the cockpit. Certain functions such as cruise or maneuver modes could be added. Maneuver mode could be used when course changes of more than ±5 degrees are to take place. This would switch the course-up display field size to make it smaller and increase update rate at the expense of maximum look-ahead. In the cruise mode, this would allow a greater course-up display frame size for greater look-ahead capability. This would be acceptable because the translation at general aviation speeds is relatively slow.

For example, if the aircraft is moving with a speed of 180 nm/hr and no heading changes or small heading changes are made, the course-up display

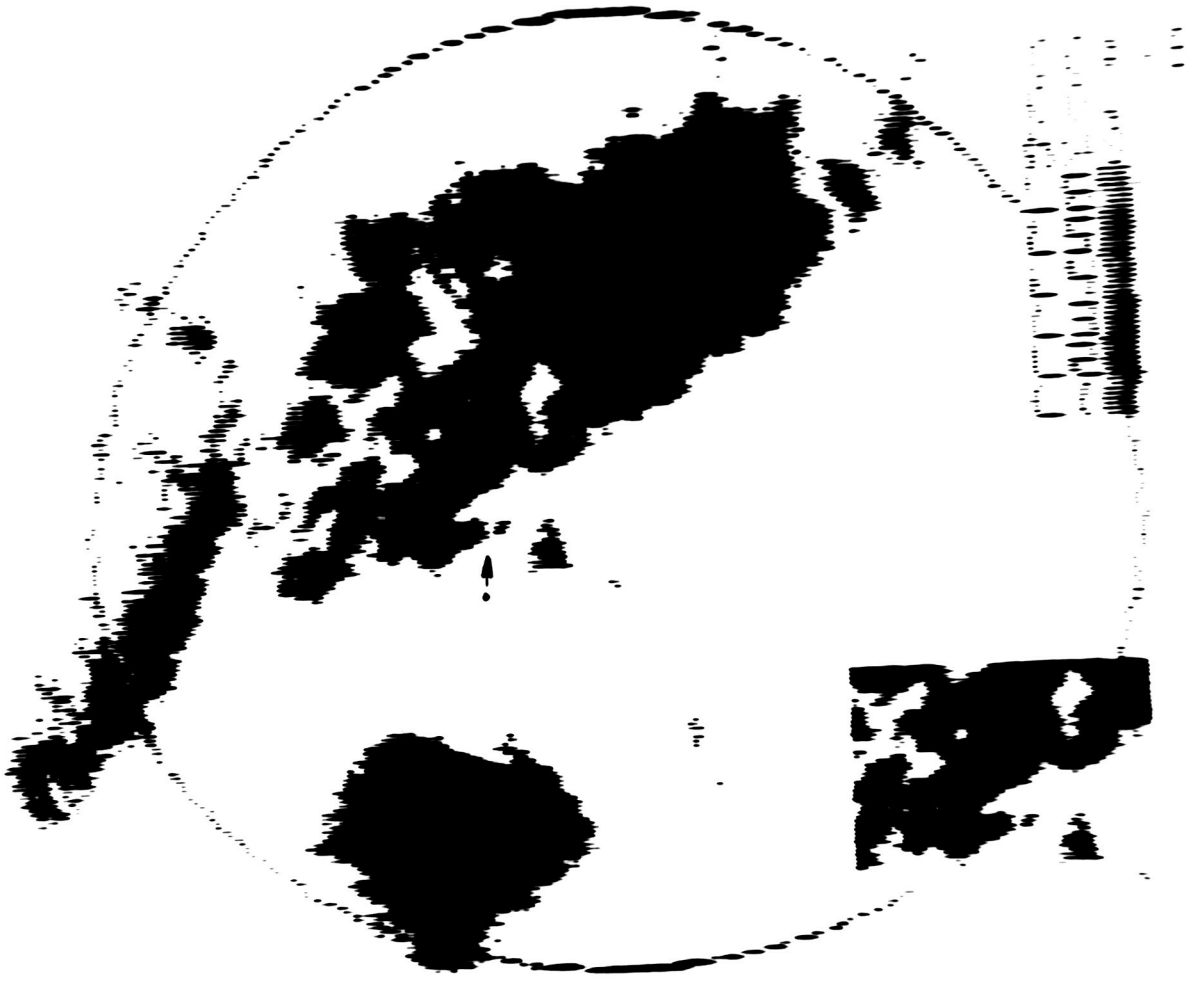


Figure 2-8. Video Display with Aircraft Heading 360° and Aircraft Position as Shown.

Figure 2-9. Video Display Aircraft Translated 7 Nm North with AC Heading 040°.

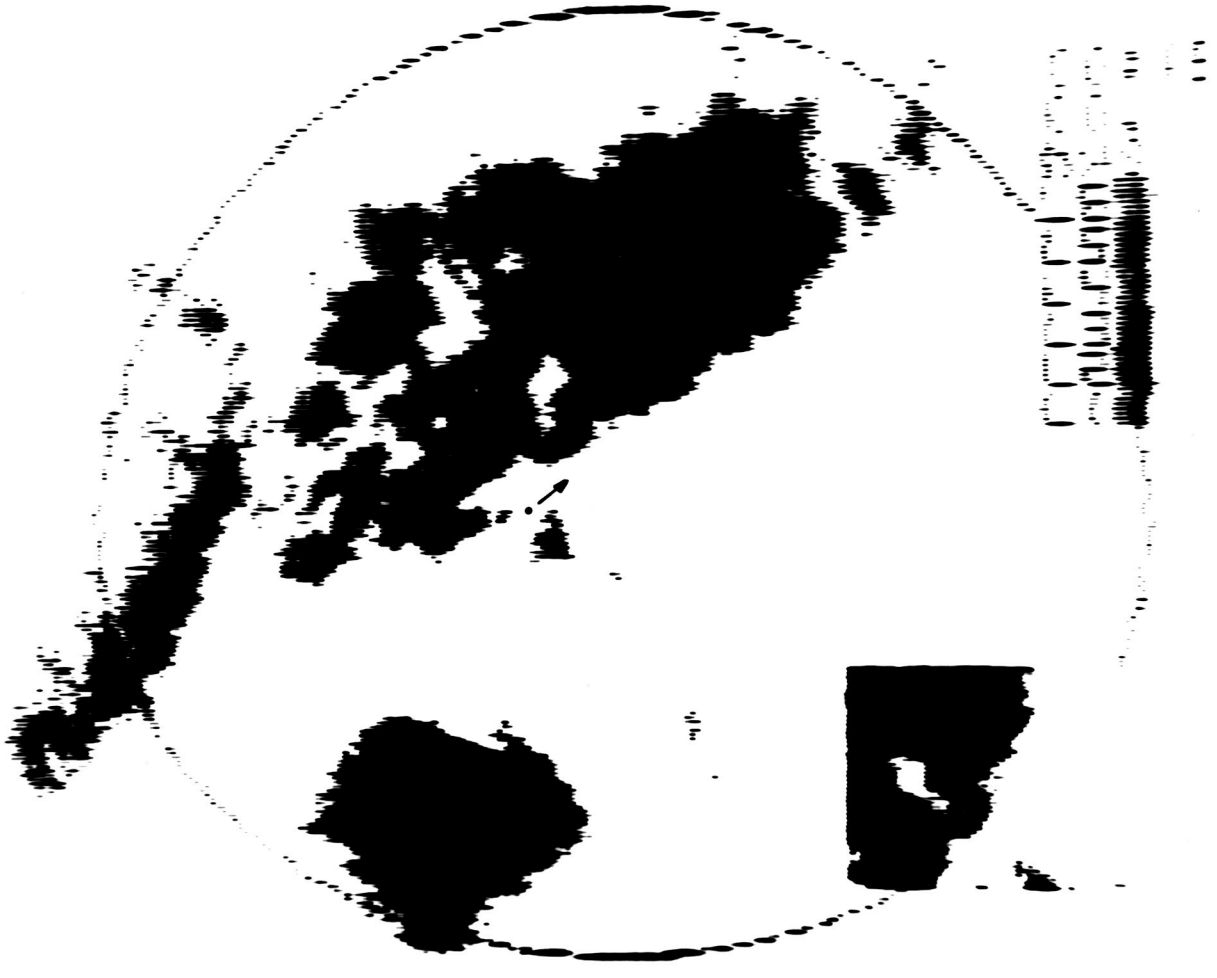


Figure 2-10. Video Display, Aircraft Translated Along 040° 7 Nm, Aircraft Heading 040°.

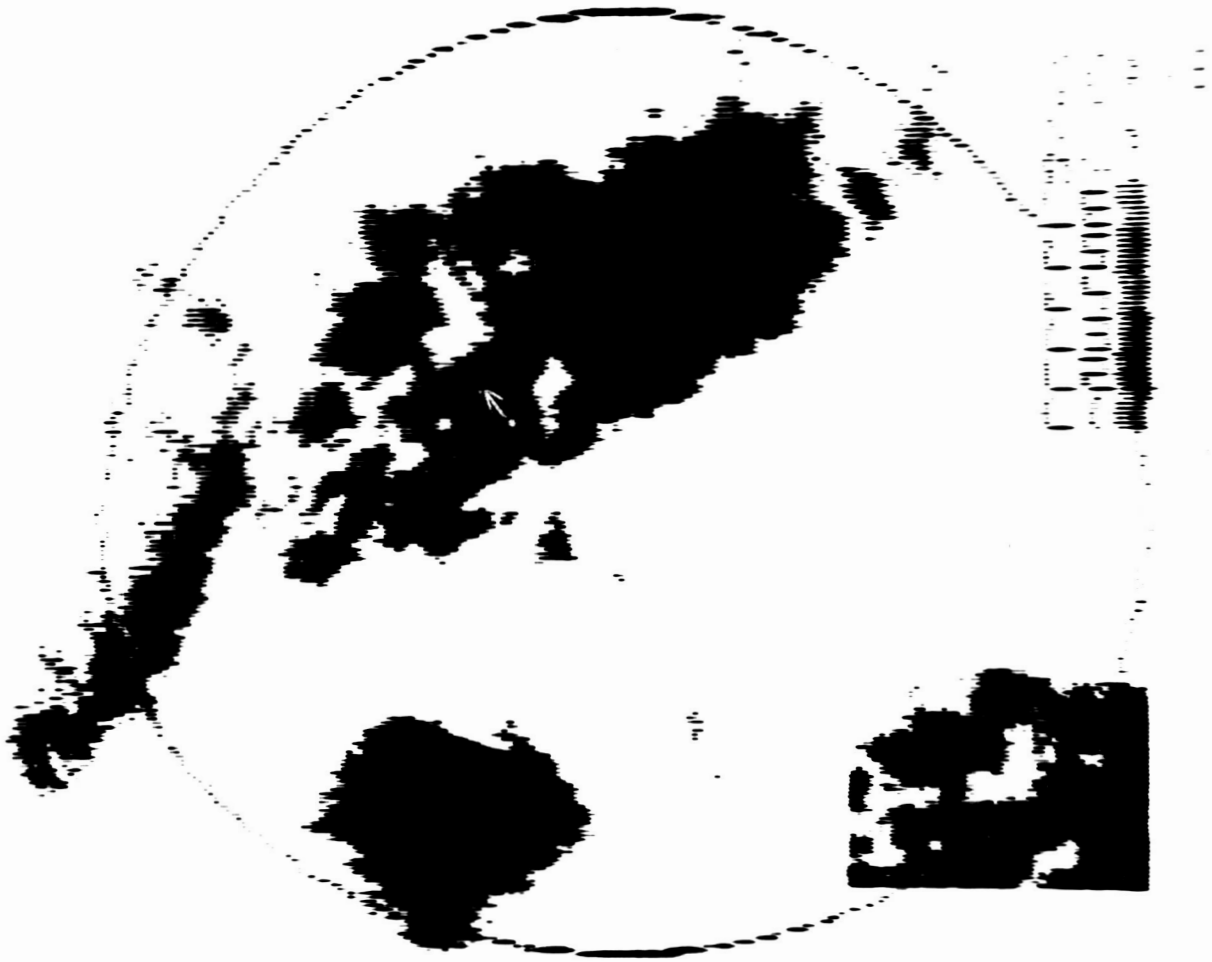


Figure 2-11. Video Display After AC Heading Change to 329°.

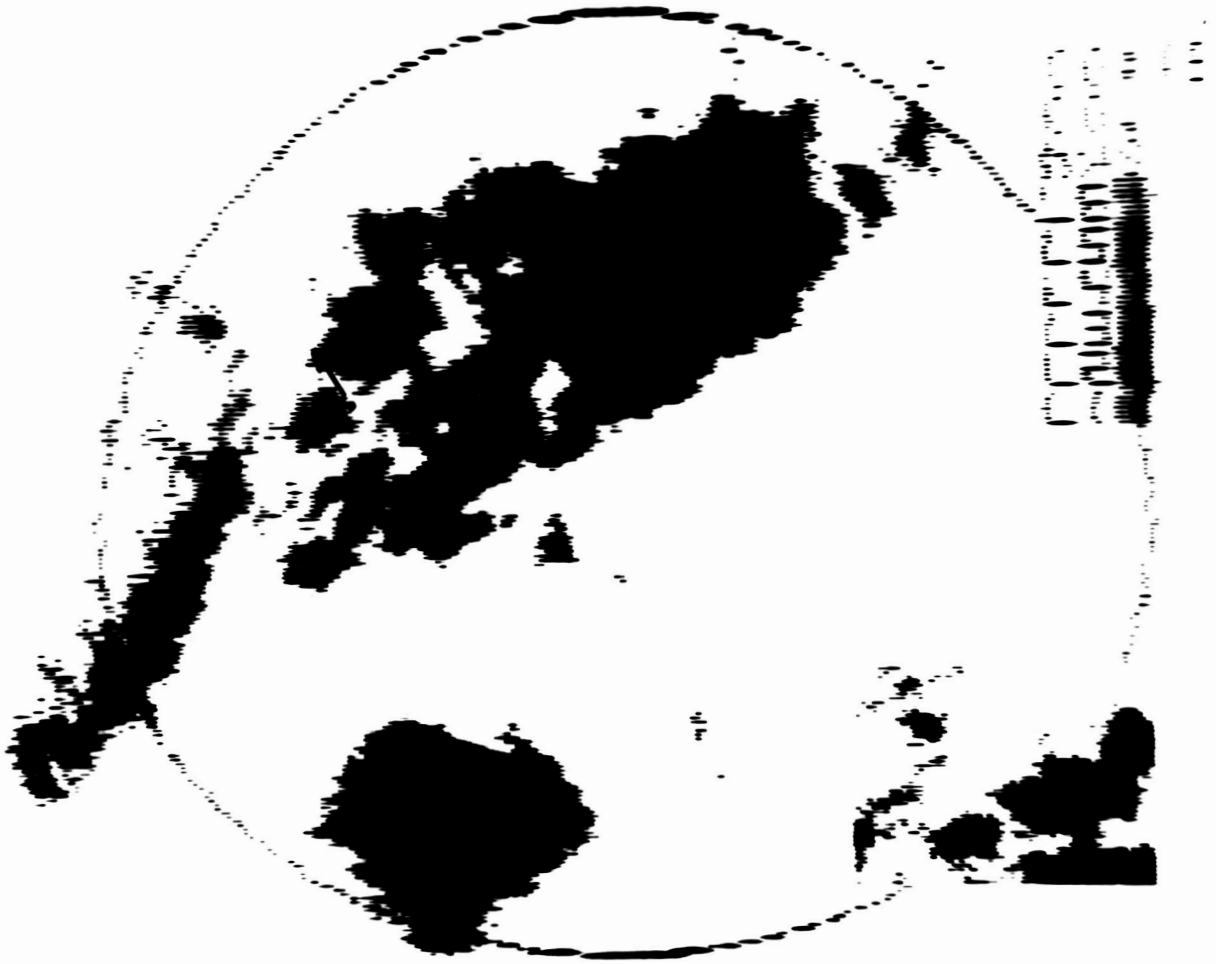


Figure 2-12. Video Display After Translation on 329° Heading.

will remain constant for at least 20 seconds. If it takes 20 seconds to translate or rotate the maximum course-up display size (in our case 256 x 256 nm), then this cruise mode will allow greater look-ahead capability with no real decrease in display capability.

III. GROUND-TO-AIR SFERICS TELEMETRY SYSTEM

A. Background

Until recently, the accepted means for detecting and analyzing weather activity was to use the reflectivity pattern obtained from a weather radar. Although today's technology has refined weather radar to obtain higher quality information, it has failed to increase its practicability in single engine aircraft and decrease the cost to the point where it is affordable to many commercial or general-aviation users.

An alternative to the active detection of weather is to use a passive sferics detector. This method has been gaining support, and several sferics detectors are available. The 3M-Ryan Stormscope Company produces a line of detectors that is geared for use with aircraft. Unlike weather radar, the Stormscope is affordable to more aircraft owners, and, since the sensing unit for the Stormscope has a very low profile, it can be installed in most aircraft with relative ease and very little effect upon the aircraft's aerodynamic performance.

The theory behind Stormscope is based upon the fact that statistically, most lightning is due to the electric potential generated within a thunderstorm, which, in turn, is due to the high convective currents produced with thunderstorms. Thus, if one avoids the areas of high electrical discharge, one will reduce the chances of being struck by lightning and, perhaps more important, avoid areas of high wind shear and resulting turbulence.

The 3M-Ryan Stormscope is capable of detecting and analyzing the electromagnetic properties of these discharges, and displaying the approximate range and bearing of such events. Since weather radar is capable of detecting only precipitation, it has been thought that a composite picture of the weather radar reflectivity pattern and the Stormscope could provide a more complete and detailed picture of the weather activity.

Due to the logistics of this task, it was deemed necessary that the composite picture would be derived at a fixed ground location and telemetered to the local air traffic. This paper documents the design and operation of a sferics-only, prototype uplink system, which could be incorporated into a composite radar/sferics uplink system.

B. Uplink System Overview

The uplink system can be broken into two major parts, the transmitter and receiver (see figure 3-1). The transmitter interfaces with the Stormscope, collecting data when made available by the Stormscope. This data has two forms, valid horizontal/vertical deflection voltages and memory erasures. The KIM microprocessor mimics the Stormscope memory configuration [7] and sequentially either erases or updates the current memory locations.

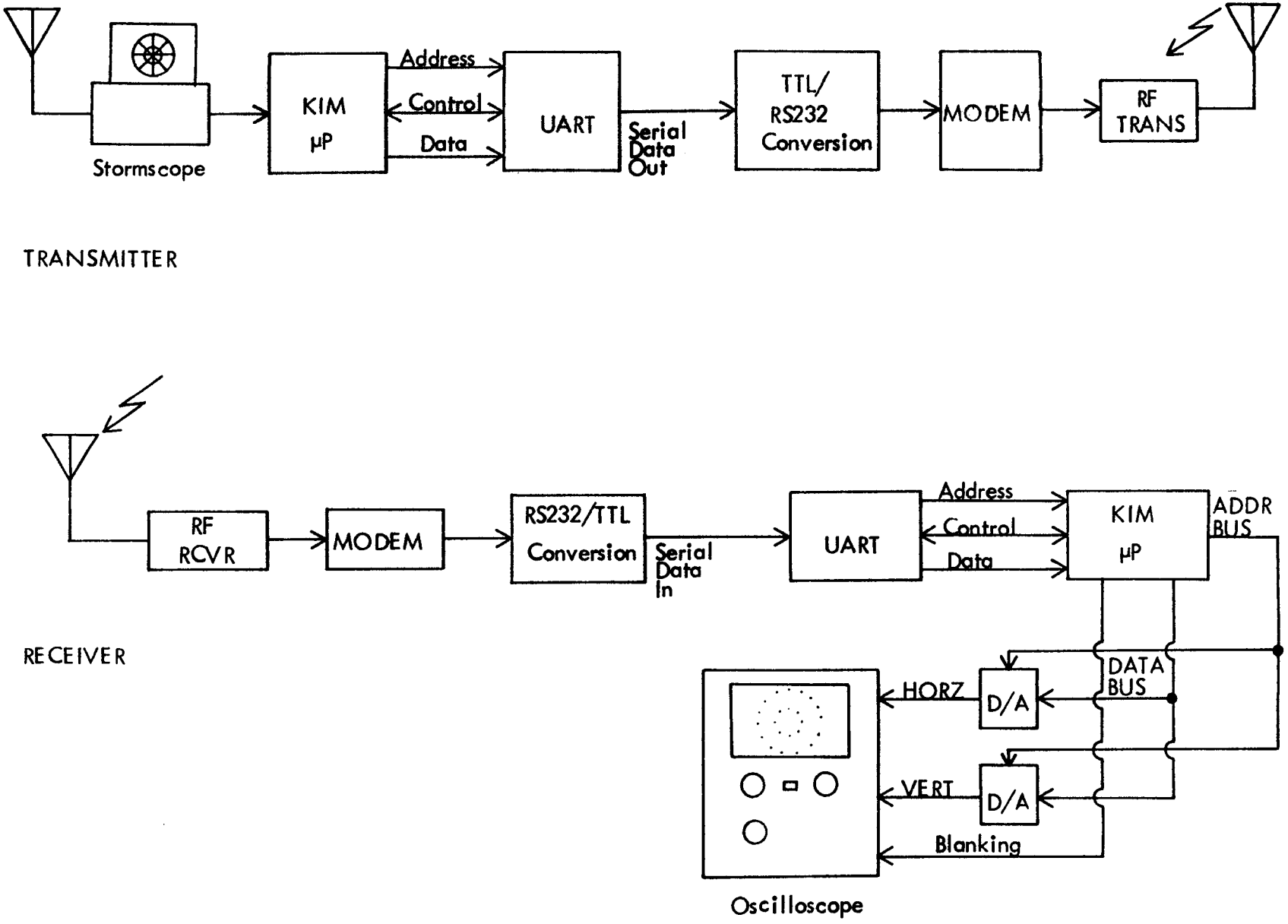


Figure 3-1. System Block Diagram.

As new pieces of data arrive (either updates or erasures), the microprocessor encodes and loads the parallel data into a universal asynchronous receiver/transmitter (UART) serializer. The UART then transmits the data serially to a device that converts the TTL levels to standard RS-232 levels. The RS-232 serial data is then fed to a modem which in turn drives the input to the RF transmitter.

The RF receiver of the uplink recovers the RF, detects the data and feeds it to a modem. The modem provides RS-232 levels for a RS-232/TTL level convertor, providing the serial TTL level data for the receiver's UART. The UART converts the serial data to parallel data for the microprocessor, and the microprocessor decodes the data and feeds it to the appropriate digital-to-analog convertor. These analog convertors provide the horizontal and vertical deflection voltages to drive an oscilloscope. The microprocessor duplicates the blanking signal, required so that the beam is off while being repositioned. The display effectively looks the same as the Stormscope, with the circular reference lines derived in video as opposed to the Stormscope screen overlay.

C. Transmitter Operations

The transmitting portion of the uplink receives the data from the Stormscope, encodes the data, converts it to a serial format (RS-232 compatible), and then transmits the data via a RF link. See Appendix C for a schematic diagram.

1. Hardware. A non-inverting, open collector buffer has been installed in the Stormscope, thus allowing access to the digital horizontal deflection data, digital vertical deflection data, the Master Clock (MCK), and the Write Gate (WG). The pull-up resistors for the buffer outputs are provided in the data acquisition circuit, attached to the inputs which the open collectors are driving.

The data acquisition circuit latches the digital deflection voltage data and produces a Non-Maskable Interrupt (NMI), upon receiving a high Write Gate and Master Clock from the Stormscope, on the outputs of the 74LS174. The outputs are read through the peripheral interface adapter (PIA) of the microprocessor during the NMI service routine.

During the background routine, the microprocessor encodes the received data by complementing the leading bit of a deflection voltage pair, the horizontal byte being sent first and the vertical byte last. Encoding is used so every horizontal data byte can be matched with its proper vertical component using simple correlation techniques in the receiver software. Unmatched data bytes will be totally ignored.

Once the data has been properly encoded, it may be sent to the UART to be converted to a serial format. The UART is accessed at any address in the 08X0H (H denotes a hexadecimal number) block (see figure 3-2); however, the software only recognizes the lowest of these addresses (i.e., 0800H). The data to be changed to serial format is stored at this address. The

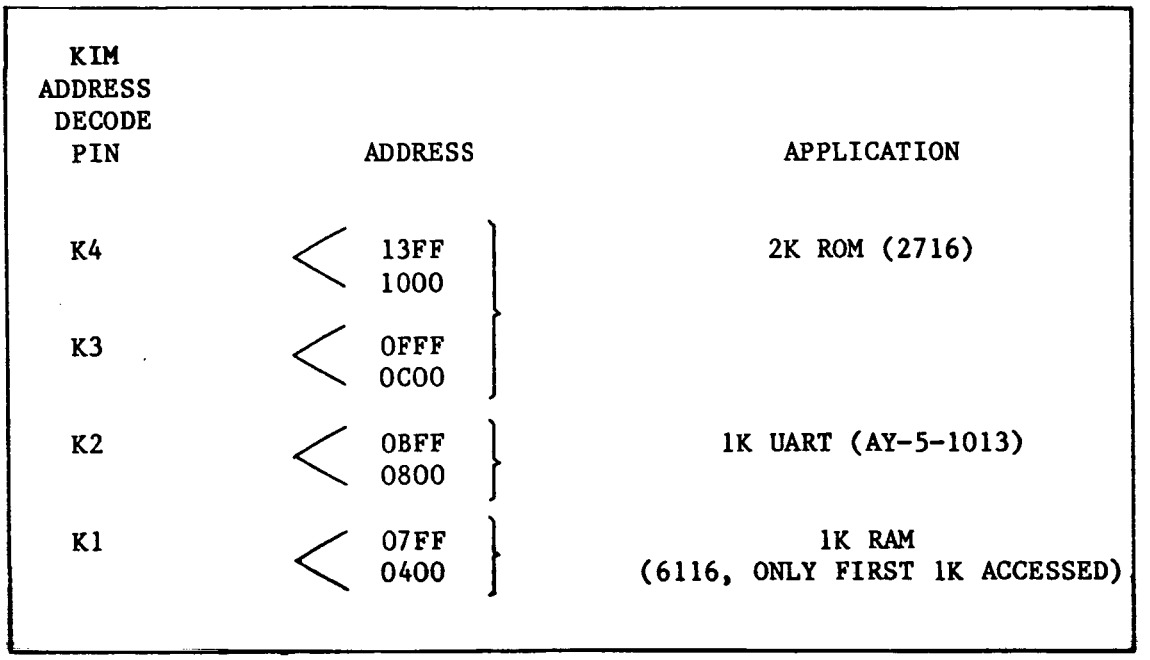


Figure 3-2. System Memory Map.

74LS374 is address-encoded so that the information on the data bus is latched to its outputs when the address 08X0H (hexadecimal) is placed on the address bus. This allows the data to be loaded into the AY-5-1013 UART. Once the data has been loaded, and the UART has acknowledged that the previous character has been sent (via PIA port PA6), the UART is strobed to send by writing to address 08X1H. This produces a pulse to strobe the UART (again, note that the software uses the lowest of these addresses, 0801H).

The AY-5-1013 is programmed to send eight data bits without parity, two stop bits, and can transmit at either 300 baud or 1200 baud. The only control lines used in this design are End of Character (EOC; a low on this line indicates the UART is waiting for a Data Strobe) and Data Strobe (DS; a low on this line instructs the UART to send data).

The serial data must now be converted from TTL levels to RS-232C levels. This is conveniently done by using a DS1488 line driver. TTL-level serial data is input and RS-232C compatible serial data is produced. This data is then fed to a standard computer modem. The modem converts the data from RS-232C digital pulses to frequency shift keying (FSK) format. For FSK, the frequencies used are in the audio range, and these are input to a VHF transmitter for relaying to the aircraft.

2. Software. The software used to control the described hardware was generated from the 6502 MPU assembler, available from the Ohio University IBM VM/370 system. The program is responsible for system initialization and all system operations. See Appendix F for complete software listings and figures 3-3 through 3-5 for all transmitter software flowcharts.

a. NMI Service Routine. Upon receiving an NMI, software control is removed from the background routine to the NMI service routine (see figure 3-3). The service routine will store the A and X registers and load X with the UART Data Index Pointer (UPNT). The UPNT, when used in conjunction with the indexed addressing mode, will mask and place the new data in a location just following the previous data and then be incremented by one. The NMI service routine then restores the A and X registers and exits the service routine.

The memory allocation for all of the received data is 256 bytes for the horizontal data and 256 bytes for the vertical data; therefore, the UPNT will be reset to zero automatically when incremented after it has attained the value of FFH. This also means that the service routine will begin to write over old data once 256 byte pairs of data have been received. It has been calculated and found in practice that the data flow from the Stormscope is slow enough for the background routine to process old data before the data destruction process begins.

b. Background Routine. The background routine has a counterpart to the UPNT of the NMI service routine; it is called the Data Available Index Pointer (DAVPNT). Upon receiving a reset or a spurious interrupt request (IRQ) (see figure 3-4), the background routine zeroes the DAVPNT,

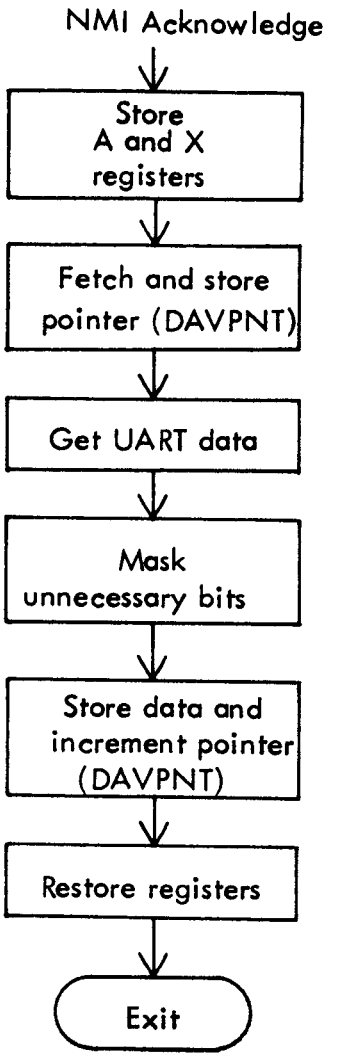


Figure 3-3. Transmitter NMI Service Routine.

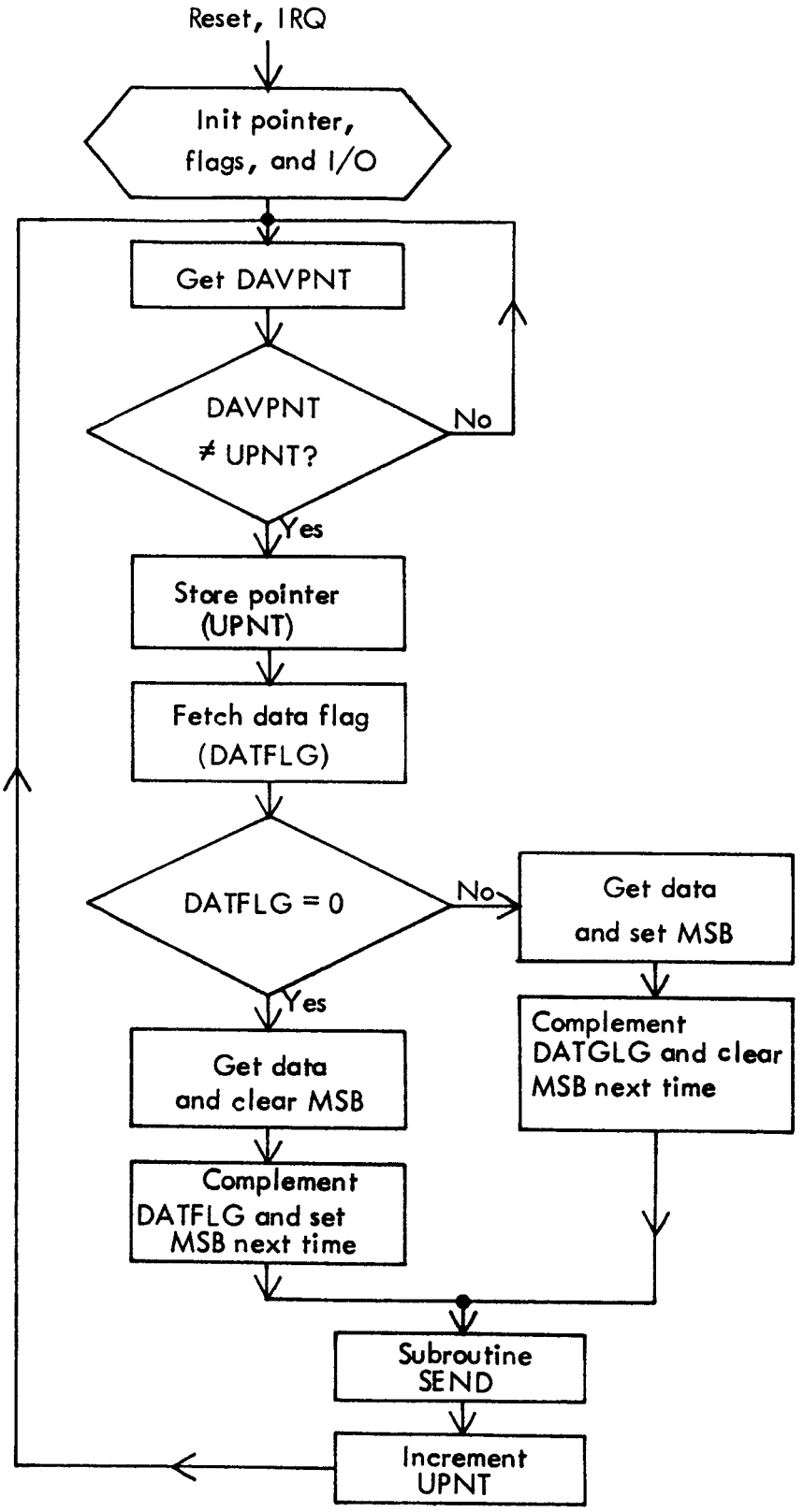


Figure 3-4. Transmitter Main Program Loop.

UPNT, Data Flag (DATFLG), and initializes all of the I/O ports. The routine then enters an interrogation loop, which constantly compares the DAVPNT to the UPNT. Obviously, when data is available the UPNT will be unequal to the DAVPNT.

Once it has been determined that the two pointers are unequal, the background routine stores UPNT in the X register and reads the DATFLG. The DATFLG is used to determine whether the matched data pairs should be encoded with ones or zeroes and turns program control over to the proper encoding routine. For example, if it is found that DATFLG is equal to zero, then the leading bit of the matched horizontal and vertical data bytes will each be zeroed by sending program control to the zero routine (this zeroes the leading bit of the matched horizontal and vertical data bytes). In this way, the receiving part of the uplink can match the horizontal with its proper vertical component. Upon leaving the encoding routine (either ZERO or ONE), the DATFLG is set so that the complement of the current leading bit is used during the next iteration.

Program control is now turned over to the SEND routine (see figure 3-5). This individually removes the encoded data pair from memory, loads it in the UART data buffer and then strobes the UART on the DS control line. The falling edge of DS loads the data into the UART's internal data registers and the rising edge initiates the serial transmission. The program then enters another interrogation loop that reads the EOC until it goes high indicating that the data transmission is complete and another character can be sent. It was decided to send the horizontal byte first, immediately followed by the vertical byte.

Once the vertical byte has been sent, program control is sent back to background routine, where UPNT is incremented and the process begins again. If DAVPNT is still unequal to UPNT, data processing continues. If DAVPNT is equal to UPNT, this indicates that the background routine has caught up with the NMI service routine and that no new data is available for transmission.

D. Receiver Operation

The receiving portion of the uplink receives the data, converts it to parallel form, decodes the data, feeds the D/A convertors with the proper data bytes, and provides a blanking signal to display the Stormscope image. The schematic in Appendix D illustrates.

1. Hardware. The data is first received by the RF receiver, which converts the modulated data into FSK audio data; this in turn is changed into digital RS-232C pulses by the computer modem. The RS-232C format pulses relayed by the modem are converted into TTL levels by the DS1489 line drivers.

The TTL level serial data is then fed into an AY-5-1013 UART, configured in the receiver mode. Once the complete serial string has entered the UART, the Data Available (DAV) goes high. This line is inverted and

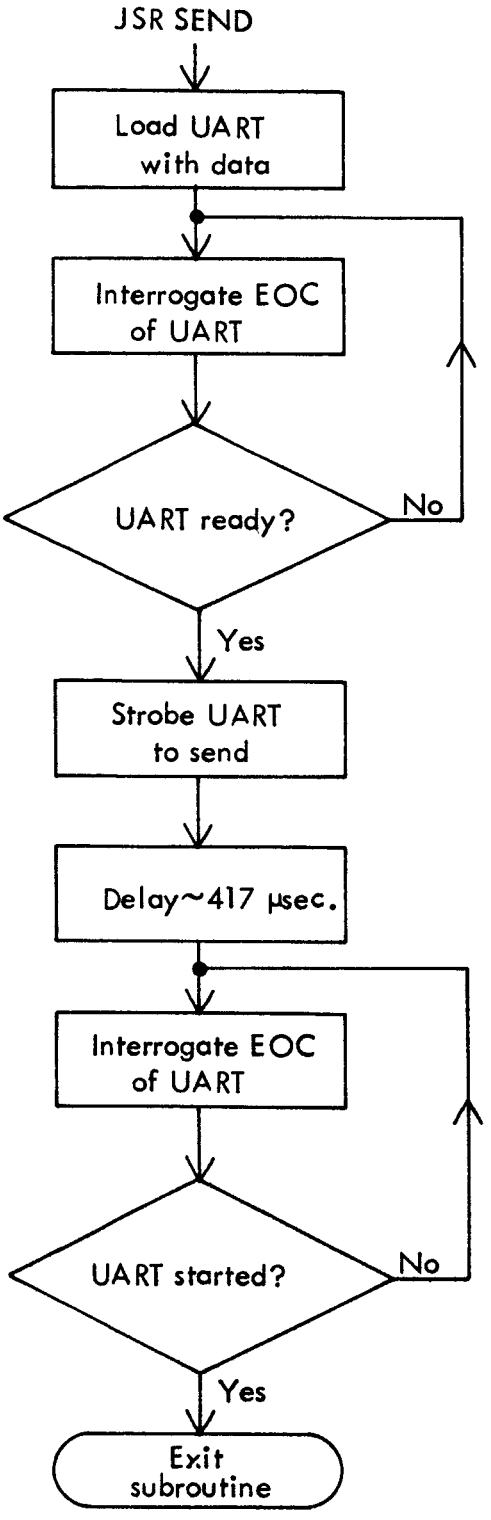


Figure 3-5. SEND Subroutine.

fed into the NMI of the microprocessor, signaling that a data byte is ready to be read into memory. The receiving UART is address encoded in much the same way as the transmitting UART. The UART is read at address 080XH; however, the software only recognizes address 0800H. The UART is read when a NMI is received. Addressing the UART in the read mode allows Received Data Available (RDE) to go low, thus allowing the parallel data buffers to be output on the microprocessor's data bus.

The NMI service routine reads the data from the UART and stores it in temporary memory. In the background routine, the available data is placed in one of two (horizontal and vertical) 128-slot recirculating memories. As in the recirculating shift registers of the Stormscope, after the first 128 pieces of data have been stored, every succeeding data will write over old data. The receiving portion of the uplink contains 128 locations of memory to duplicate the capabilites of the 3M/Ryan Stormscope Model WX-7A.

To display this data, the microprocessor will continually pick corresponding data bytes (horizontal and vertical) from these two memory vectors and write them to their respective digital-to-analog convertors (D/A). The D/A convertors used in this system are the AD558 DACPORTs, and they are memory-mapped to addresses 08X0H for the horizontal deflection voltage and 08X1H for the vertical deflection voltages. As before, the receiver software only recognizes 0800H and 0801H for horizontal and vertical, respectively. Once the D/A convertors have been programmed, the microprocessor provides a blanking/unblanking signal to the Z axis of the oscilloscope to turn the beam of the CRT display on, once the beam has been properly positioned. This signal is provided through PIA port A, output number PA0.

2. Software. The receiver software is responsible for reading the UART for new parallel data upon receiving an NMI, continually displaying the data in its recirculating horizontal and vertical memories, and provide a blanking/unblanking pulse to the Z axis of the CRT display. See figures 3-7 through 3-11 for software flow charts and Appendix D for software listings.

a. NMI Service Routine. On the occurrence of an NMI, control is transferred to the NMI servicing software (see figure 3-7). The service routine first stores the A and X registers. Next the data byte is read from the UART and stored at address TEMP. The microprocessor then fetches the status register (STATUS; figure 3-6) and sets the leading bit high, signifying that a new byte has been written into TEMP. Now RDAV is set low, then high by storing a zero and then a one in bit PA0 of PIA port B. The UART is now ready to signal the microprocessor with an NMI as soon as a new byte has arrived. The microprocessor then restores the A and X registers and returns to the main program.

b. MAIN Program. When the microprocessor recognizes a RESET or an IRQ, it begins to initialize the range ring display routine (see figure 3-8). Since the microprocessor displays video data by retrieving beam

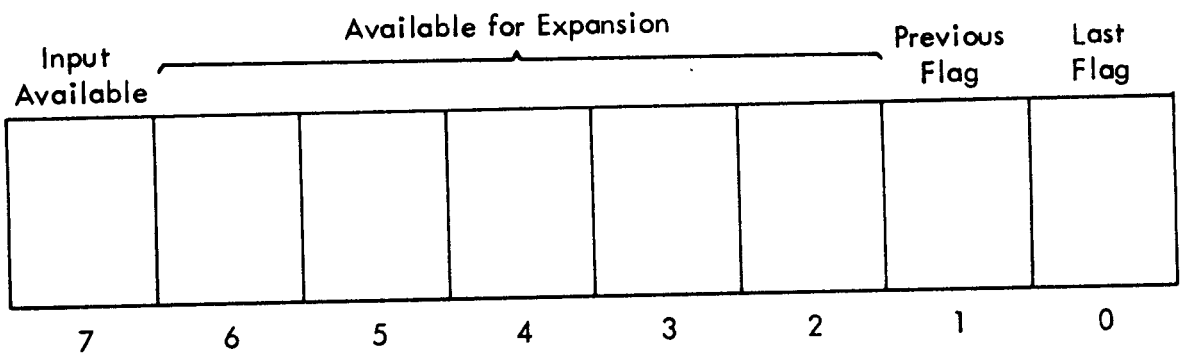


Figure 3-6. STATUS Register.

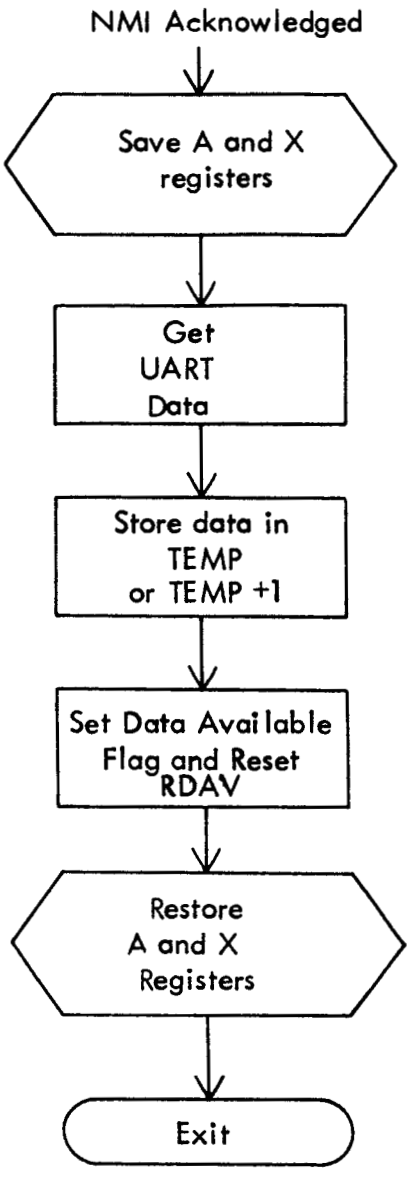


Figure 3-7. Receiver NMI Service Routine.

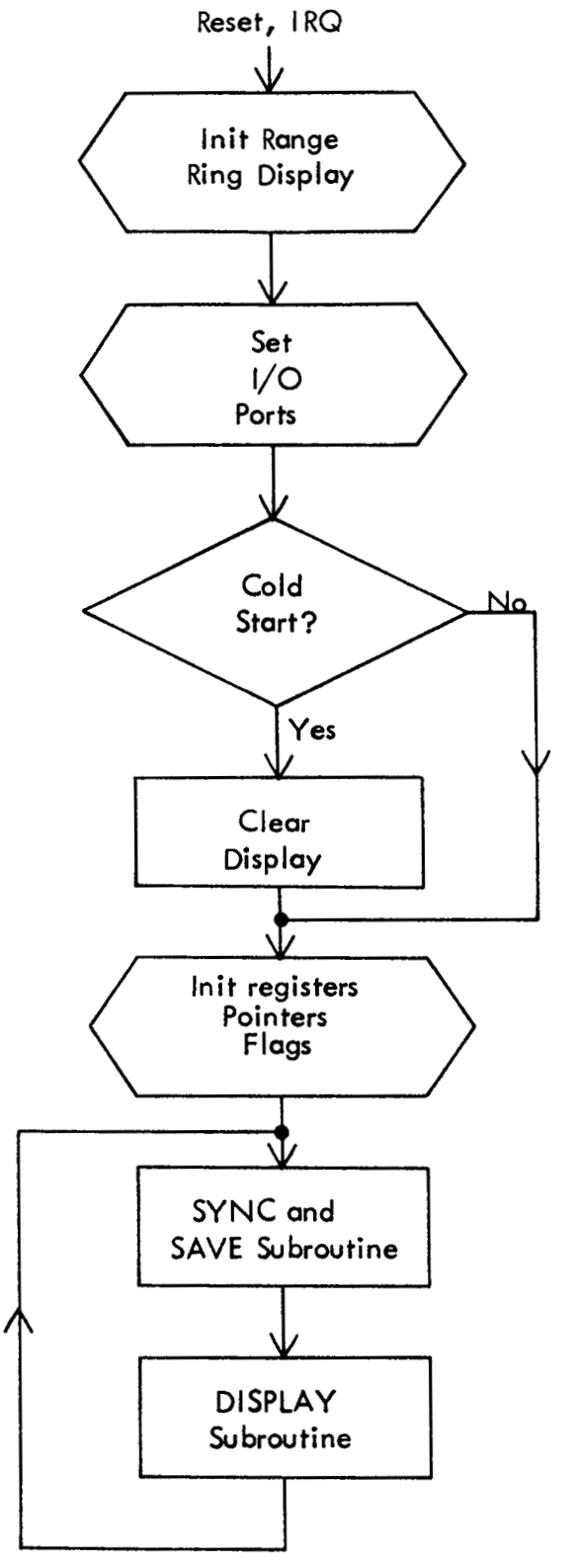


Figure 3-8. Receiver Software MAIN Loop.

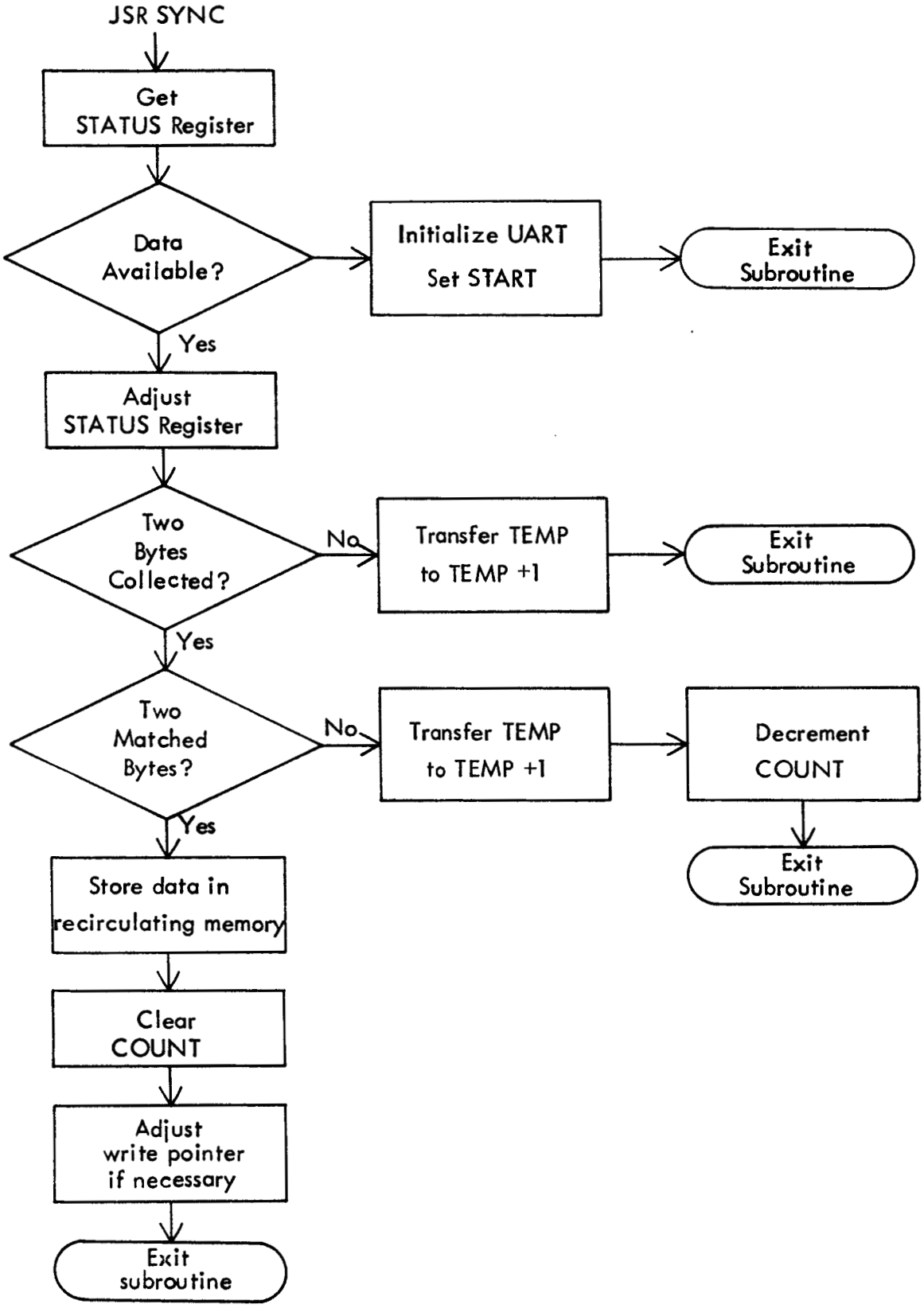


Figure 3-9. SYNC (and SAVE) Subroutine.

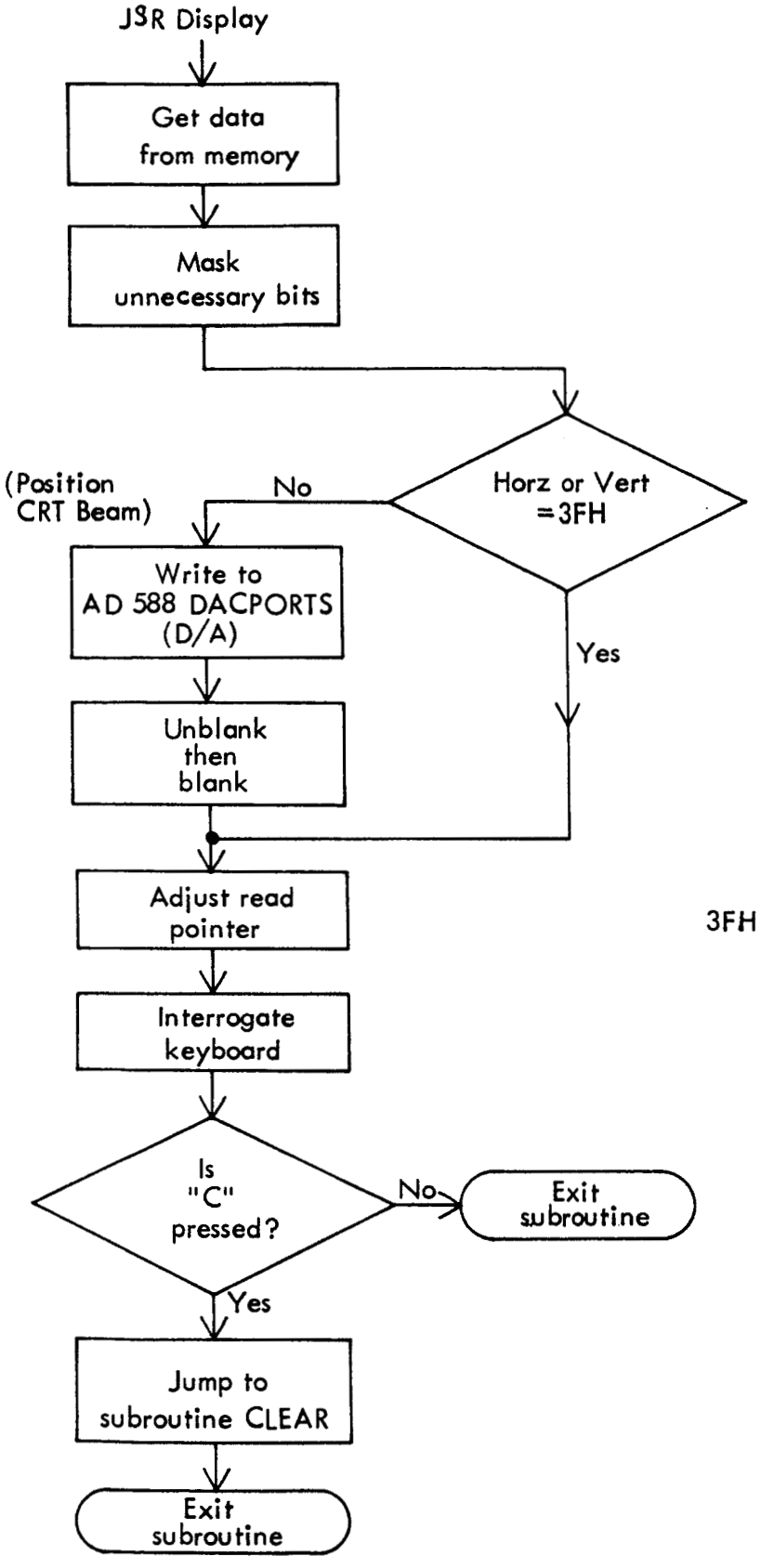


Figure 3-10. DISPLAY Subroutine.

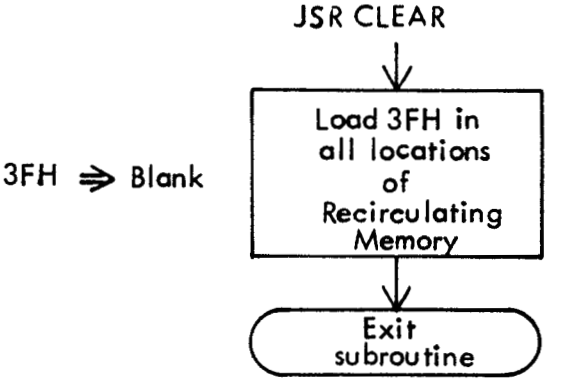


Figure 3-11. CLEAR Subroutine.

positional data from memory and using that data to place a dot on the CRT, the range rings are created by reading data provided in ROM and placing the data in the display queue.

Next, all of the I/O ports are programmed for proper direction and polarity (i.e., the blanking port is set as an output and is high) and a check is performed to determine if this is the first iteration. If it is the first iteration, all of the data in the display queue, with the exception of the range rings, is cleared. This way, the display will not show data until it is received from the transmitter. Finally, before entering the main loop, all of the registers, pointers, and conditional flags are cleared.

The main loop consists of jumping to two main subroutines, the SYNC and SAVE subroutine (hereafter called the SYNC subroutine) and the DISPLAY subroutine (see figures 3-9 and 3-10, respectively). The SYNC routine gets the STATUS register and checks bit 7. If it is low, then no new data is available. The START flag is checked to see if this is the first iteration of the SYNC routine. If so, the UART is set to receive and the START flag is set high so that this portion of the software is not entered again. The UART only needs to be reset once and operates asynchronously afterwards. The subroutine is then exited. If bit 7 is high, then new data is ready for processing, and the subroutine is continued. First the STATUS register is changed so that bit 7 is low, and the previous sync bit (bit 7 of the received data) is moved from bit 0 (of the STATUS register) to bit 1. The sync bit of the current data is obtained and stored in bit 0 of the STATUS register. The memory location COUNT is also incremented during this process.

The COUNT register is then interrogated, and if it is equal to zero or one, the new data received is stored in TEMP+1 and the subroutine is exited. If it is equal to two, then two bytes of data have been collected and the sync bits (bit 7) are compared. If they are not equal, the new data is transferred from TEMP to TEMP+1, COUNT is decremented, and the subroutine is exited. If the two sync bits are equal, the two matched data bytes are stored in the recirculating data display queue (indexed by the WRTI pointer), COUNT is cleared, and WRTI is incremented and then set to 25 if the incremented value is equal to 153. This way, the data bytes for the range rings are saved in locations 0-24, and 128 locations 26-152 are available for Stormscope data (128 vertical data bytes and 128 horizontal data bytes). Once adjusted properly, WRTI is stored and the SYNC subroutine is exited.

The main routine then turns software control over to the DISPLAY subroutine, which displays the dots placed into the recirculating display memory by the SYNC subroutine. The DISPLAY routine first fetches the READI index pointer and stores it into the X register. Then the horizontal and vertical data bytes are index retrieved from the recirculating memory, masked, and written to the appropriate D/A convertor. The Z axis of the oscilloscope (blanking) is then pulsed to turn the beam on, off again, and the READI pointer is incremented and saved.

The subroutine then checks to see if the 'C' key has been pressed on the KIM keyboard. If so, the control is given to the CLEAR subroutine (see Figure 3-11), which writes a blank in all of the recirculating memory locations (3F hex constitutes a blank). If the 'C' is not pressed, or upon reentering DISPLAY from CLEAR, program control is returned to the main program loop. If, when the horizontal and vertical byte are read and found to equal 63, the beam positioning and blanking portions of the DISPLAY routine are skipped and all other checks and changes occur as described above.

E. The RF Link

The RF link was completed by using two ordinary VHF aviation transceivers, the Bendix RT-241A as the transmitter and a Terra TPX 720 as the receiver.

On the transmitting side of the uplink, the FSK was obtained from the computer communication pair of a ComData Model 302B2-12 modem. The binary FSK was fed through a 1μf capacitor, for DC isolation, with a common ground, to drive the microphone input of the Bendix transceiver. The microphone input was keyed (grounded) continuously, so that the transmitter and UART operated asynchronously. The transmitter was turned off at intervals to remain in accordance with the duty cycle specifications of the Bendix transceiver. The transmitting modem was driven by a carrier generator so it would perform in a stand-alone mode, thus eliminating the need for a two-way link between the transmitting and receiving modems.

The receiving end of the uplink accessed the RF modulated data directly from the headphone output of the Terra transceiver. The audio data was fed through a 1μf capacitor, again for DC isolation, with a common ground and directly into the FSK input of a ComData Series 300 receive modem (this modem is usually interfaced with a computer to communicate with the user's terminal modems, such as the ComData 302B2-12), thus completing the RF link.

F. Conclusions and Recommendations

The sferics uplink was constructed and tested in the laboratory as defined by the design specifications. No in-flight tests were made and none are planned for the near future. However, new applications are being sought, one possibility being a ground-to-air data link now used in an MLS evaluation system. These applications will be flown, using the techniques described in this paper.

The playback capabilities were easily obtained by feeding "Byte Bucket" digital cassette recorder inputs with the standard RS-232C level data from the transmitting modem. The playback gave a high-speed replay of the sferics activity and truly displayed the storm tracking capabilities of the WX-7A Stormscope. Minor changes in software would allow for playback at normal speed.

Changes to the system for this particular application should include:

1. The range ring display should be removed from the video, and replaced by a glassine overlay. The dots currently used to show the range rings tend to confuse the observer.

2. The entire system should be repackaged for field experiments. This should include a printed circuit board and better electromagnetic shielding. This would eliminate the beam positioning problems that were experienced due to the high frequency noise on the outputs of the D/A convertors (small crosses were seen instead of dots).

IV. BIBLIOGRAPHY

[1] QRGB-256 LSI-11 Plug-In Singleboard Color Imaging System, Matrox Electronic Systems, Ltd.

[2] QFG-01 LSI-11 Frame Grabber Manual #154MO-03-0, Matrox Electronic Systems, Ltd.

[3] AIM-65 Microcomputer Users Guide, Rockwell International, Revision 2, March 1979.

[4] FORTH Users Manual, Rockwell International, June 1981.

[5] AIM-65 Floating Point Software Package, Rockwell International.

[6] Standard Mathematical Tables, CRC Press, Inc., Twenty-second edition, 1974, p. 374.

[7] Ryan, P., et al., "United States Patent No. 4,023,408 - Stormscope," Dytronics Company Inc., Columbus, Ohio, June 10, 1976.

[8] MOS/LSI Data Book, National Semiconductor Corp., 1977.

[9] Apple II Reference Manual, Apple Computers, Inc., 1979.

V. ACKNOWLEDGEMENTS

This research was sponsored by NASA Langley Research Center, Hampton, Virginia, under Grant NAG-1-124, in conjunction with NASA's Single Pilot IFR Research Program. The authors would like to express their appreciation to Dr. Robert W. Lilley, Associate Director of AEC, for his helpful suggestions and advice, and to AEC student interns Messrs. Douglas Dietz and David Hartwig for constructing the sferics uplink circuit designs in a most expeditious and accurate manner.

VI. APPENDICES

A. FORTH Software Description for CWRDD.

B. Custom Interface Description.

C. Transmitter Schematic.

D. Receiver Schematic.

E. Component Board Layout.

F. Software.
   1. Transmit
   2. Receive

G. Photographs.

A. FORTH Software Description For CWRDD

The CWRDD software adds 14 new words to the FORTH vocabulary that perform certain functions. The topmost word which is the one used to execute the entire program is ROTATE. The following is a description of the functions of each of the 14 new FORTH words. The words are described from most primitive to least primitive, e.g., each succeeding word requires the previous word. A listing of the FORTH source code follows the word description.

INIT

This word will perform initialization of the custom interface to the Matrox video display and frame grabber boards.

PIXADR

This word will, based on the contents of variables XADDR and YADDR, store these variable contents into the X address and Y address buffers of the Matrox display board. This must be done before writing or reading a pixel value from the display.

CDWRT

This word will take the contents of variable DATA and store it into the pixel on the matrox display board addressed by the previously stored X and Y addresses written using the PIXADR word. Additionally, the variable CNTRL is written into the control word of the Matrox display board.

CDREAD

This word is identical to CDWRT, except the pixel addressed is read instead of written. The contents of the Matrox control word is also read and placed in variable CNTRL.

SCREG

This word will write the contents of SCROLL to the scroll register of the Matrox display board. This is normally set to 0 for the CWRDD software.

MULT

This word will take two 32-bit values from the stack and multiply them together, leaving the result on the stack.

DIV

This word will take a 32-bit value from the stack and divide it by 1024 and return a 16-bit value to the stack. This is done to unscale the value scaled using the trigonometric functions. All trig functions are multiplied by 1024 since fixed point math is used in this software.

ADD

This word will add two 32-bit values from the stack and return a 32-bit value to the stack.

SUB

This word will change the sign of the top 32-bit value on the stack.

Typically, this is used in conjunction with the ADD word to perform subtraction of two values on the stack.

STICK

This word will read the joystick on both axes and return a value proportional to the left/right joystick position in a variable called HDG and a value proportional to the fore/aft joystick position in a variable called SPD.

INPUTH

This word will take a value from variable DEGRE, which is the heading angle, and convert it to floating point format and place it in the AIM-65 floating point software's floating point register.

WRITEP

This word performs the translation and rotation of the course-up display by implementing the calculations described in the software description of Section III of this report. These calculations are performed on the 52 x 52 pixel array.

NEXTPIX

This word determines what the next aircraft position is in the Matrox weather display field based on the variables input from the joystick. This word also writes the heading angle, speed, and x and y coordinates of the aircraft position.

ROTATE

This word is the main program word. Evoking this word will run the program for the CWRDD. This word includes all software value initialization and control of the selection of sync input to the frame grabber. Additionally, the sine and cosine values of aircraft heading are computed and scaled by 1024, and saved as constants for the WRITEP and NEXTPIX words. ROTATE executes forever or until the reset on the AIM-65 is pressed.

This software is the code necessary to produce the CWRDD using the AIM-65, Matrox Video Display, Frame Grabber, and a video tape recorder with recorded weather radar information. This software can be executed from the AIM-65 FORTH environment by issuing the word ROTATE.

```
DECIMAL
52 CONSTANT FRAME  ( SQUARE FRAME SIZE
26 CONSTANT HALFF  ( HALF FRAME SIZE
128 VARIABLE X  ( AIRCRAFT X COORDINATE
128 VARIABLE Y  ( AIRCRAFT Y COORDINATE
0 VARIABLE SINANG
0 VARIABLE COSANG
0 VARIABLE YI  ( AIRCRAFT X COORD. IN WX PLANE
0 VARIABLE XI  ( AIRCRAFT Y COORD. IN WX PLANE
0 VARIABLE DEGRE
0 VARIABLE SPEED
```

```
( WORD INPUTH LOADS HEADING IN FLOATING PT ACCUM.

: INPUTH
  DEGRE @ S>F ;

( WORD WRITEP WRITES THE COURSE UP DISPLAY

: WRITEP
FRAME 0
DO I 46 - YI !
FRAME 0
   DO I HALFF - XI !
     XI @ COSANG @ MULT
     YI @ SINANG @ MULT
     SUB ADD DIV
     X @ + XADDR C!
     XI @ SINANG @ MULT
     YI @ COSANG @ MULT
     ADD DIV
     Y @ + YADDR C!
   PIXADR
   CDREAD
   219 XI @ + XADDR C!
   199 YI @ + YADDR C!
   PIXADR
   02 CNTRL C!
   CDWRT
  LOOP
LOOP ;


( WORD NEXTPIX DETERMINES THE NEXT AC POSITION IN
( WX DISPLAY PLANE
: NEXTPIX
STICK HDG @ DUP 1 -
   0< IF
      DROP 1
      THEN
DUP 169 -
   0< NOT IF
         DROP 169
         THEN
1 - 359 168 */ 180 + DUP 360 -
0< NOT IF 360 - THEN DEGRE !
SPD @ DUP 1 -
   0< IF
      DROP 1
   THEN
DUP 147 -
   0< NOT IF
       DROP 147
```

```
          THEN
1 - 10 * 146 / SPEED !
  SPEED @ COSANG @ MULT DIV Y @ SWAP - Y !
  SPEED @ SINANG @ MULT DIV X @ + X !
CR DEGRE @ . SPEED @ . X @ . Y @ .  ;

( WORD ROTATE THIS WORD WILL PERFORM THE COURSE UP
( WEATHER DISPLAY DEMONSTRATOR

: ROTATE
128 X ! 128 Y ! 0 DEGRE ! 0 SPEED !
INIT
0 SCROLL C!
SCREG
BEGIN
BEGIN
    CDREAD
     CNTRL C@ 64 AND
0= UNTIL
 66 CNTRL C!
CDWRT
6 0 DO
INPUTH
 RADIANS PAD F>M SIN 1024 S>A F* F>S SINANG !
 PAD M>F COS 1024 S>A F* F>S COSANG !
WRITEP
NEXTPIX
LOOP
AGAIN ;

HEX

0 VARIABLE HDG
0 VARIABLE SPD

 ( WORD STICK RETURNS VALUE FOR HEADING IN HDG
 (             RETURNS VALUE FOR SPEED IN SPD

CODE STICK

0 # LDY,
20 # LDA,
A002 STA,
A000 STA,
BEGIN,
   A000 LDA,
   C0 # AND,
0= UNTIL,
00 # LDA,
A000 STA,
20 # LDA,
```

```
A000 STA,
BEGIN,
   A000 LDA,
   NOP,
   0< IF,
      INY,
      0= IF,
         FF # LDY,
         THEN,
      THEN,
NOP,
A000 LDA,
0< NOT UNTIL,
HDG STY,
BEGIN,
   A000 LDA,
   C0 # AND,
0= UNTIL,
0 # LDY,
00 # LDA,
A000 STA,
20 # LDA,
A000 STA,
BEGIN,
   A000 LDA,
   .A ASL,
   0< IF,
      INY,
      0= IF,
         FF # LDY,
         THEN,
      THEN,
A000 LDA,
.A ASL,
0< NOT UNTIL,
SPD STY,
NEXT JMP,
END-CODE

( VARIABLES FOR MULT AND DIV WORDS

 0 VARIABLE PROD
2 ALLOT
0 VARIABLE MC
2 ALLOT
0 VARIABLE MPLR
0 VARIABLE SI
HEX

( WORD MULT TAKES TWO 16 BIT WORDS AND MULTIPLIES
( THEM WITH A 32 BIT RESULT PUT ON STACK
```

```
CODE MULT
0 # LDA,
PROD STA,
PROD 1+ STA,
PROD 2+ STA,
PROD 3 + STA,
MC 2+ STA,
SI STA,
TOP 1+ LDA,
PHA,
80 # AND,
0<
    IF,
          PLA,
          FF # EOR,
          MC 1+ STA,
          TOP LDA,
          FF # EOR,
          CLC,
          01 # ADC,
          MC STA,
          00 # LDA,
          MC 1+ ADC,
          MC 1+ STA,
          80 # LDA,
          SI EOR,
          SI STA,
    ELSE,
          PLA,
          MC 1+ STA,
          TOP LDA,
          MC STA,
    THEN,
 SEC 1+ LDA,
 PHA,
 80 # AND,
 0<
    IF,
          PLA,
          FF # EOR,
          MPLR 1+ STA,
          SEC LDA,
          FF # EOR,
          CLC,
          01 # ADC,
          MPLR STA,
          00 # LDA,
          MPLR 1+ ADC,
          MPLR 1+ STA,
          80 # LDA,
          SI EOR,
```

```
         SI STA,
   ELSE,
         PLA,
         MPLR 1+ STA,
         SEC LDA,
         MPLR STA,
   THEN,
08 # LDY,
BEGIN,
   PROD ASL,
   PROD 1+ ROL,
   PROD 2+ ROL,
   PROD 3 + ROL,
   MPLR ASL,
   CS
      IF,
         CLC,
         MC LDA,
         PROD ADC,
         PROD STA,
         MC 1+ LDA,
         PROD 1+ ADC,
         PROD 1+ STA,
         MC 2+ LDA,
         PROD 2+ ADC,
         PROD 2+ STA,
   0 # LDA,
   PROD 3 + ADC,
   PROD 3 + STA,
      THEN,
   DEY,
   0=
UNTIL,
SI LDA,
0<
   IF,
         CLC,
         FF # LDA,
         PROD EOR,
         01 # ADC,
         TOP STA,
         FF # LDA,
         PROD 1+ EOR,
         0 # ADC,
         TOP 1+ STA,
         FF # LDA,
         PROD 2+ EOR,
         0 # ADC,
         SEC STA,
         FF # LDA,
         PROD 3 + EOR,
```

```
         0 # ADC,
         SEC 1+ STA,
   ELSE,

         PROD LDA,
         TOP STA,
         PROD 1+ LDA,
         TOP 1+ STA,
         PROD 2+ LDA,
         SEC STA,
         PROD 3 + LDA,
        SEC 1+ STA,
   THEN,
NEXT JMP,
END-CODE

( CODE WORD DIV TAKES 32 BIT VALUE FROM STACK AND
( DIVIDES BY 1024 RETURNS 16 BIT VALUE TO STACK

CODE DIV
02 # LDY,
BEGIN,
   TOP 3 + LSR,
   TOP 2+ ROR,
   TOP 1+ ROR,
   TOP ROR,
   DEY,
0=
UNTIL,
TOP 2+ LDA,
TOP 3 + STA,
TOP 1+ LDA,
TOP 2+ STA,
POP JMP,
END-CODE

( WORD SUB COMPLEMENT 32 BIT DATA ON STACK

HEX

CODE SUB
CLC,
   FF # LDA,
   TOP EOR,    ( GET LEAST SIG BYTE AND COMPLEMENT
   01 # ADC,    ( OBTAIN TWOS COMPLEMENT
   TOP STA,    ( SAVE IT
   FF # LDA,
   TOP 1+ EOR,    (  DO THE SAME TO THE NEXT BYTE
   00 # ADC,
   TOP 1+ STA,
   FF # LDA,
```

```
   TOP 2+ EOR,    ( AGAIN, TO THE NEXT BYTE
   00 # ADC,
   TOP 2+ STA,
   FF # LDA,
   TOP 3 + EOR,   (  AND AGAIN TO THE LAST BYTE
   00 # ADC,
   TOP 3 + STA,
NEXT JMP,
END-CODE

( CODE ADD ADD TOP TWO 32 BIT VALUES ON STACK
( RETURN 32 BIT WORD TO STACK
HEX
ASSEMBLER

CODE ADD
CLC,
TOP LDA,
TOP 4 + ADC,
TOP 4 + STA,
TOP 1+ LDA,
TOP 5 + ADC,
TOP 5 + STA,
TOP 2+ LDA,
TOP 6  + ADC,
TOP 6 + STA,
TOP 3 + LDA,
TOP 7 + ADC,
TOP 7 + STA,
POPTWO JMP,
END-CODE

( THIS IS THE CONSTANT AND VARIABLE LOCATIONS

HEX
8000 CONSTANT LOUT
8001 CONSTANT HOUT
8002 CONSTANT LIN
8003 CONSTANT HIN
A00F CONSTANT PIAA
A003 CONSTANT PIAD
A00B CONSTANT ACR
0 VARIABLE DATA
-1 ALLOT
FF VARIABLE CNTRL
-1 ALLOT
FF VARIABLE XADDR
-1 ALLOT
FF VARIABLE YADDR
-1 ALLOT
FF VARIABLE SCROLL
-1 ALLOT
```

```
( THIS IS THE CUSTOM INTERFACE INIT CODE

CODE INIT
7F # LDA,
PIAD STA,
0 # LDA,
ACR STA,
0 # LDA,
PIAA STA,
NEXT JMP,
END-CODE

( THIS IS THE STORE PIXEL ADDRESS CODE

CODE PIXADR
 FD # LDA,
LOUT STA,
0F # LDA,
HOUT STA,
08 # LDA,
PIAA STA,
09 # LDA,
PIAA STA,
01 # LDA,
PIAA STA,
FF # LDA,
XADDR EOR,
LOUT STA,
FF # LDA,
YADDR EOR,
HOUT STA,
5 # LDA,
PIAA STA,
BEGIN,
  PIAA LDA,
0< UNTIL,
01 # LDA,
PIAA STA,
0 # LDA,
PIAA STA,
NEXT JMP,
END-CODE

( THIS IS THE CONTROL AND DATA WORD WRITE CODE

CODE CDWRT
0F # LDA,
HOUT STA,
FF # LDA,
LOUT STA,
08 # LDA,
```

```
PIAA STA,
09 # LDA,
PIAA STA,
01 # LDA,
PIAA STA,
FF # LDA,
CNTRL EOR,
HOUT STA,
FF # LDA,
DATA EOR,
LOUT STA,
05 # LDA,
PIAA STA,
BEGIN,
  PIAA LDA,
0< UNTIL,
01 # LDA,
PIAA STA,
0 # LDA,
PIAA STA,
NEXT JMP,
END-CODE

( THIS IS THE CONTROL AND DATA WORD READ CODE

CODE CDREAD
0F # LDA,
HOUT STA,
FF # LDA,
LOUT STA,
08 # LDA,
PIAA STA,
09 # LDA,
PIAA STA,
03 # LDA,
PIAA STA,
BEGIN,
  PIAA LDA,
0< UNTIL,
FF # LDA,
LIN EOR,
DATA STA,
FF # LDA,
HIN EOR,
CNTRL STA,
1 # LDA,
PIAA STA,
0 # LDA,
PIAA STA,
NEXT JMP,
END-CODE
```

```
( THIS IS THE WRITE TO SCROLL REGISTER CODE

CODE SCREG
FB # LDA,
LOUT  STA,
0F # LDA,
HOUT STA,
08 # LDA,
PIAA STA,
09 # LDA,
PIAA STA,
01 # LDA,
PIAA STA,
FF # LDA,
SCROLL EOR,
LOUT STA,
05 # LDA,
PIAA STA,
BEGIN,
  PIAA LDA,
0< UNTIL,
01 # LDA,
PIAA STA,
0 # LDA,
PIAA STA,
NEXT JMP,
END-CODE
FINIS
```

B. Custom Interface Description.

1. Q-BUSS to AIM-65 Interface

Figure B-1 is the Q-BUSS to AIM-65 interface schematic. This circuit will interface the 16-bit bidirectional address and data buss of the Matrox boards to the separate 16-bit address and 8-bit bidirectional data buss of the AIM-65's 6502 microprocessor.

All of the data in and out of the AIM-65 is transferred through U1 which is an 8-bit bidirectional buss driver. All addresses and data on the Q-BUSS are complemented data; therefore, any data or addresses output from the AIM-65 must be complemented before transfer. For output of address or data from the AIM-65 to the Matrox Q-BUSS, device U1 is set up to input on the left and output on the right. The address or data from the AIM-65 is placed with high order 8 bits transferred to U4 and low order 8 bits transferred to U5. If the data contained in U$ and U5 is an address for the Q-BUSS, then the sync control line from the AIM-65 is asserted. If the data in U4 and U5 is data then DOUT is asserted, which enables the outputs (right side) of U4 and U5. The actual clocking of the data from the 8-bit AIM-65 data busses into U4 and U5 is a memory reference operation. Address $8000 is U5 and $8001 is U4. The Buss drivers U2 and U3 have all inputs and outputs floating during the address output operations from the AIM-65 to Q-BUSS because the CE bar inputs to U2 and U3 are disabled.

The process of reading data to the AIM-65 from the Q-BUSS also involves memory-mapped operations. The high 8 bits are read from U2 at $8003, and the low 8 bits are read from U3 at $8002. The read process involves asserting DIN, which causes U2 and U3 to have inputs on the right and outputs on the left. This makes the data on the 16-bit Q-BUSS available to U2 and U3 which transfer this data to the AIM-65 when reads to 8003 and $8002 are performed respectively. The outputs of U4 and U5 do not present a problem, as they are floated by keeping the output enables high.

This interface is relatively simple and relies on software for operation. Significant speed improvement could be realized if an all-hardware approach were used, but for the CWRDD the current implementation has performed successfully.

2. Composite Sync Generator

Figure B-2 is the schematic for the composite sync generator used to provide solid sync for viewing the weather radar displays. The circuit is implemented very easily by the use of a National Semiconductor Corp. MM5321 TV Camera Sync Generator chip [8]. Essentially, all that is needed is a source of 1.26 MHz and some TTL buffers. Devices U8 and U9 provide a buffered 12.60 MHz reference oscillator and a divide by 10 counter to produce 1.26 MHz at the output of U11-12. This drives the clock input to the MM5321 (U12) and with the proper condition strapping, as shown, a MOS level composite sync signal is produced at U12-16. This output is buffered and provided to a BNC connector on the custom interface board. Also provided is a buffered vertical driver output which can be used to externally sync

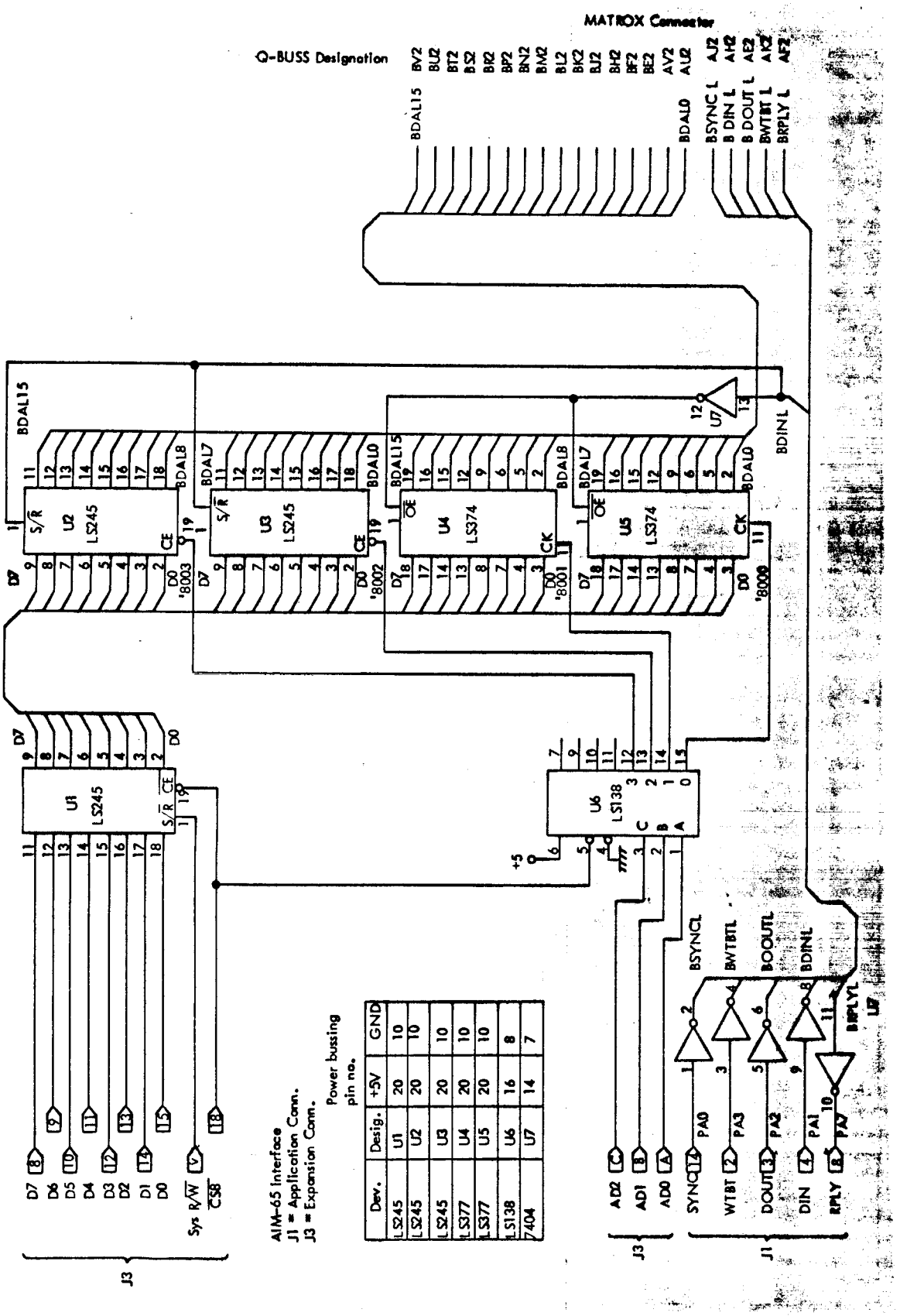
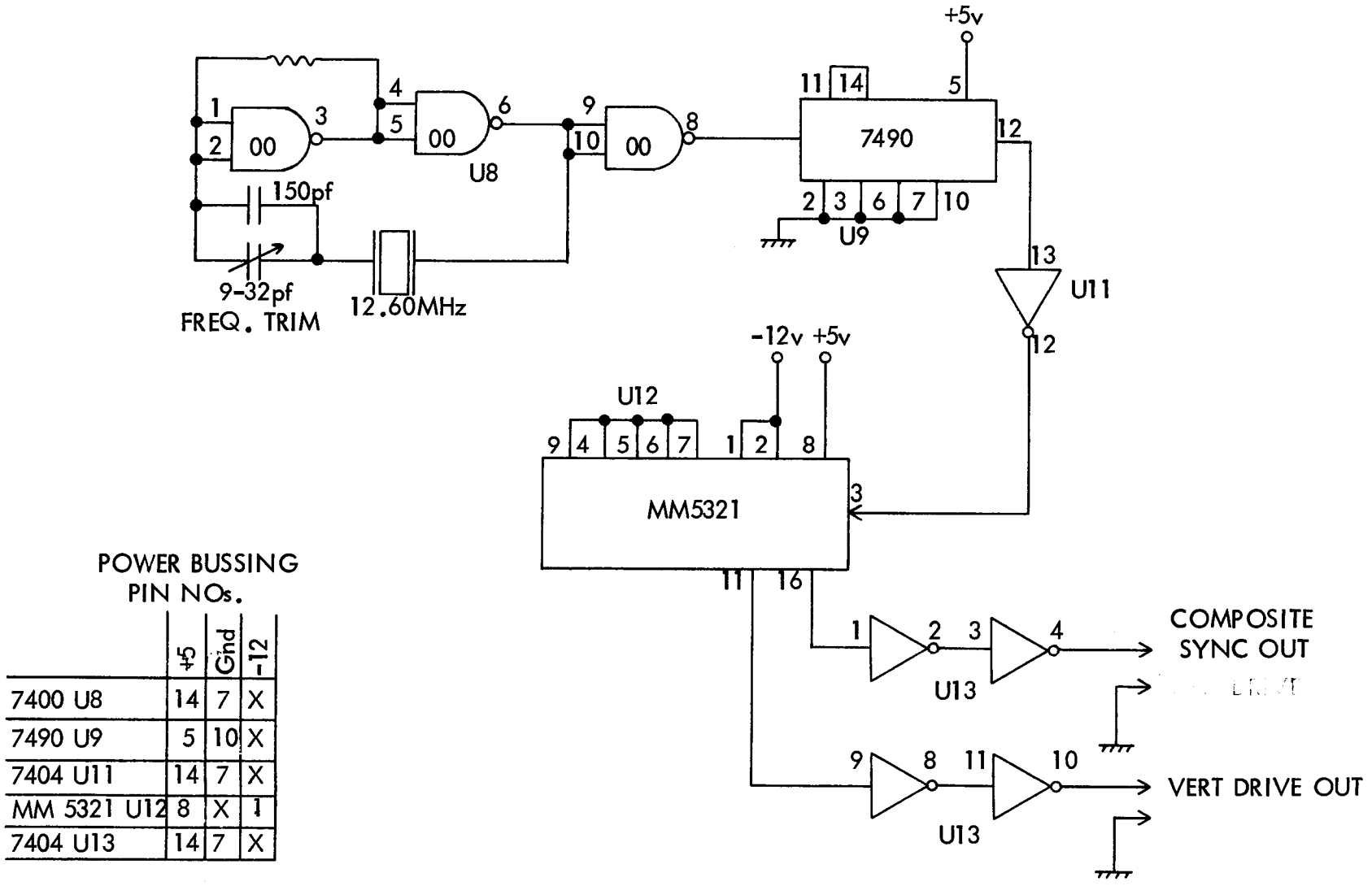


Figure B-1. Custom Interface; Q-BUS to AIM-65.



POWER BUSSING
PIN NOs.

	+5	Gnd	-12
7400 U8	14	7	X
7490 U9	5	10	X
7404 U11	14	7	X
MM 5321 U12	8	X	1
7404 U13	14	7	X

Figure B-2. Custom Interface Composite Sync Generator.

the VTR if necessary. The composite sync generator produces excellent stability, and video presented with this sync generator is very sharp in detail.

3. Joystick Hardware

The joystick schematic is presented in figure B-3. Each timer is set up in the triggered one shot mode such that a pulse on the trigger input will produce a pulse whose time duration is proportional to the position of the joystick pot for that axis. The FORTH assembly context software increments a counter until the pulse ends, and then the count that is accumulated is proportional to the joystick position. A common trigger is used because only one axis is measured at a time. The idea for this circuit is adapted from the similar implementation used in the APPLE II computer game paddle input [9]. Figure B-4 is the parts placement for the circuits included on the custom interface board.

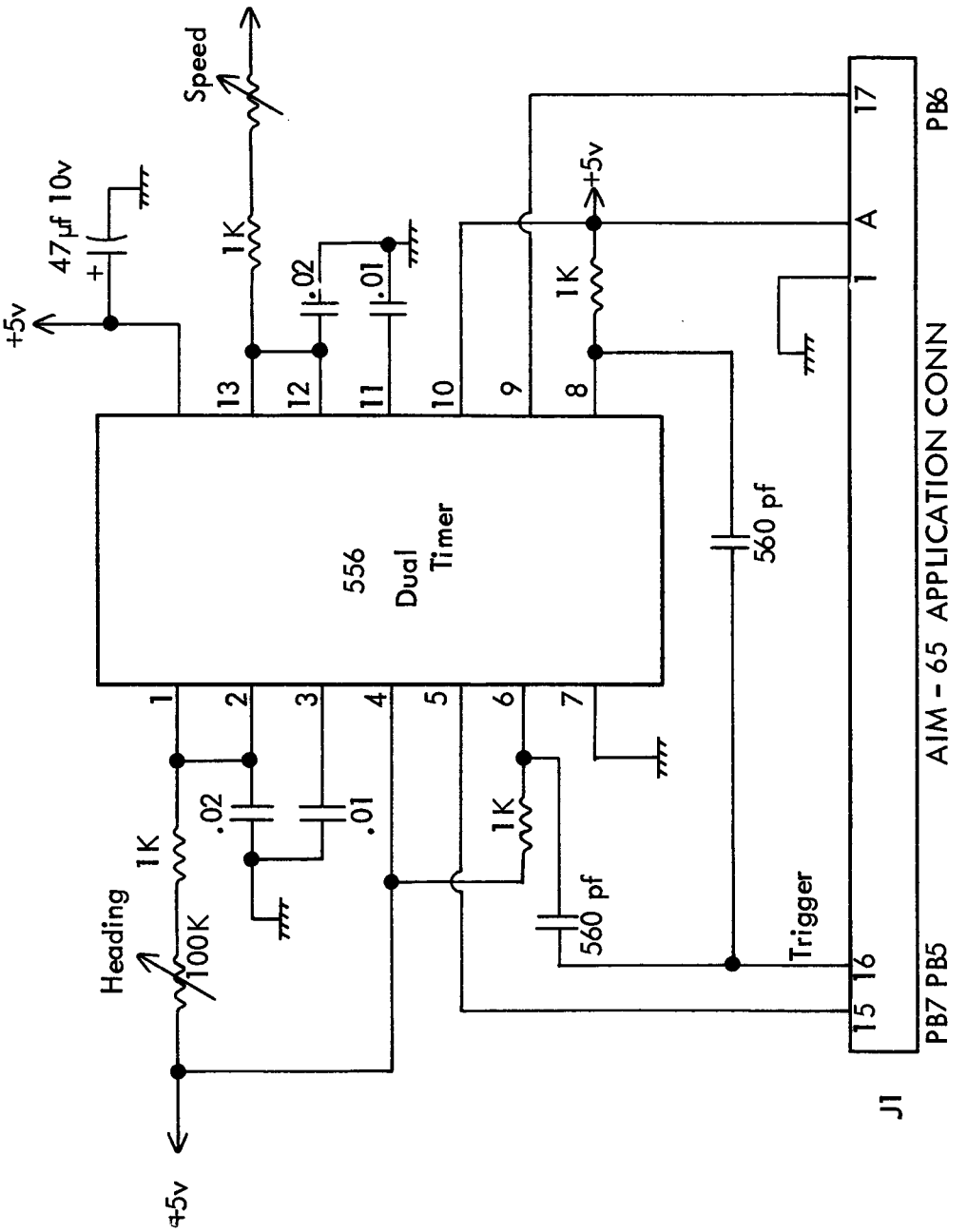


Figure B-3. Custom Interface Joystick.

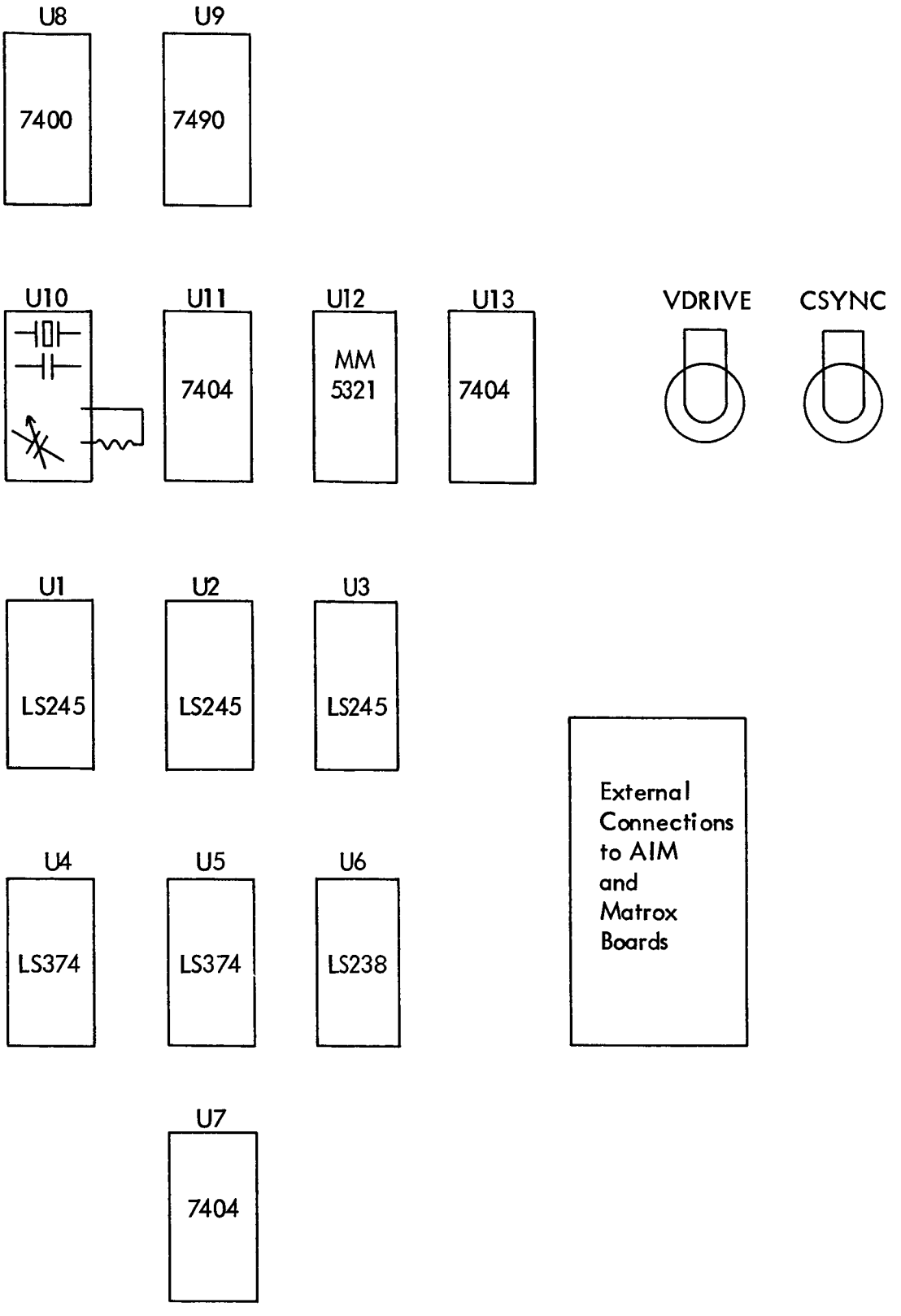
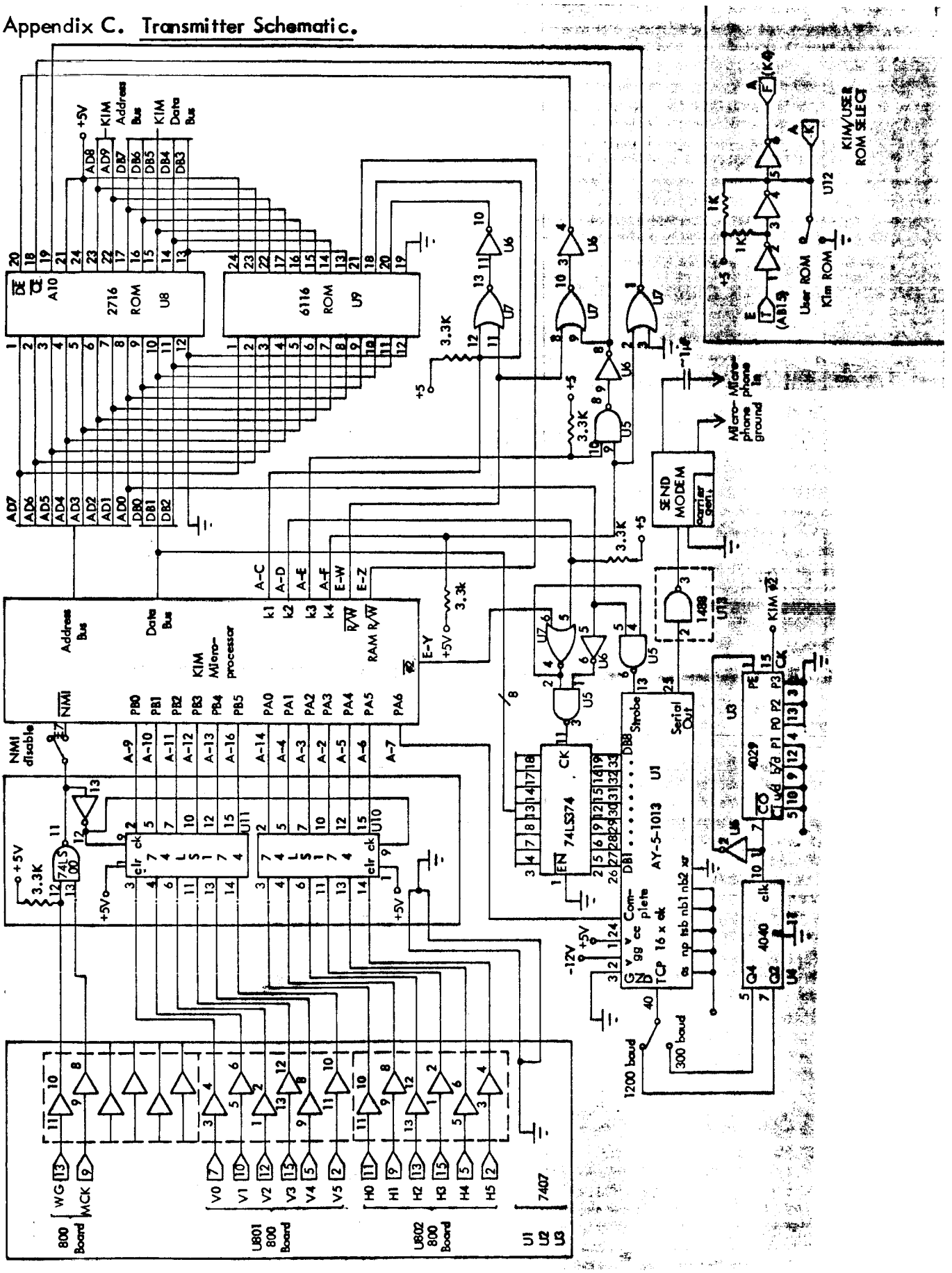
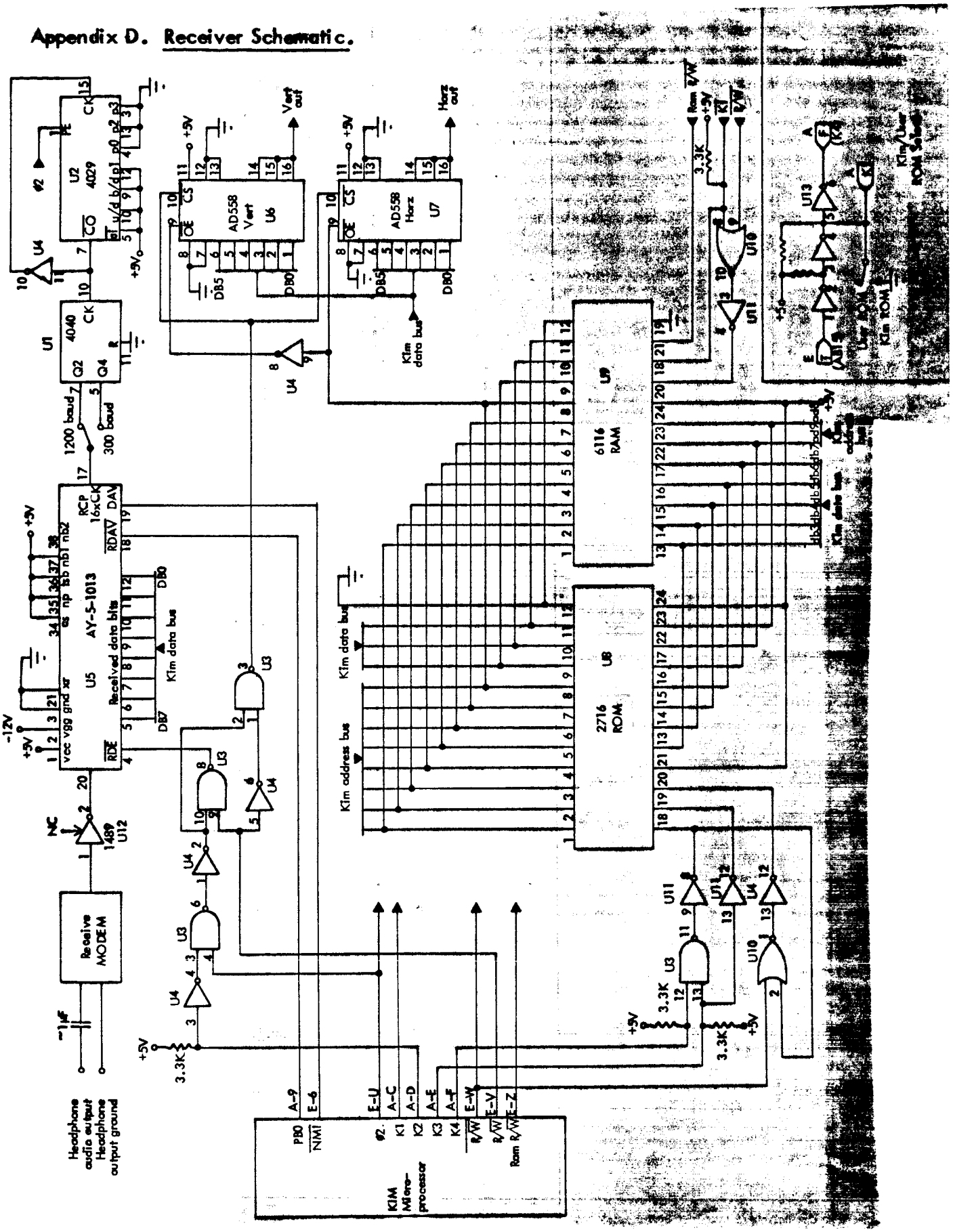


Figure B-4. Custom Interface Parts Layout.

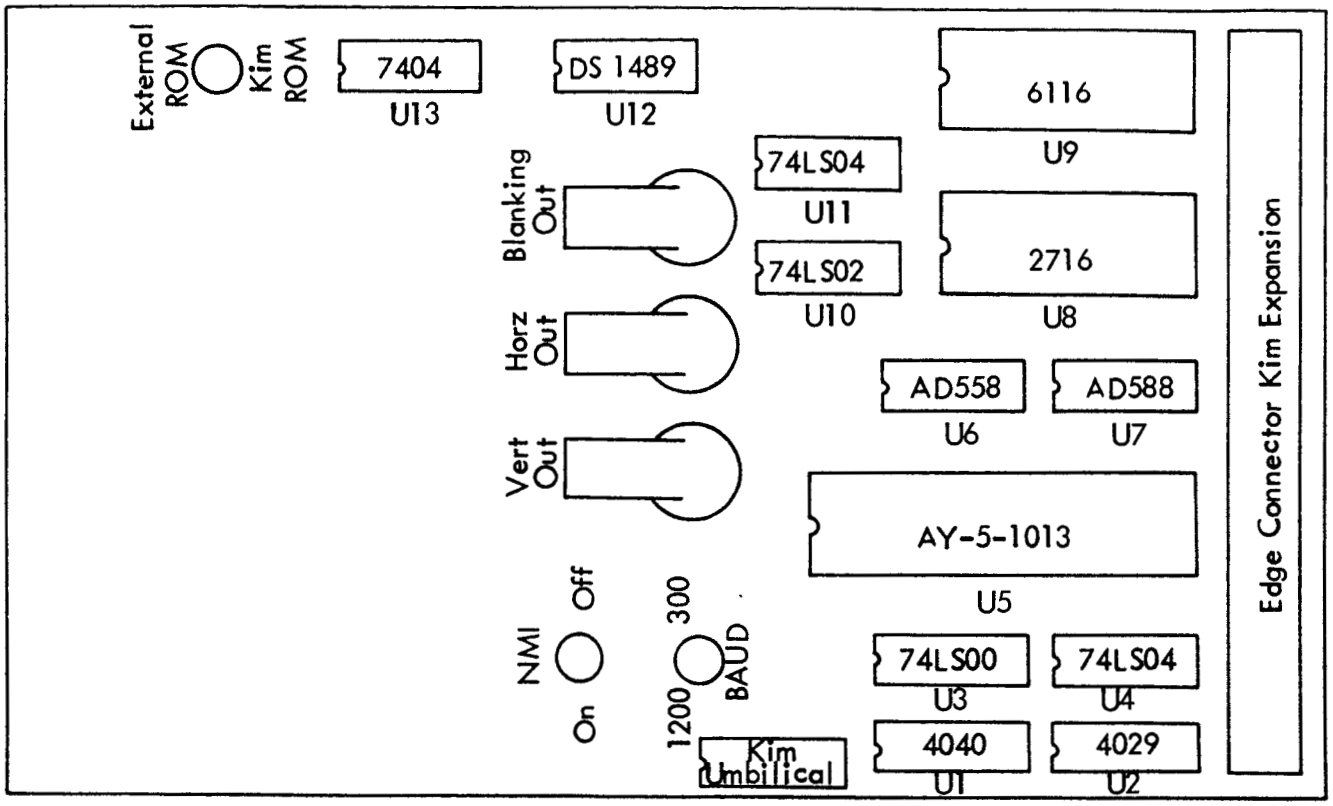
Appendix C. Transmitter Schematic.



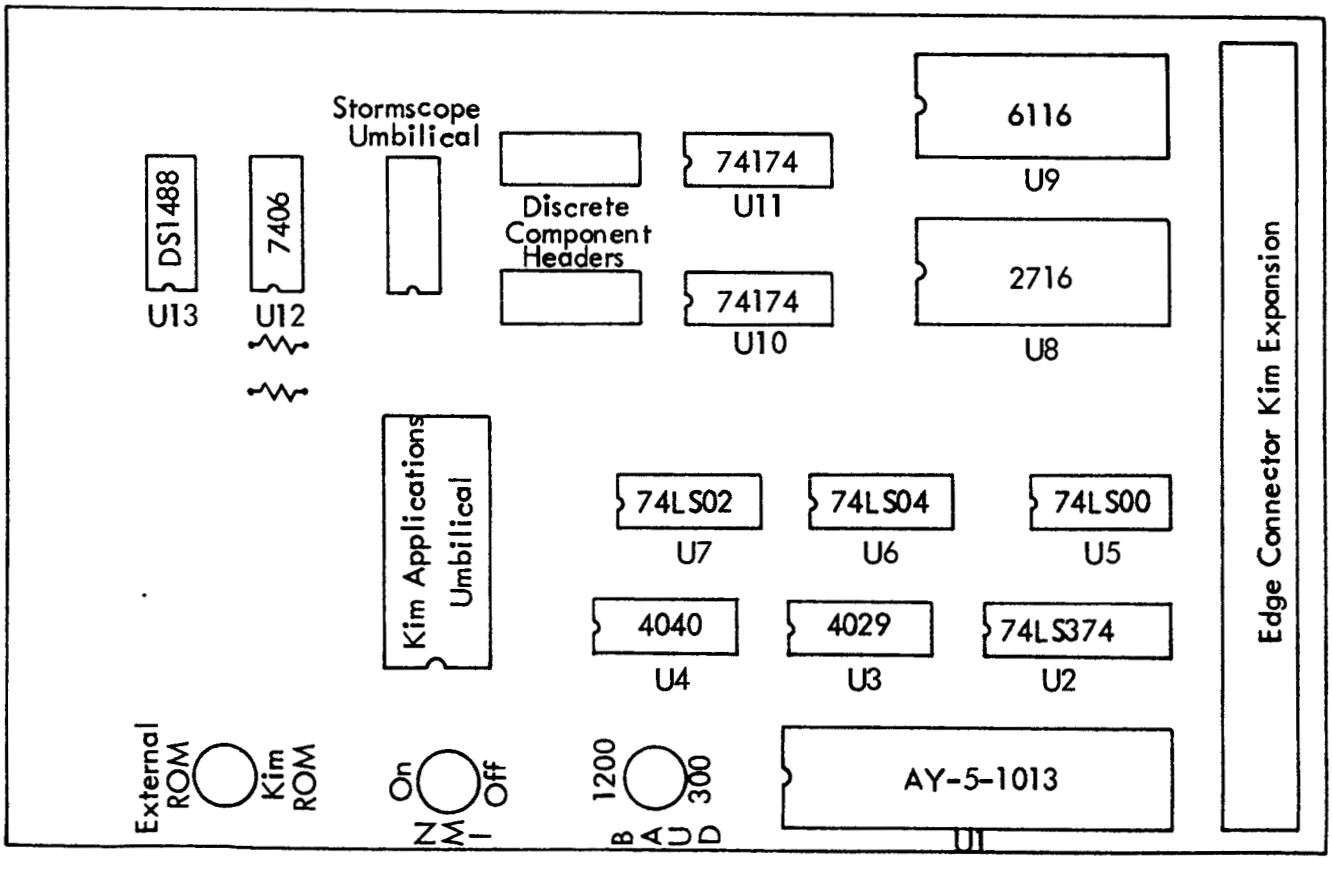
Appendix D. Receiver Schematic.



Appendix E. Component Board Layout.



Receiver Component Layout



Transmitter Component Layout

Appendix F. Software.

1. Transmit.

```
*************************************************************************
*
*          ADDRESS DEFINITIONS AND RESERVED MEMORY
*
*************************************************************************
*
*
       ORG $0000
PIAA   EQU $1700        PIAA DATA PORT ADDRESS
DDRA   EQU $1701        PIAA DATA DIRECTION REGISTER
PIAB   EQU $1702        PIAB DATA PORT ADDRESS
DDRB   EQU $1703        PIAB DATA DIRECTION REGISTER
UDAT   EQU $0800        UART DATA TO BE TRANSMITTED ADDRESS
STROBE EQU $0801        ADDRESS TO BE ACCESSED TO SEND DATA VIA UART
DAVPNT BSS 1            DATA AVAILABLE TO SEND INDEX POINTER
UPNT   BSS 1            UART DATA INDEX POINTER
DATFLG BSS 1            DATA FLAG TO DETERMINE SENSE OF 8TH BIT OF DATA
*
*
*************************************************************************
*
*          HORIZONTAL AND VERTICAL DATA MEMORY
*
*************************************************************************
*
*
       ORG $0200
HORZ   BSS 256
VERT   BSS 256
*
*
*************************************************************************
*
*          MAIN PROGRAM
*
*************************************************************************
*
*
       ORG $0C00
*
*          PROGRAM INITIALIZATION
*
REF    LDA 0
       STA DAVPNT       ZERO THE DATA AVAILABLE INDEX POINTER
       STA UPNT         ZERO THE UART DATA INDEX POINTER
       STA DATFLG       ZERO THE DATA FLAG
       LDA =%00000000
       STA DDRA         PORT A ALL INPUTS
       LDA =%00000000
       STA DDRB         PORT B ALL INPUTS
*
*          LOOP UNTIL DATA IS AVAILABLE
*
LOOP1  LDA DAVPNT       GET DATA AVAILABLE INDEX POINTER
       CMP UPNT         COMPARE IT WITH THE UART DATA INDEX POINTER
```

```
*
*       IF UPNT = DAVPNT, THEN ALL THE DATA AVAILABLE HAS BEEN SERVICED
*   AND TRANSMITTED.
*       IF UPNT < DAVPNT, THEN (DAVPNT-UPNT) SETS OF DATA STILL REQUIRE
*   ATTENTION.
*
        BEQ LOOP1       LOOP UNTIL DATA IS AVAILABLE
*
*           WHEN DATA IS AVAILABLE, PROCESS AND SEND DATA
*
LOOP2   LDX UPNT        GET UART DATA INDEX POINTER
        LDA DATFLG      GET THE DATA FLAG
*
*          BECAUSE THE DATA MUST BE PAIRED, EVERY HORIZONTAL BYTE MATCHED
*      WITH ITS RESPECTIVE VERTICAL BYTE,  THE EIGHTH BIT OF BOTH BYTES
*      OF EVERY OTHER PAIR ALTERNATE  0'S AND 1'S.
*
*                HORIZONTAL N   = 1XXXXXXX
*                  VERTICAL N   = 1XXXXXXX
*
*                HORIZONTAL N+1 = 0XXXXXXX
*                  VERTICAL N+1 = 0XXXXXXX
*
*
*
        BEQ ZERO        IF THE DATA FLAG IS ZERO, 8TH BIT = 0, SO JUMP
*
*           IF DATA FLAG IS A ONE, THEN 8TH BIT = 1, SO CONTINUE
*
*
*         ROUTINE TO SET THE 8TH BIT = 1 AND SEND
*
*
        LDA HORZ,X      GET HORIZONTAL DATA
        ORA =%10000000  SET THE 8TH BIT
        JSR SEND        SEND IT !
        LDA VERT,X      GET VERTICAL DATA
        ORA =%10000000  SET THE 8TH BIT
        JSR SEND        SEND IT !
        DEC DATFLG      CLEAR THE DATA FLAG AND USE ZEROS NEXT
        JMP SENT        JUMP OVER THE ZEROS ROUTINE
*
*
*       ROUTINE TO CLEAR THE 8TH BIT AND SEND
*
*
ZERO    LDA HORZ,X      GET HORIZONTAL DATA
        AND =%01111111  CLEAR THE 8TH BIT
        JSR SEND        SEND IT !
        LDA VERT,X      GET VERTICAL DATA
        AND =%01111111  CLEAR THE 8TH BIT
        JSR SEND        SEND IT !
        INC DATFLG      SET THE DATA FLAG AND USE ONES NEXT
SENT    INC UPNT        INCREMENT THE UART DATA POINTER TO GET NEW DATA
        JMP LOOP1       LET'S DO IT AGAIN .....
```

```
*****************************************************************
*
*          THIS SUBROUTINE WILL :
*
*               1)  INTEROGATE THE UART TO SEE IF IT HAS COMPLETED TRANS-
*                 MITTED THE PREVIOUS CHARACTER.
*
*               2)  IF THE UART IS NOT BUSY, THE UART STORBE WILL BE ACCESSED
*                 TO BEGIN THE TRANSMISSION OF A NEW CHARACTER.
*
*               3)  IF THE UART IS BUSY, THE PROGRAM WILL LOOP UNTIL THE
*                 UART SIGNIFIES THAT IT IS READY TO BEGIN TRANSMISSION
*                 OF A NEW CHARACTER AND RETURN TO THE MAIN PROGRAM.
*
*****************************************************************************
*
*
SEND    STA UDAT         SEND DATA TO UART
LOOP3   LDA PIAA         GET UART READY FLAG (PA6 OF PIAA)
        AND =%01000000    IS IT A ZERO
        BEQ LOOP3        IF SO THE UART IS BUSY, LOOP BACK AND TRY AGAIN
        STA STROBE       STROBE THE UART TO TRANSMIT NEW DATA BYTE
*
        LDY =105         SET DELAY APPROX 417 US
LOOP4   DEY              DECREMENT TIME
        BNE LOOP4        IF NOT ZERO LOOP
*
LOOP5   LDA PIAA         GET END OF CHARACTER FLAG
        AND =%01000000    MAKE SURE UART GETS BUSY
        BNE LOOP5
        RTS              ALL DONE, GO BACK TO MAIN PROGRAM
*
*
*****************************************************************
*
*
*          THIS IS THE NMI SERVICE ROUTINE. UPON RECEIVING AN INTERUPT THIS
*     PROGRAM WILL:
*
*               1)   SAVE THE A AND X REGISTERS
*
*               2)   OBTAIN AN INDEX POINTER
*
*               3)   READ TWO BYTES FROM THE PERIPHERAL (THE STORMSCOPE)
*
*               4)   USE THE POINTER TO STORE THE TWO BYTES IN THEIR PROPER
*                  MEMORY LOCATIONS.
*
*               5)   INCREMENT THE POINTER FOR THE NEXT NMI
*
*               6)   RESTORE THE A AND X REGISTERS
*
*               7)   RETURN TO THE MAIN PROGRAM
*
*
```

```
*************************************************************************
*
*
NMI     PHA             SAVE THE A REGISTER
        TXA             GET THE X INDEX REGISTER
        PHA             SAVE IT
        LDX DAVPNT      GET THE DATA AVAILABLE INDEX POINTER
        LDA PIAA        GET THE HORIZONTAL DATA (PA0-PA5)
        AND =$3F        MASK OFF UNNEEDED BITS
        STA HORZ,X      STORE DATA FOR UART
        LDA PIAB        GET VERTICAL DATA (PB0-PB5)
        AND =$3F        MASK OFF UNNEEDED BITS
        STA VERT,X      STORE DATA FOR UART
        INC DAVPNT      INCREMENT THE DATA AVAIL. POINTER FOR NEXT NMI
        PLA             GET OLD X REGISTER VALUE
        TAX             RESTORE X REGISTER
        PLA             RESTORE A REGISTER
        RTI             GO BACK TO THE MAIN PROGRAM
*
*************************************************************************
*
*          SET VECTORS FOR ROM
*
*************************************************************************
        ORG REF+$07FA
NMIVEC  ADR NMI
RSTVEC  ADR REF
IRQVEC  ADR REF
        END
```

2. Receive

```
*************************************************************************
*     THIS ASSEMBLY CODE WILL RECEIVE RS-232 WORDS FROM A MODEM AND
*     DISPLAY THEM ON AN X Y OSCILLOSCOPE AS DOTS IN A FIELD OF 64 X 64
*     TO MIMIC THE DISPLAY ON THE RYAN 3M STORMSCOPE.  THE FORMAT OF
*     THE RECEIVED WORDS MUST ALTERNATE BETWEEN HORIZONTAL AND VERTICAL
*     IN THAT ORDER WITH EVERY PAIR HAVING THE D7 BIT EITHER SET OR RESET.
*
*************************************************************************
*
*
*
*************************************************************************
*
*     ADDRESS DEFINITIONS AND MEMORY ALLOCATION
*
*************************************************************************
*
*
       ORG $0000
PIA    EQU $1700
PIAD   EQU $1701
PIBD   EQU $1703
PAB    EQU $1702
SPIAA  EQU $1740
SPIAB  EQU $1742
SPIAD  EQU $1741
SPIBD  EQU $1743
OUTH   EQU $0800       HORIZ D/A
OUTV   EQU $0801       VERT D/A
UART   EQU $0800       UART INPUT PORT
STATUS BSS 1           SOFTWARE STATUS WORD
COUNT  BSS 1           INPUT LOOP COUNT
TEMP   BSS 2           TEMP STORAGE FOR INPUT
WRTI   BSS 1           INDEX TO WRITE TABLE
READI  BSS 1           INDEX TO READ TABLE
COLD   BSS 1
START  BSS 1
*
*************************************************************************
*
*     READ AND WRITE BUFFER STORAGE AREAS
*
*************************************************************************
*
       ORG $0200
VERT   BSS 256
HORIZ  BSS 256
*
*************************************************************************
*
*     MAIN PROGRAM STARTS HERE
*
*************************************************************************
*
*
```

```
        ORG $0C00
VEC     LDX =$00
INIT    LDA HORIZ2,X
        STA HORIZ,X
        LDA VERT2,X
        STA VERT,X
        TXA
        CMP =24
        BEQ REF
        INX
        JMP INIT
REF     LDA =$01
        STA PIAD
        STA PIBD
        LDA =01          BLANK DISPLAY
        STA PIA
        LDA =$AA
        CMP COLD         TEST IF COLD START
        BEQ CONT1        JUMP IF NOT COLD START
        JSR CLEAR        CLEAR THE DISPLAY
        LDA =$AA
        STA COLD         SET WARM START FLAG
CONT1   LDA =0
        STA STATUS       CLEAR STATUS
        STA TEMP         CLEAR IN WORDS
        STA TEMP+1
        STA COUNT        CLEAR INPUT INDEX
        STA START        CLEAR START FLAG
        LDA =01
        STA PAB
        LDA =25          SET FIRST WRITE ADDR
        STA WRTI         SAVE INDEX
        LDA =$FF
        STA SPIBD        SET ALL BITS OUT
        LDA =00
        STA SPIAD        SET ALL BITS IN
OVER    JSR SYNC         DO THE SYNC AND SAVE
        JSR DISPLA       JUMP TO DISPLAY ROUTINE
        JMP OVER         DO IT AGAIN
*
*
*********************************************************************************
*
*      THE SYNC AND SAVE ROUTINES ARE HERE
*
*********************************************************************************
*
*
SYNC    LDA STATUS       GET STATUS
        BPL END          DATA AVIAL SET
        ASL A            MOVE LAST TO PREVIOUS
        AND =02          MASK BITS
        ORA STATUS       MERGE TO STATUS
        AND =$7F         MASK BITS
        STA STATUS       RESET DATA AVIAL BIT
```

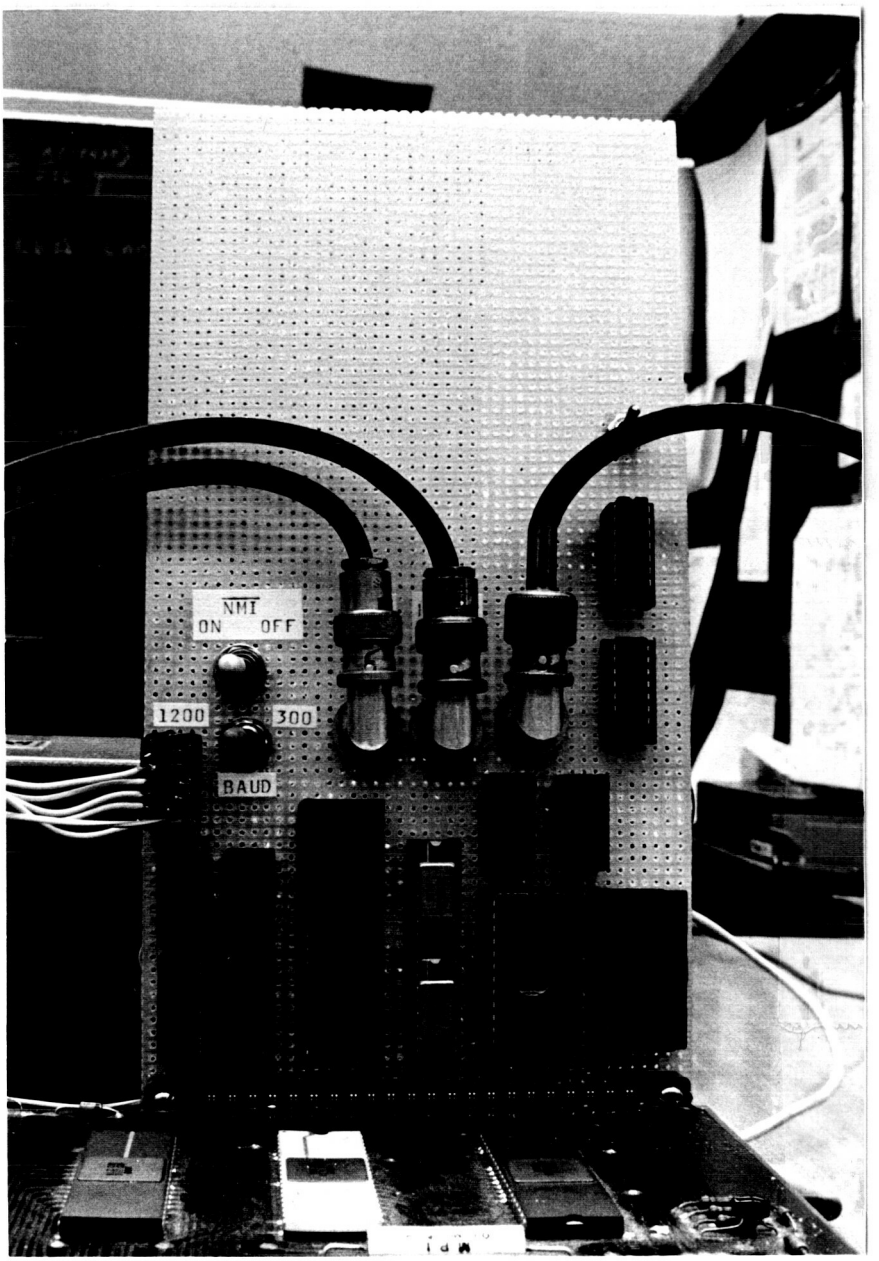
```
        INC COUNT
        LDA =$FF        SET START FLAG
        STA START
        CLC
        LDA TEMP        GET SYNC BIT
        ROL A
        ROL A
        AND =01         MASK ALL BUT LAST BIT
        ORA STATUS      SET LAST BIT
        STA STATUS      SAVE TO STATUS
        LDA =02         COUNT=2 ?
        CMP COUNT       IF YES CONTINUE
        BNE END3        TRY AGAIN
        LDA STATUS      GET STATUS
        AND =03         MASK FLAG BITS
        BEQ SAVE        IF BOTH =0 SAVE WORDS
        EOR =$03
        BEQ SAVE        IF BOTH =1 SAVE WORDS
        LDA TEMP        GET LAST WORD
        STA TEMP+1      SAVE TO PREVIOUS
        DEC COUNT       SET COUNT DO AGAIN
        JMP END
END3    LDA TEMP        GET FIRST WORD
        STA TEMP+1      PUT TO LAST WORD
END     LDA START
        BNE END2        IF NOT 0 TICKLE UART
        LDA UART
        LDA =00
        STA PAB
        LDA =01
        STA PAB
END2    RTS
SAVE    LDX WRTI        GET WRITE INDEX
        LDA TEMP+1      GET HORIZ VALUE
        STA HORIZ,X     SAVE IN TABLE
        LDA TEMP        GET VERT VALUE
        STA VERT,X      SAVE IN TABLE
        LDA =0
        STA COUNT
        INX             BUMP INDEX
        CPX =153        TEST IF 128TH ADDR.
        BNE CONT2       IF NOT 128TH SKIP
        LDX =25         SET TO BEGIN OF BUFF
CONT2   STX WRTI        SAVE INDEX
        JMP END         BYE
*
*
*********************************************************************************
*
*      THE ROUTINE TO DISPLAY THE DOTS
*
*********************************************************************************
*
*
DISPLA LDX READI
```

```
        LDA HORIZ,X     GET HORIZ WORD
        AND =$3F        MASK OFF TOP TWO
        CMP =63         TEST IF BLANK WORD
        BEQ END1        IF YES END
        LDA VERT,X
        CMP =63
        BEQ END1
        STA OUTV
        LDA HORIZ,X
        STA OUTH
        LDA =$00        SET UNBLANK
        STA PIA         UNBLANK
        LDA =01         SET BLANK
        STA PIA         BLANK
END1    INC READI       BUMP TO NEXT WORD
        LDA READI       GET READ ADDR
        CMP =153        TEST IF 128TH ADDR.
        BNE END4        IF NOT 128TH SKIP
        LDA =0
        STA READI       SET TO BUFF START
END4    LDA =02
        STA SPIAB       PULL C KEY DRIVER LOW
        LDA SPIAA       GET CROSS POINT OF C
        AND =02         MASK ALL BUT LAST
        BNE DONE        IF NOT C KEY THEN CONTINUE
        JSR CLEAR       CLEAR THE DISPLAY
DONE    RTS             RETURN
*
*
*************************************************************************
*
*      THIS ROUTINE CLEARS THE DISPLAY EXCEPT THE REFERENCE MARKS
*
*************************************************************************
*
*
CLEAR   LDX =25         START INDEX ADDRESS
        LDA =63         BLANKING VALUE
LOOP    STA HORIZ,X     CLEAR THE DOT
        INX             BUMP TO NEXT
        CPX =153
        BNE LOOP        LOOP TILL DONE 128 DOTS
        RTS             RETURN
*
*
*************************************************************************
*
*     DELAY ROUTINE TO DWELL ON OUTPUT DOT
*
*************************************************************************
*
*
DELAY   DEY
        BNE DELAY
        RTS
```

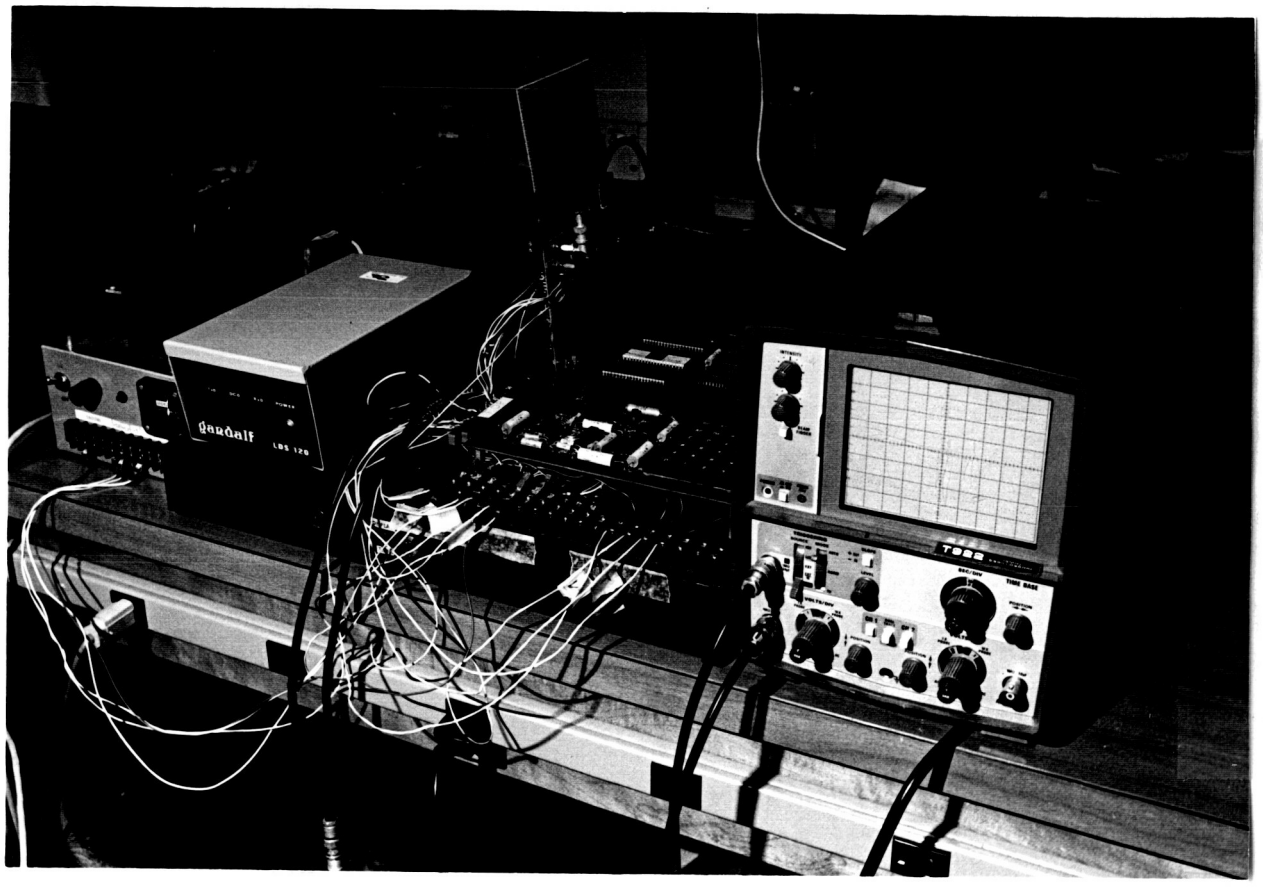
```
*
*
*********************************************************************************
*
*      INTERUPT ROUTINE
*
*********************************************************************************
*
*
NMI     PHA             SAVE A
        TXA             GET X
        PHA             SAVE X
        LDA UART        READ UART INPUT
        STA TEMP        SAVE TO TEMP MEMORY
        LDA STATUS      GET STATUS
        ORA =$80        SET DATA AVAIL.
        STA STATUS      SAVE STATUS
        LDA =00
        STA PAB
        LDA =01
        STA PAB
        PLA             GET SAVED X
        TAX             RETURN TO X
        PLA             GET SAVED ACCUM.
        RTI             RETURN FROM INTERUPT
*
*
*
VERT2   HEX 1F,1F,2C,36,3A,36,2C,1F,11,08,04,08,11
        HEX 1F,26,2B,2D,2B,26,1F,18,13,11,13,18
HORIZ2  HEX 1F,3A,36,2C,1F,11,08,04,08,11,1F,2C,36
        HEX 2D,2B,26,1F,18,13,11,13,18,1F,26,2B
        ORG VEC+$07FA
NMIVEC  ADR NMI
RSTVEC  ADR VEC
IRQVEC  ADR VEC
        END
```

Appendix G. Photographs. (Photos by James Nickum)

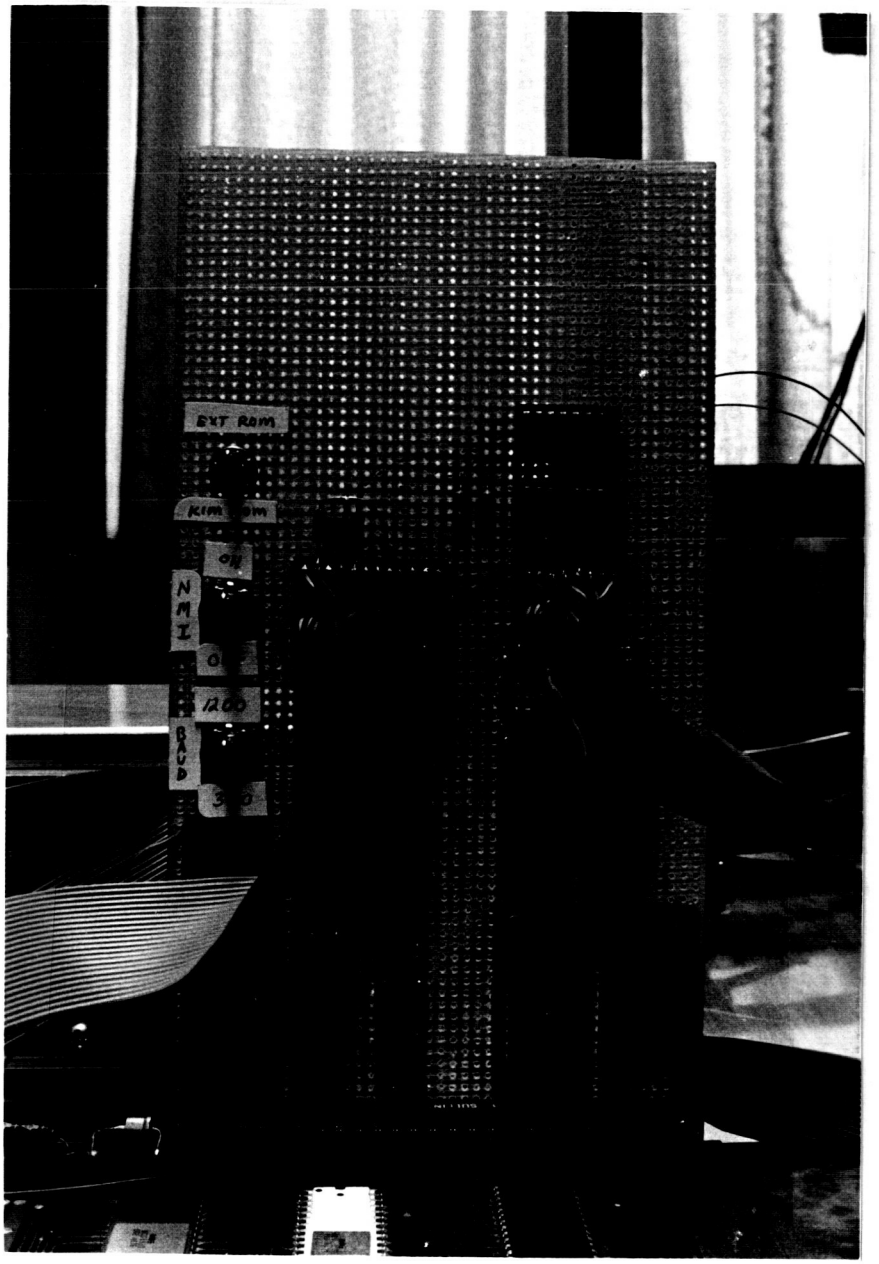
Uplink Processor Hardware, Transmission Portion, Interfaced with the Stormscope (Modem and VHF Transmitter Not Shown).



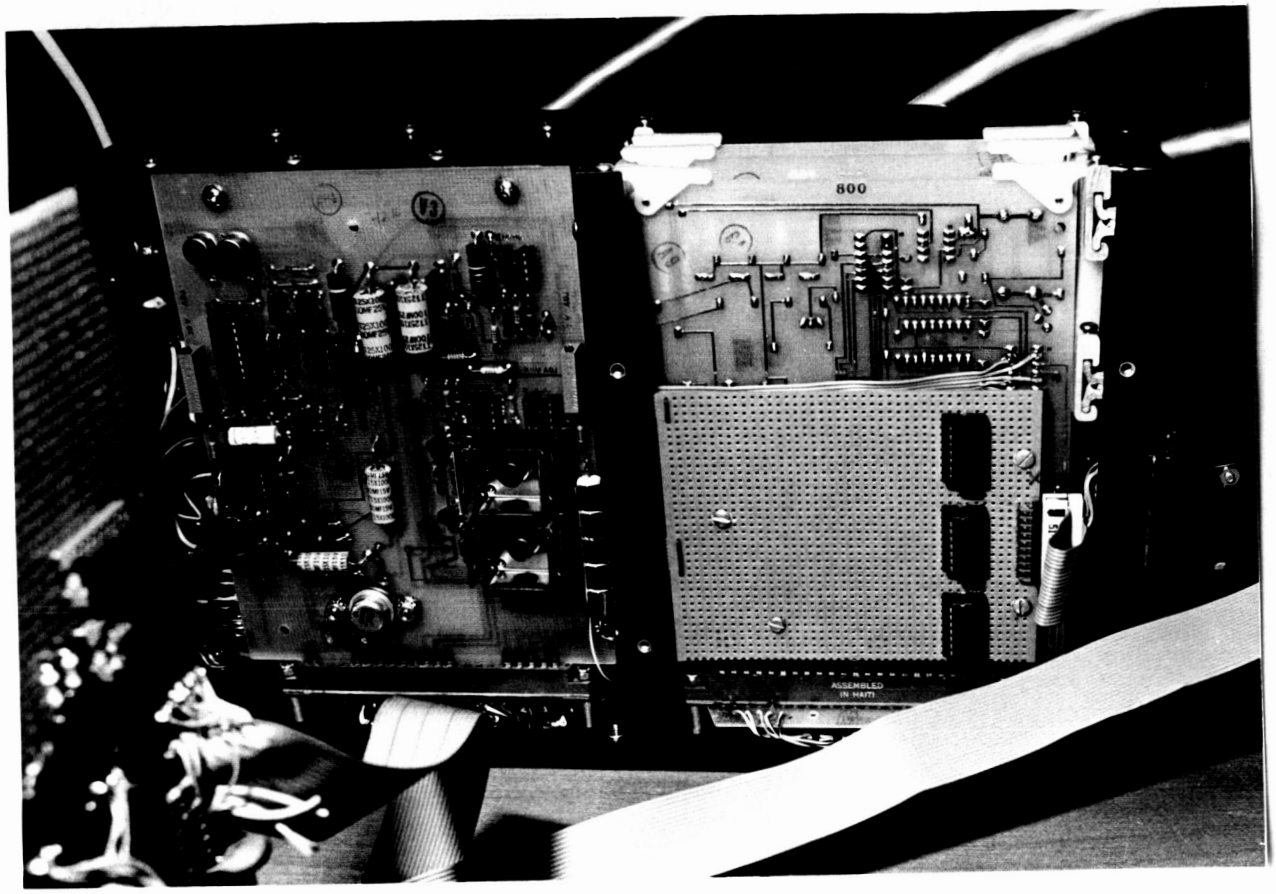
Close-Up of Uplink Processor Board, Receiver Portion.



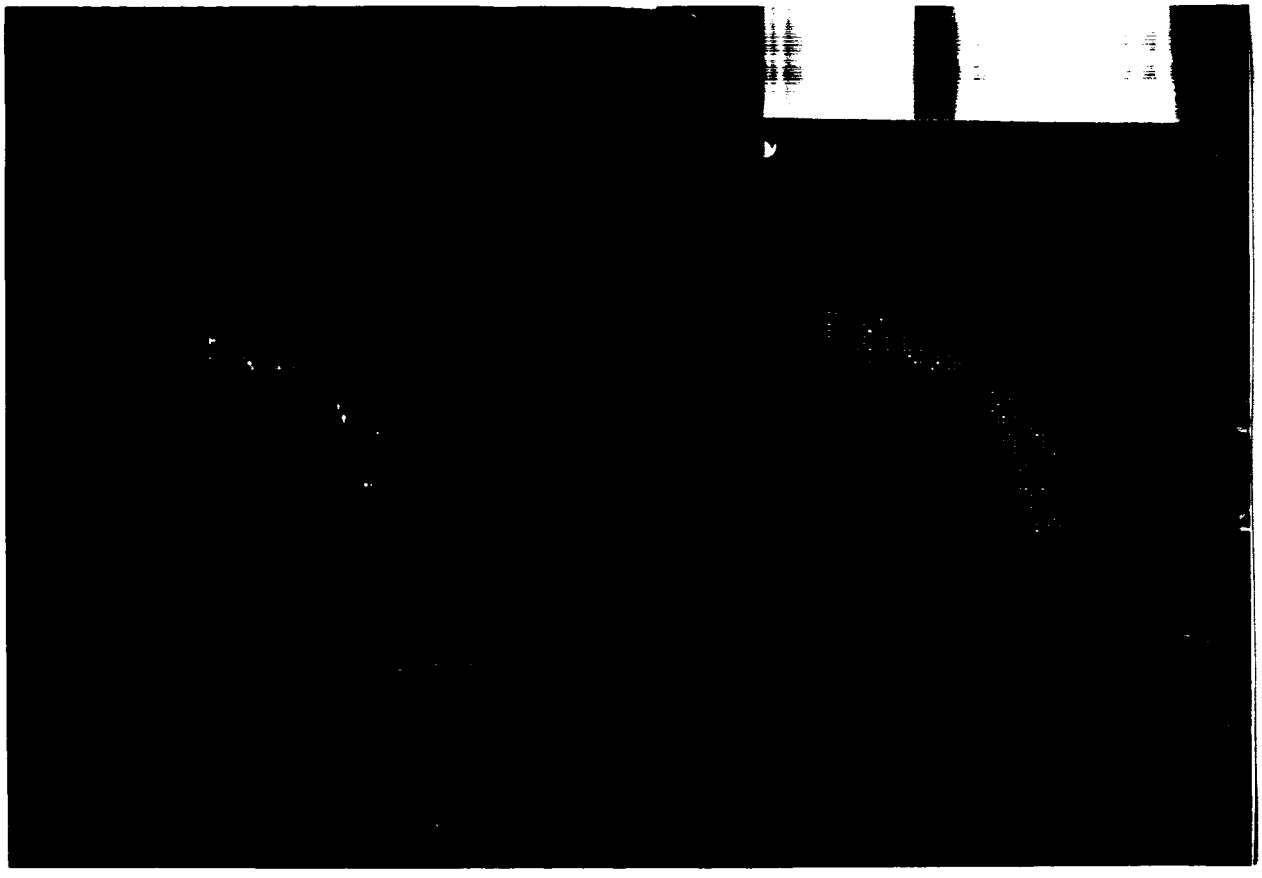
Uplink Processor Hardware, Receiver Portion, and Oscilloscope Display (The Gandalf Line Driver Was Replaced with a ComData Receiving Modem, VHF Receiver Not Shown).



Close-Up of Uplink Processor Board, Transmission Portion.



Close-Up of Stormscope Interface Board.



The Oscilloscope (Left) Displays the Transmitted Stormscope Data as Seen on the Stormscope Display (Right).