Microprocessor Programming
for Visual Data Display

By

David Horton

Honour's Thesis

Department of Physics
University of Queensland

1980

Table of Contents

CHAPTER 1

INTRODUCTION

The Bribie cross radar is an instrument for measuring the direction and doppler shifts of high frequency radio echoes from the ionosphere. It is complicated in operation and requires some skill on the behalf of the operator to reduce the costs incurred by magnetic tape for recording and computer analysis time, without causing a deterioration in the quality of the recorded data. Control of the system is by a microprocessor, with a visual display unit (VDU) portraying the data in real time to assist the operator in making the appropriate decisions and to check on the status of the system.

The purpose of this project was to produce a human recognisable display of useful operating data. Use of special symbols to convey amplitude and phase information has provided improvement over the old systems, a poor resolution display on a storage CRO and a VDU display limited to numerical information.

Further the display may form a basis for more complicated displays, while being a meaningful and useful monitor for the current procedure.

CHAPTER 2

THEORY

Elec waves

Consider an electromagnetic wave incident on a plasma. If the free electron density is high then the plasna will look like a conductor and hence the wave would not propagate in this plasma but rather reflect from it. At the other extreme is a low charge density, like air in a normal room, where all frequencies are able to propagate. An electromagnetic wave incident on a plasma of increasing electron density in the direction of the wave normal will be reflected when its frequency equals the so-called critical frequency of the plasma. The critical frequency, ignoring complicating effects introduced by the presence of a magnetic field or electron collisions, is the electron plasma frequency given by:

$$\omega_e^2 = N_e e^2 / M_e \varepsilon_o$$

where e=electronic charge, N_e=electron density, M_e=electron mass and ε_o=permitivity

N (e/cu. m)	f (Hz)
10^8	90K
10^9	280K
10-10	900K
10^11	2.8M
10^12	9.0M
10^13	28M

Table 2-1
Electron plasma frequency variation with electron density.

In the ionosphere we find a generally gradual variation in the electron density with height (see Fig 2-1). If we propagate a particular frequency towards the ionosphere, we will get a reflection from the region where the electron density reaches the critical value. Due to variations in the electron density horizontally the contours of equal electron density (isoionic contours) may be tilted to the horizontal. It is possible to show that an equivalent reflection surface similar in shape to the isoionic contours may be constructed (Whitehead PhD thesis). Now if the radio wave is in the form of a beam and is pointed in a direction so as to strike this reflecting surface at right-angles then we will receive an echo from that particular direction. From many measurements of this type it is possible to deduce the form of the reflecting surface.

The ionosphere is classified into regions depending on height and electron density. The lowest region, the D-region extends from a height of about 40kms to about 90kms. Electron densities are from about a maximium of 10^10 electrons per cubic metre down to about 10^8 e/cu m. In the height range 90kms to 160 kms is the middle (E) region with variations in electron density from 10^10 to 2x10^11 e/cu m. The upper levels (F-regions) have densities from 10^11 to 2x10^13 e/cu m and extend from 160kms to around 800kms where it becomes the magnetosphere (extending to several Earth radii). In the current experiment, the range of frequencies used are 2-6MHz, so ,from Table 2-1, the E and F regions are being investigated.

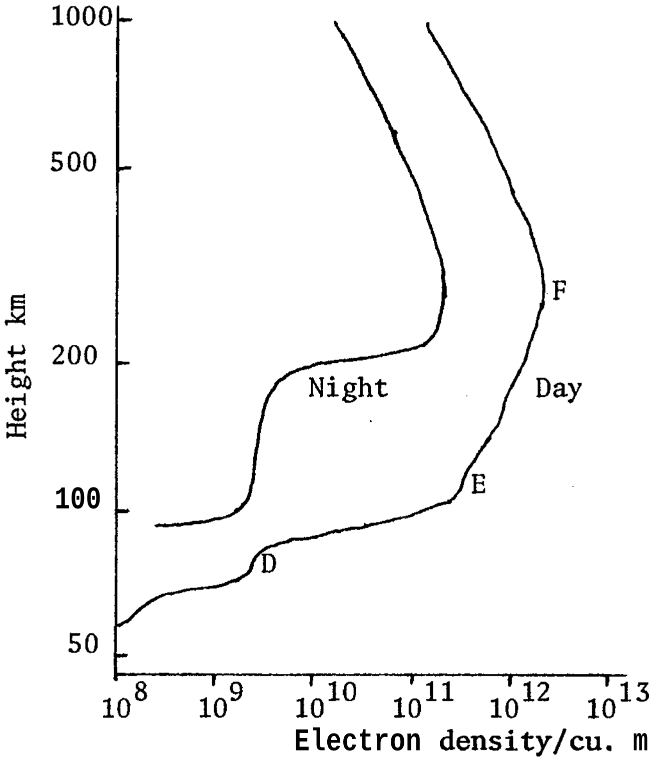
Neutral gas waves

In the ionosphere the majority of particles are neutral. Waves of various types may propagate in this neutral gas, eg sound waves. Any class of waves for which the major restoring force arises not from the pressure variation, but from the buoyancy forces under gravity are known as internal gravity waves. Their density and pressure amplitude is proportional to

exp (z/2H) sin (wt-K.R) where H is the scale height

The wave energy of internal gravity waves, if imperfectly reflected from the mesosphere (heights 50-80km) can lead to horizontal ducted waves in the ionosphere (Hines 1960).

The neutral gas waves have various effects on the ionisation. In the simplest situation, the neutral gas motions carry the ions and electrons with them, so that changes in the electron density reflect exactly the changes in air density. The Earth's magnetic field has a marked effect on the ionisation motion and in the F-region restricts it to motion along the field lines. Changes in air density may also effect changes in ionisation production. One way or another the ionisation distribution reflects the form of the internal gravity wave.



Ion isation profile for ionosphere

Fig 2-1

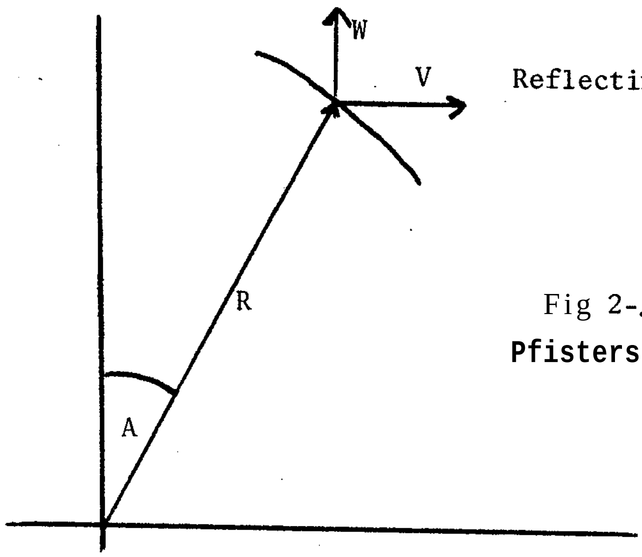


Fig 2-2
Pfisters Theorem

One important interaction occurs in the E-region. Associated winds can lead to vertical converging movement of ions and electrons giving rise to the formation of thin layers of ionisation (Whitehead 1960). Instabilities in the plasna also arise from gradients in fields passing through the plasna (gravitational, magnetic and electric). So it can be seen that reflections from the ionosphere need not appear simple.

ation of Ionisation Structure

To observe these structures, a useful theorem due to Pfister(1970) is described. Suppose we have a reflection point in a one dimensional reflecting surface in the ionosphere at a radial distance, R, at an angle, A, to the vertical moving with velocity V in the horizontal direction and vertically with velocity W (see Fig 2-2). It can be seen fairly easily that the rate of change of the radial distance or more generally phase range is given by:

$$dR/dt = V \sin A + W \cos A$$

This expression allows for the motion of the reflecting point along the reflecting surface. So if several measurements of dR/dt and A are taken, we can plot out the motion of the reflecting point, be it a wave, cloud or other irregularity.

Obviously the reflecting surface is not one dimensional. Generalising the above, we expand the angle A into its N-S component, N, and its E-W component, E, with velocity components v and w. It can be shown that in the small angle approximation $(1 >> A > N,E)$ that

$$dR/dt = Nv + Ew + W$$

This is a practical statement of Pfisters theorem given by Brownlie, Dryburgh and Whitehead (1973).

CHAPTER 3

EXPERIMENT AT BRIBIE ISLAND

In this chapter, instrumentation of the pencil beam radar on Bribie Island will be discussed. Transmitter and receiver aerials are arranged as two 10 element lines forming a cross, with the receivers oriented North–south. By transmitting and receiving at an angle to the vertical in two planes with a beam width of about 4 degrees, a small region of the sky is observed (see Fig 3-1).

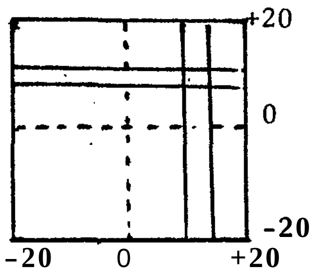


Fig 3-1
Beam Steering

The method of directing the beam in a particular direction is achieved by appropriate phase changes of the ten transmitters and of the ten local oscillators in the receivers. In turn, the phase changes are produced digitally by dividing down a higher frequency square wave. The twenty phase changes are read from a memory containing the digital beam direction (Brownlie 1973). Using six bit words for the beam control word, there can be a beam increment of 1/64 of the total angular range. Depending on the scan type used, the current system will look at either 10 or 11 directions in each plane yielding either 100 or 121 data sets for a scan. Vertical heights can be seen (Fig 3–2) to be determined from the travel time of the pulses ie.

H = c. travel time/2 (travel time of order 2mS)

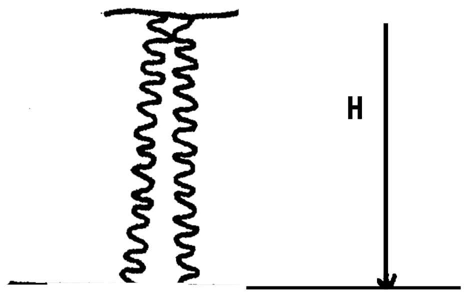


Fig 3-2
Height determination

H = *c* At /2

In order that echoes from both the E–region and F–region may be observed simultaneously, two identical recording systems operate with different receiver time delays and widths (gates 1 and 2). The maximium signal received during a gate is fed to an analogue to digital converter (ADC). Here the input is determined to be one of 64 (2^6) voltages derived from a reference voltage. Within a scan this reference voltage is constant, but if during the scan the maximium echo becomes too large or too small then the reference voltage is increased or decreased for the next scan. This is done using a digital to analogue converter (DAC) with a six bit number to specify the fraction of another reference voltage, which will be used as the reference for the next scan. This automatic reference voltage system (ARV) allows about 2 digit (Log 64 = 1.8) data to be collected over a three decade range (Log 4096=3.6), thus giving an expanded dynamic range.

The maximium echo received during the gate is recorded on magnetic tape and used to drive the VDU display. The transmitter radiates an approximately gaussian pulse 60 microseconds long. If two or more echoes are received within a gate, only the larger is recorded. However if the echoes are for different durations, first one then the others are recorded. Since the actual height of the peak echo is also recorded, subsequent data analysis can recover each echo separately. If this does not provide sufficient resolution, data sampling at intervals throughout the gate is possible, but this fast data recording is expensive in time and money and is only important for special recordings.

Suppose there is a reflecting point at some phase Range R, it is of interest when trying to follow the motions of the reflecting point to know how fast the range changes with time (see Pfisters theorem). If we define the phase (Ph) of the return RF signal with respect to an internal RF signal (see Fig 3-3), then the rate of change of range is related to the rate of phase by:

$$dR/dt = L(RF)/4\pi \; dPh/dt$$

where L(RF) is the wavelength of the RF being used (of order 100m)

In practice dPh/dt is determined by measuring Ph in consecutive scans separated by time dt which is of order 1 second. Two components of the return pulse are recorded, those in phase with the internal RF signal and those 90 degrees out of phase with it (quadrature component). Unless the phase changes by more than π, corresponding to the reflecting point moving more than a quarter of the RF wavelength, then dPh/dt can be approximated unambiguously. These two components can then be transformed to polar coordinates giving the amplitude and phase of the reflection.

It is too complicated, with the available microprocessor power, to calculate the rates of change of phase of the echoes which might be present. This must be estimated by the operator. The purpose of **this** project was to provide the operator with a display which would allow him to make this judgement.

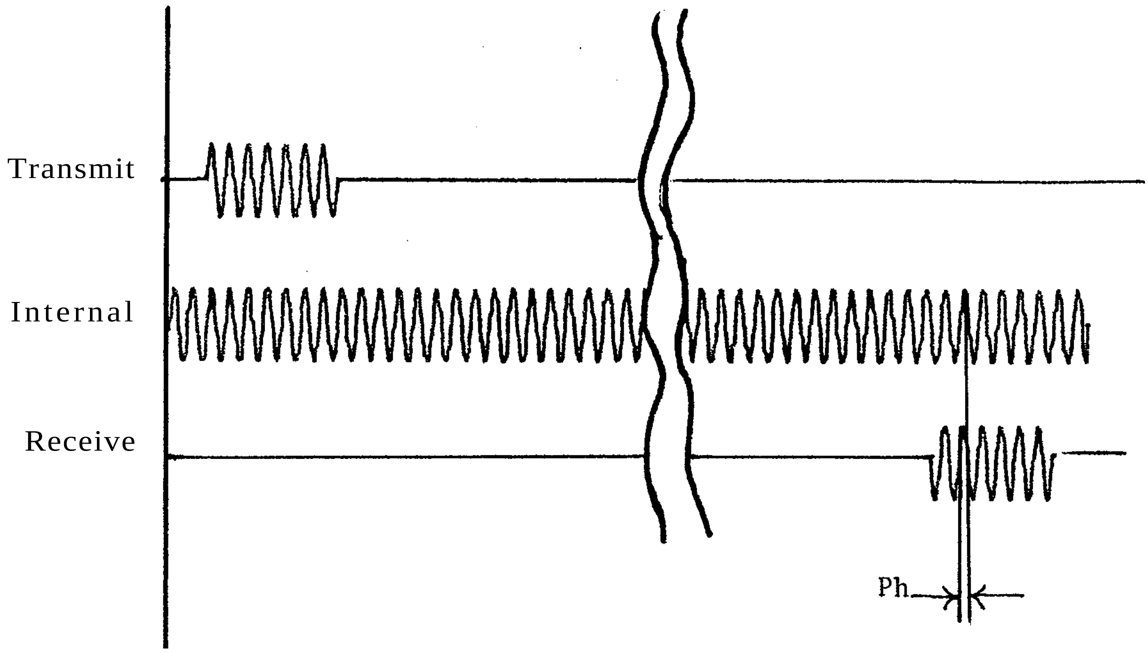


Fig 3-3

Phase difference between transmit and receive pulse

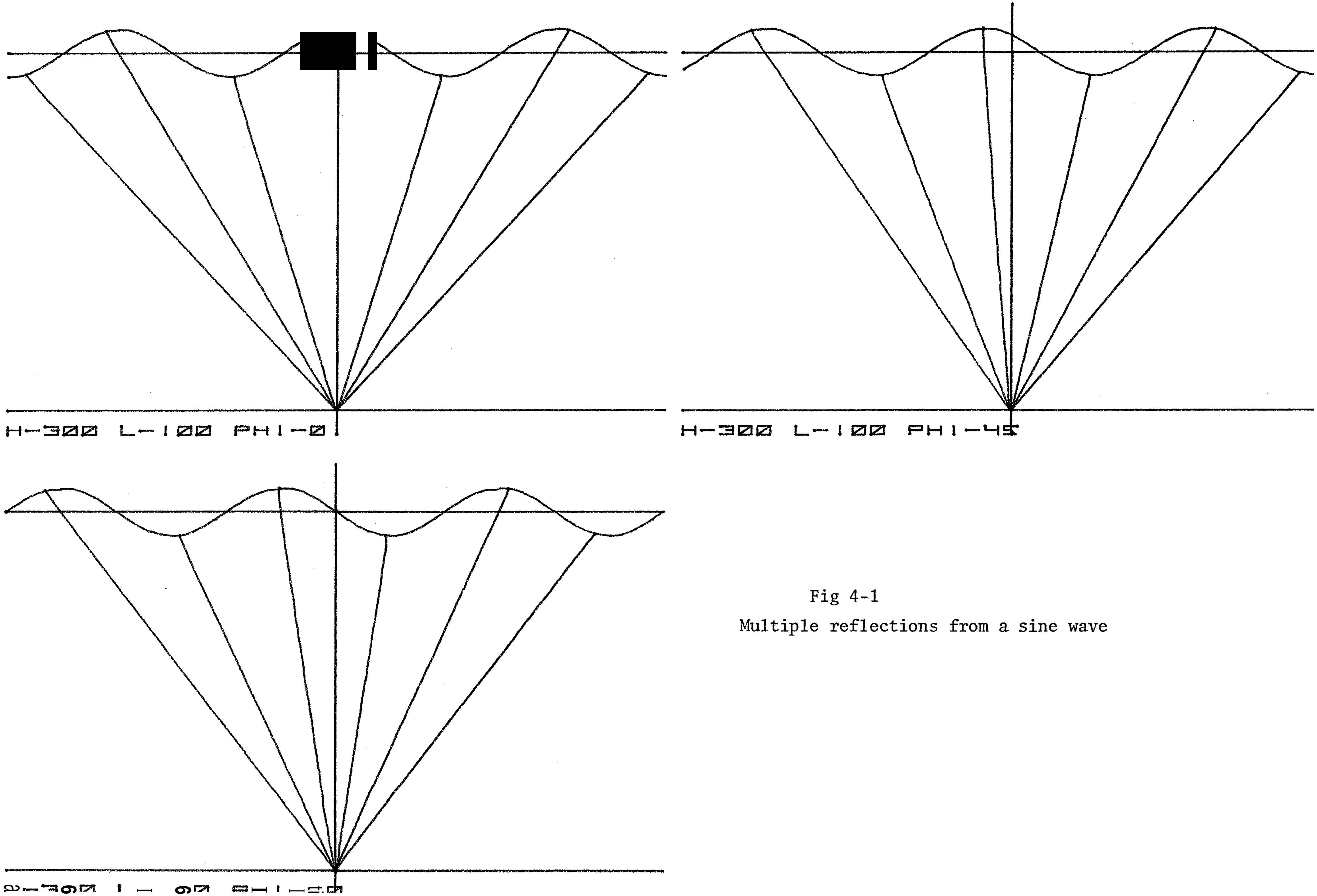


Fig 4-1

Multiple reflections from a sine wave

CHAPTER 4

DATA: INTERPRETATION AND REPRESENTATION

It was shown in the previous chapter that the rate of change of phase range could be monitored by knowledge of the phase of the reflection. From Fourier analysis, it can be shown that for a sinusoidal signal, sampling to avoid ambiguity must be done faster than a rate giving half wavelength changes between scans. In practice however, the slowest scan rate possible gives maximium efficiency in data storage and processing. With the VDU display an ambiguous phase change will appear as a half rotation of the phasor arrow. So with this phase information, and amplitude information, the operator can increase the sampling rate by decreasing the time between scan or changing to a mode where several scans are taken in rapid succession followed by a longer wait. This gives accurate measurements of dR/dt at more discrete times, but doesn't produce the huge volwne of data that continuous scanning at this rate would produce.

Reflections from the reflecting surface are not as simple as from a flat plane directly overhead. Complications can arise from wavelike structures passing overhead causing the reflecting surface to be tilted and hence the reflection appearing off-vertical. The presence of thin clouds of high ionisation (sporadic-E) which reflect the beam also give rise to reflection points which may be off-vertical. Multiple reflections also may arise from other sections of a wavelike structure as shown in Fig 4-1. An added feature that can occur if the the second gate is looking at a high altitude is a second hop reflection, resulting from an intermediate reflection from the ground as shown in Fig 4-2.

Reflection points appear on the VDU display as clump of larger amplitude arrows which stand out from the smaller amplitude noise. There is almost always such a collection in the centre of the display corresponding to a reflecting point almost directly overhead. An earlier VDU display which displayed a matrix of one or two digit numbers gave this impression when the second digit appeared. Having noticed a (new) reflection point, the phase information allows the sampling rate to be adjusted. In addition, signal strength in the gates is represented by displaying the automatic reference voltages for each gate.

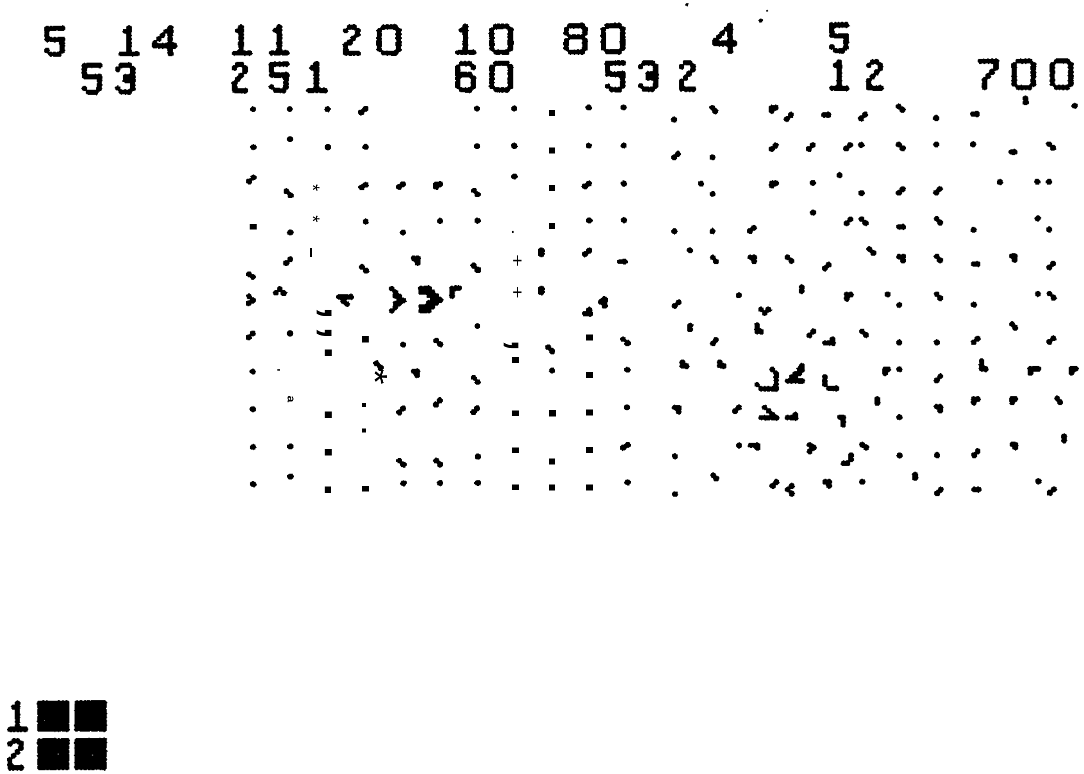


Fig 4-3(a) VDU display
showing off-centre reflection point

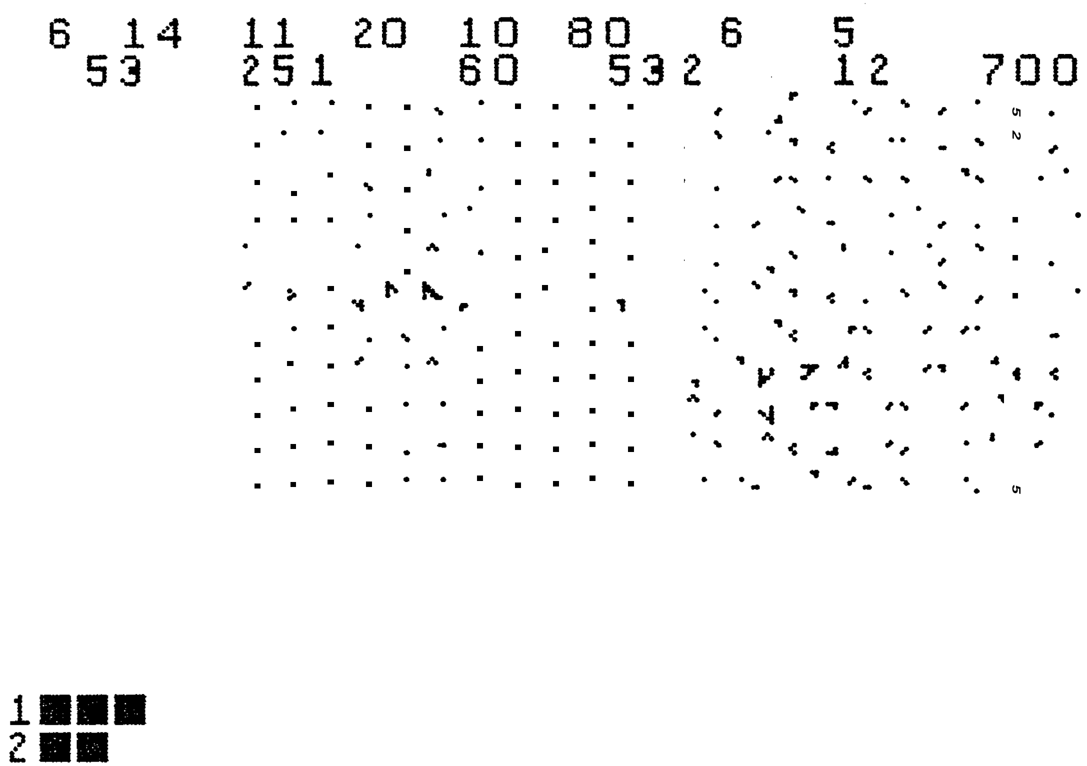


Fig 4-3 VDU display
showing changes in phase and ARVs

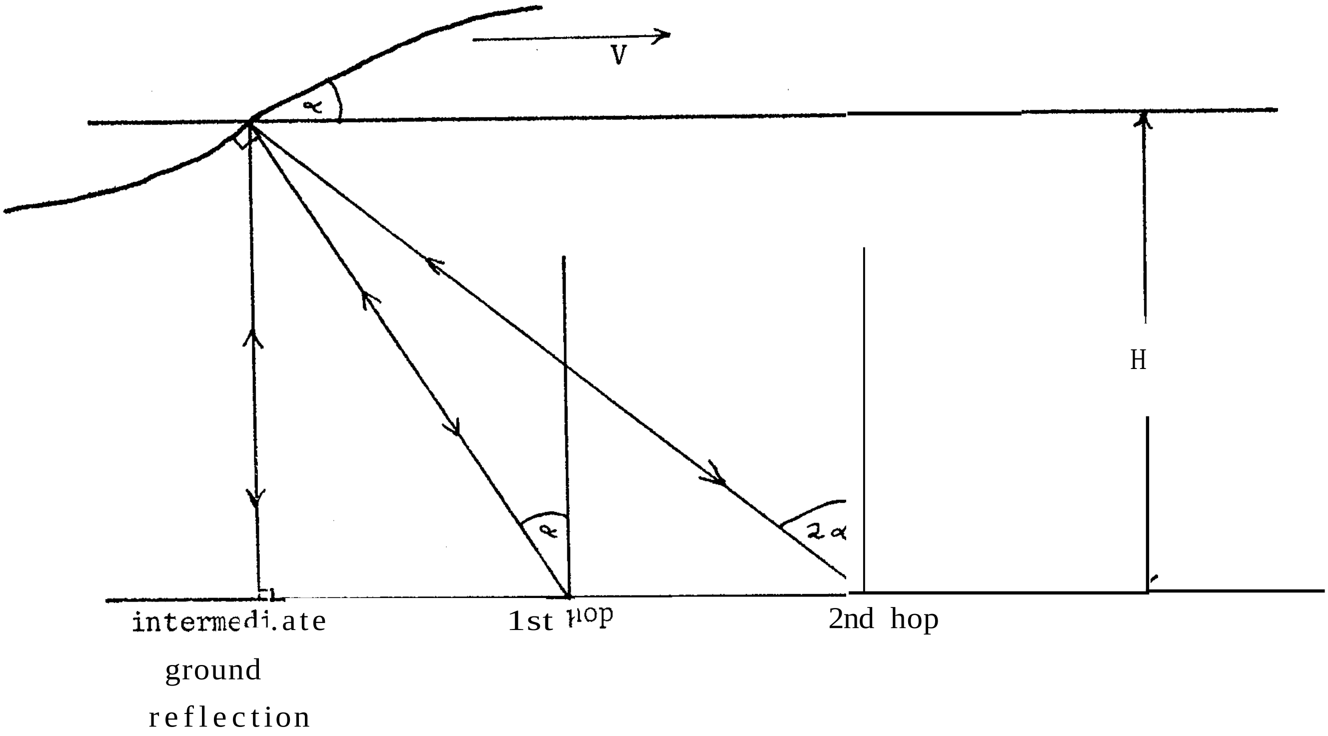


Fig 4-2 First and second hop reflections

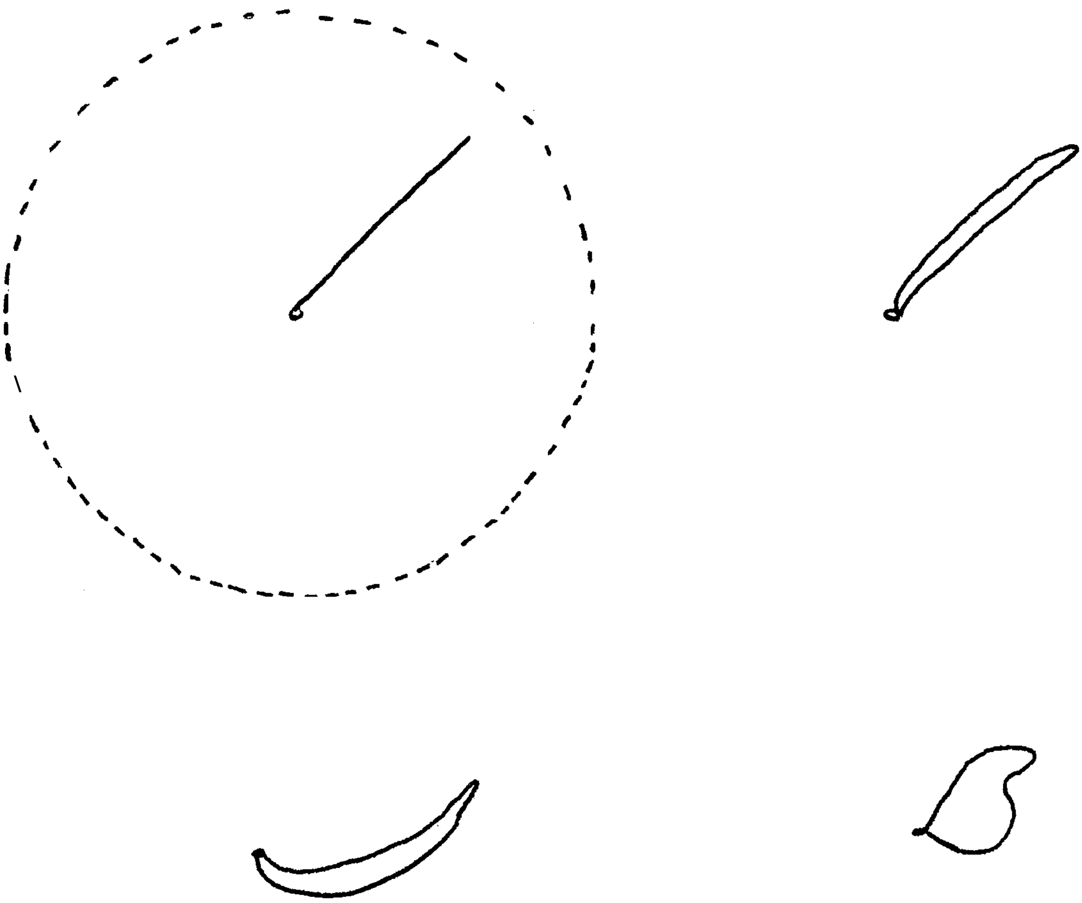


Fig 4-4 Various shapes of phasors

An analogue CRO display of the two phase components is available in the system and provides exceptional resolution of the inphase-quadrature component relationship. (This is more complicated than the straight line assumed in the VDU display). fie various shapes of the phasor on the oscilloscope (Fig 4-4) can provide additional information. Since the beam is steered very rapidly (greater than 100 directions per second), to display all the information would require a matrix of oscilloscopes and would be redundant anyway since the operator could not absorb the information. There is however a period between the complete scans of the sky when the beam is pointed in one direction while the last scan is processed. With operator control the beam is pointed at an interesting structure and the shape of the phasor interpreted. To enable the beam to be directed towards a structure deemed interesting by the VDU display, a mechanism for displaying the current interscan beam position on the VDU was developed. fie mechanism used was dependent on a feature of the MC6845 CRT controller (see chap 5), which was the ability to place a flashing cursor at any character position on the screen. fie cursor used was a thin line in the centre of a character, which was made to flash over the position of the nearest phasor display arrow of gate one. with this combination of phasor display and cursor the'beam can be accurately directed to the structure.

Numerical information that is also of use, like the starting position and widths of the gates (1,2 and the enveloping main gate) are written on the screen along with the time,date and the numerical settings of the ARVs. Other information is recorded and not displayed *(eg* the heights of the reflection), but possible options are discussed in chapter 7.

The most useful feature of the VDU display is that the rate of change of phase between samples can be determined, allowing the sampling rate to be set to an optimium without fear of aliasing. It also provides an immediate indication of major echoes and the absence of significant echoes. The old numerical VDU display could display the same information, but could not be read so easily. (Remember that there is typically only one second before the display changes).

CHAPTER 5

PRODUCTION OF CHARACTERS

As the M6800 microprocessor series are 8-bit byte machines it is easiest to allow peripherals to use 8 bits or less (a normal ASCII character requires 7 for example). Using 8 bits we can represent up to 256 characters (2^8). It was then decided that to represent both amplitude and phase information in a character set a resolution of 16 directions and 15 amplitudes would be a good compromise. The 16 remaining characters are allotted to book-keeping purposes (the digits 0–9, space, block (for use in a level meter display) , question mark as an error and asterix, hyphen and full stop (see appendix 1)).

To make the video display program run as quickly as possible the character set was ordered in such a way that the four lower bits specified the 16 angles (0=E, 3=NNE etc) increasing in 22.5 degree steps to 15 (= 1111 binary). Similarly amplitudes are increased from 0 to 14 in the higher bits. (See appendix 1, where low order bits are plotted on X-axis and high order bits on Y-axis). Using the three basis directions, (N,NE,NNE), the complete set of 16 directions can be obtained by reflection as shown in Table 5–1.

Using the transformations in Table 5–1 we can reduce the number of characters to be made from 256 to 61 (45+16).

In order to speed up the choice of the final character set and to reduce the probability of errors (with 256x64 dots to consider) and to correct them when they do occur, several programs to manipulate arrow characters have been written for the department's PDP-11. Listings of these can be found in appendix three. Facilities provided are character inputing, editing, representation on terminal or plotter, coding described above and conversion to and from binary form for transfers to ROM.

OBJECTIVE	BASIS	AXIS 1	AXIS 2
ENE	NNE	NE	null
NE	NE	null	null
NNE	NNE	null	null
N	N	null	null
NNW	NNE	N	null
NW	NE	N	null
WNW	NNE	NE	N
W	N	NW	null
WSW	NNE	NW	null
SW	NE	NW	null
SSW	NNE	E	N
S	N	E	null
SSE	NNE	E	null
SE	NE	E	null
ESE	NNE	NE	E
E	N	NE	null

Table 5-1

To input characters one line of 0's and 1's is typed for each line of the character e.g. 00001000 indicates the first line of the amplitude zero north arrow. characters are input either one by one in the editor or all at once in the general main program. The only non-obvious feature about this program is that 16 North then 16 North-East then 16 Nor-Nor-East arrows must be entered so that the coding program knows where each is.

Editing is performed in whole characters only and used in conjunction with the terminal display routine allows reflection in **axes,** copying swapping and inputing of characters. To allow the reduction algorithm, an automated version of the editor was written. Plotting can be done in two ways, either character by character or automatically from a "plot" file which has a particular format. *An* example of #is is shown in Appendix 1, where the plot file was also automatically generated.

Since the read only memory was to be used in the video display circuit, a transfer had to be made from the PDP11 to Dr. Hainsworths' M6800 which has an EPROM programmer as a peripheral. Because the characters were stored internally as ASCII encoded binary numbers (by fortran) it was trivial to create a proper binary file which could be copied onto paper tape for the transfer.

In all, this collection of programs provided a useful tool for developing the characters without becoming too unwieldy to use, as can be seen by the inclusion of several character plots.

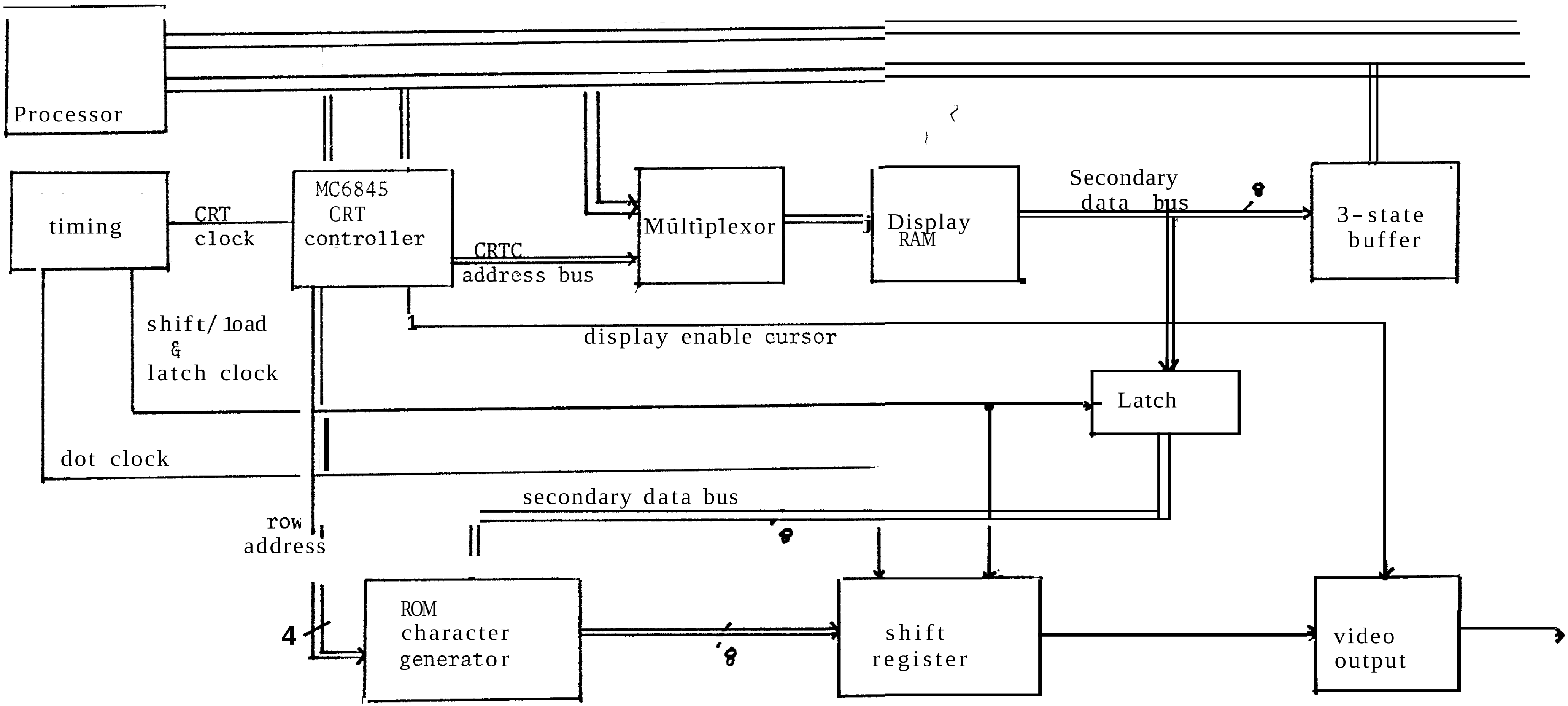


Fig 5-1 Block diagram for video display

Characters displayed on the screen are handled by a memory map system centered around a MOTOROLA MC6845 CRT controller. A block diagram is shown in fig 5-1. The circuit has been modified to allow square 8x8 characters and 1K memory etc. . The video output of fig 1 is mixed with t horizontal and vertical sync pulses from the CRTC to provide a signal for the TV's video amplifier.

To the microprocessor the memory map consists of the control-register (pointer) and the data register of the CRTC itself and 1K of random access memory. This 1K of memory appears as normal memory to the processor because the CRTC and processor addresses are muliplexed with the processor given priority.

In operation each line of points on the screen is accessed once per screen refresh, so if we take a particular line, the points are accessed in order shown in fig 5-2. i.e. the order the dots are clocked out of the shift register to the video output is along the top line of first character then top line of second till last character on line then the second lines of each char on line. Every dot that is input to the video output (see fig1) has been clocked out of the 8-bit shift register (plus the space as shown in Fig-2) by the dot clock which currently runs at 8.3 MHz. Each row which is loaded into the shift register is specified by the particular character and the particular row of the character. This specifies one of the 256 (no of characters) X 8(lines per character) bytes in the character generator ROM. Row addressing is handled directly by the CRTC. The character to be output is selected on the CRTC address bus (not necessarily the same as seen by the processor) and the byte from the display RAM is directed to and held by the one byte latch. The full circuit description and the chip description of the CRTC can be found in the references.

One of the advantages of using the 6845 system is that most of the features are controlled by software eg. number of characters on a line, the number of lines and control for the hardware cursor which is described later. A fairly general interactive program has been written which produces a table of 6845 control parameters suitable for inclusion in a Motorola program. TABLE is written in Fortran and transports between the three operating systems used (RT-11 (Bribie), Tops-10 and RSX-11) with only a change to the system dependant file specification. A listing and sample output is found in Appendix 2.

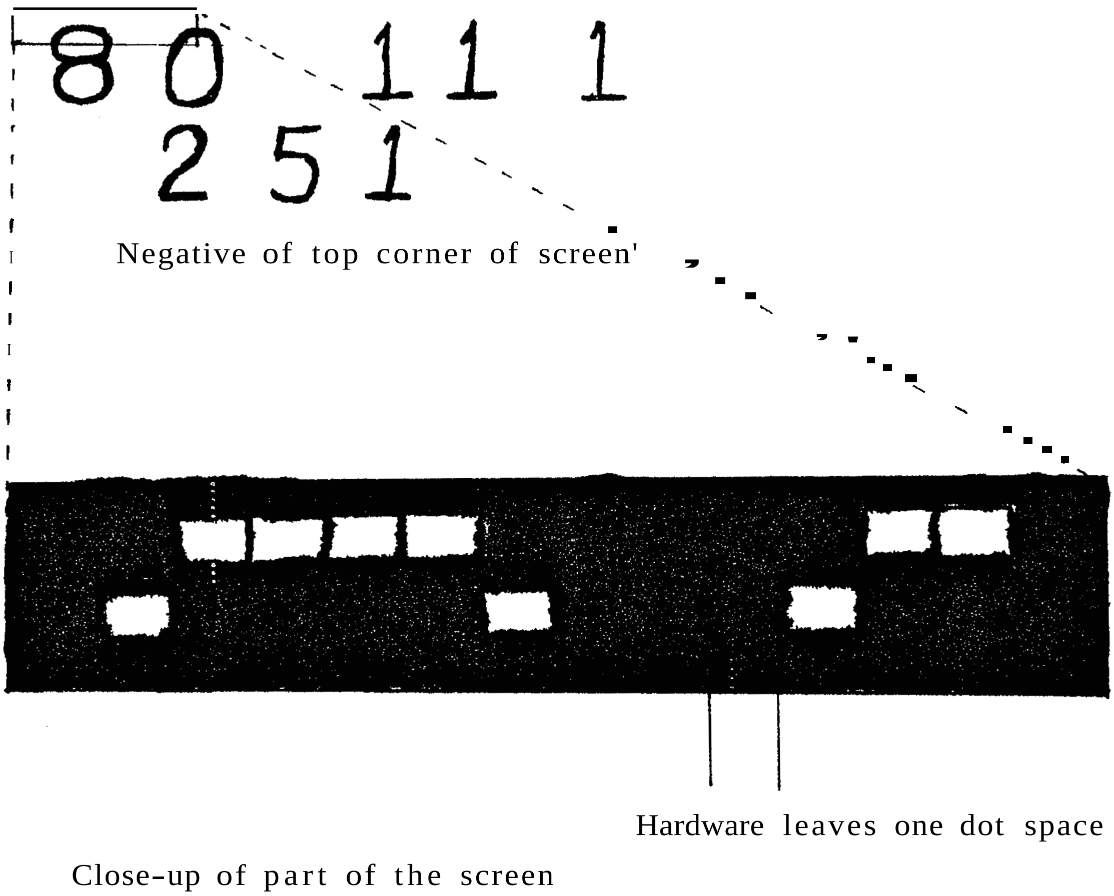


Fig **5-2** Clocking dots onto screen

CHAPTER 6

DECODING, DISPLAY AND ACCURACY

At the present the VDU display has three sections as shown in fig 4-3. The top two lines are a decimal number display of the date-time and the automatic reference voltage settings on the top line and the gate(height) settings on the second. These are refreshed every cycle, but usually only the top line changes. In the following lines is the phasor display. Currently this involves two matrices (10x10 or 11x11) matrices of arrows along-side one another, representing the phase and amplitude as seen by the two gates. (The 10x10 matrix is currently being expanded to a software switchable 11x11 or 16x16 matrix or a combination of the above . This allows significant improvements to a sinc interpolation for intermediate angles, used in later analysis). Near the bottom of the screen is a pair of bar graphs showing the ARV amplitudes. There is another feature that will write a message on the bottom of the screen, but is only used in a fatal error case at present and is limited anyway to arrow set characters.

I will now describe these displays in a little more detail. Data displayed on the screen will be written to tape for data analysis. After the beam control programs have run and the data collected is stored for the transfer to digital tape recorder, MMSYS is called (see App-2). MMSYS reads and interprets this stored data and displays a summary on the screen.

Firstly the top two lines give mainly control information and certain dial settings. Here the numbers are read from the top of the data block and decoded from various formats and output using routines which convert hexadecimal numbers to BCD numbers and which print BCD numbers. Most of this section was written for the earlier numerical display and little change was required.

Simple bar graphs are used to display the automatic reference voltages which have settings from 0-63. Since the screen width is currently only 44 characters wide, the resolution was reduced to 0-31. Each step corresponds to one "block" character across the screen. Each gate is identified by a "1" or a "2" in the first character position on the line. This display gives an indication whether a generally strong or weak signal reflection is received during the gate. The ARV is determined by the maximium signal strength during the previous scan, that is when the beam is pointing in the direction of the major echo.

The dominating feature of the display is the 2 matrices of arrows. Each of the 100 or 121 positions corresponds to a particular setting of the North-South and East-West angles determined by the transmitter and receiver phases. Each arrow represents the phase and amplitude relative to an internal standard of the reflected pulse from the ionosphere. After the data pair of in-phase and quadrature components are selected and some centering of the display performed, the arrow character is chosen, output and in the process moving across the screen one position. One line is output at a time from each gate alternately and so we zig-zag through the data while scanning across the screen. (Gate one data is separated from gate two data by about 500 memory locations). So we output 10 or 11 arrows from gate one (starting lower left) then a space, then 10 or 11 arrows from gate two progressing up the screen. Decoding the characters from the in-phase and quadrature components involves finding both the amplitude and phase of the phasor (first checking that the data is legal).

Amplitude is calculated using the approximation A= X + Y/2 if X > Y or A= Y + X/2 if Y > X. This approximation is quite fast to calculate and quite accurate (see table 6-2). Unfortunately the calculation of the phase is not as simple, but since we need only a resolution of one in sixteen, we can get very good results using multiplication of the smaller component by 2, 3/2 and 4 only (all fairly fast to program). The partitioning can be achieved by testing the data to be in the ranges shown in Table 6-1.

Required angle	Actual angle	Tangent	Test done
11.25 degrees	14.0 deg	1/4	X-4Y
33.8	33.7	1/1.5	X- Y+Y/2
45	45	1/1	X-Y

Table 6-1. calculation of angle

By five comparisons it is possible to narrow down the angle to the quadrant and an angle range within the quadrant. (These are 0 to 14 degrees, 14 to 34, 34 to 56, 56 to 76 or 76 to 90 with accuracy shown in table 6-2). In the process the comparisons required for the amplitude calculations are also done. Individual amplitudes are 6 bits long giving a possible range of 0-63, however only 0-57 are used, providing a check on the system. With the encoding algorithm used (X+Y/2), the final amplitude will be in the range 0-85 (= 57+57/2 rounded down). Having found the amplitude, we then divide the 86 possible results into 15 ranges for representation in the character. (Currently these are 0-1-4-10-16-22-28-34-40-46-52-58-64-70-76-85 but are easily changed).

For the amplitude approximation
average error = 0
mean absolute error = 4 from max of 85
standard deviation 5 from max of 85
maximium relative error = 0.3

For the phase choice approximation
No of choices not nearest desired direction = 0
mean absolute difference = 5 degrees

These figures are for the possible range of input data only.

Table 6–2 Errors associated with MMSYS algorithms

If any data value is out of the legal range a "?" is output instead. Possible reasons for this are (1) the program samples the data after it has been sent and been corrupted (as can occur in 200Hz scan mode) (2) another section of code may not have written the data in the expected format (eg when no height information is sent) (3) the operator is using the diagnostic unit and is sending different information on the bus, (4) various hardware error or bus noise that may have crept in. Thus the presence of illegal characters indicates a fault in the system.

In an earlier chapter, the interscan beam position was described. The implimentation of this and positioning of the matrix centred on two tables. One was a set of tables dividing the 64 beam positions into 10, 11 or 16 for display on the 10x10 etc matrix. The second was a table containing the addresses of the starts of each line. These tables allow different screen formats to be implimented.

CHAPTER 7

FUTURE DEVELOPMENTS OF THE VDU DISPLAY

One major lacking of the current display is that not all of the relevant information that is recorded is presented to the operator. Heights of the reflections are also stored. It is possible that within a gate two echoes of comparable intensity are received and the recording hardware may flicker between them. If this can be detected then it would allow the operator to decide whether or not to use the fast scan mode. Two simple implimentations of displaying the height are to either complicate the current display by superimposing on the present display the height information or to have a separate display for the heights. One way to superimpose the height information would be to code the heights as colors. To use a modification of a conventional television under the control of the MC6845, rather than a special color graphics unit, would require, in principle, little change to be made to the system. Six colors plus black and white can be achieved by switching on and off the three guns in the colour TV set acting as the MU. As far as the processor is concerned the interface controlling the guns would just be another interface. Six colors would probably be all that the display could use without being to complicated for the operator to interpret. It does however limit the height resolution to about six values. An alternative display is to separately display the height information in a similar manner to the phasor display. For this purpose "arrow" characters would probably not be suitable, so further characters would have to be added. A simple set of additional characters is shown in fig 7-1, where the amplitude resolution has been decreased by one. The problem with using a separate display for the heights is that the current screen is not big enough, so either more displays are needed or a larger display (with appropriate hardware changes to keep the characters square).

An annoying feature of the cursor display is that the position is only updated when the whole screen is changed. If the scanning is slow then the time lag between manually moving the beam to its display may be excessive. This would require only minor changes in the programming of the processor if it were dedicated to controlling the display, but the processor is used as the system controller and hence is not available.

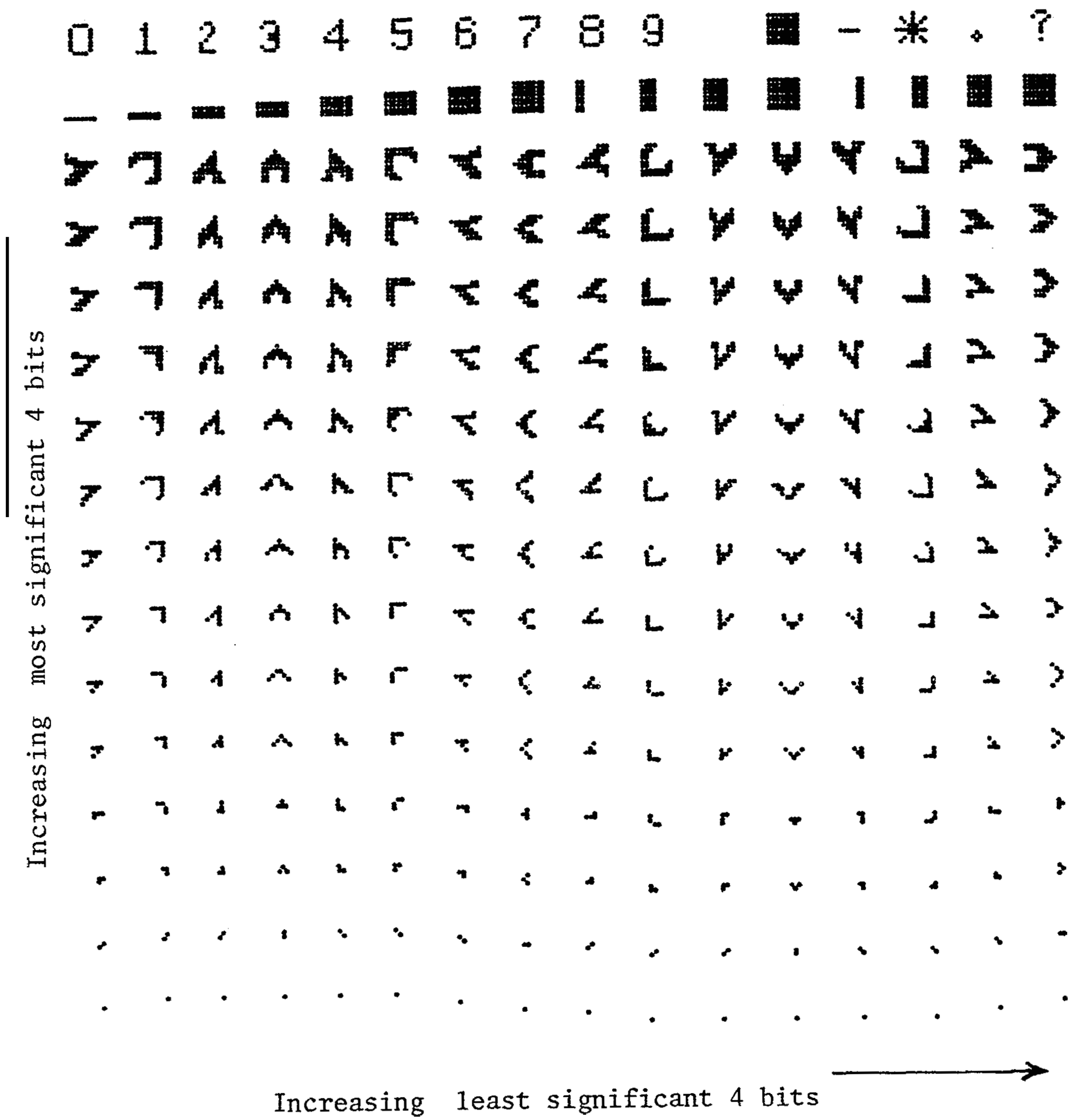


Fig 7-1

"Height" characters

Since a major use of the display is to allow the operator to monitor phase changes between scans, it would be desirable if the differences could be displayed. This would require all the components from the previous scan to be stored and the differences taken. This in itself is trivial to code, but there is a major complication in that the reference voltages for the gates changes at almost every scan. When the project was started the processor being used was a Motorola M6800 on which the required multiplications of data with the ARVs would have been very time consuming, but now a later model M6809 processor is being used which does have a (simple) hardware multiply instruction. This may make the process feasible. A logical extension of this difference display is the control of the scan rate. In a very simple scheme #is would involve measurement of the phase changes of the phasors with amplitudes greater than a certain noise limit and making intelligent choices of scan rate if #is change became too large or too small over a few scans.

Another feature which would be desirable is a scan of the amplitude against time for the main gate. This combined with a display of the gate position would allow the operator to tell when when a reflecting point was about to move outside the measurement gate and to correct appropriately. At present this display is made on an oscilloscope with gates appearing as more intense traces by use of the z-axis. By use of the above "height" characters, both amplitude and gate positions could be shown (using both the horizontal and vertical varying characters).

One unfortunate problem with the large scale monitoring of the system is the time required and time lags between measurement and display. The current form is preferable to finding bad data days later during processing, but the monitoring could grow to become a real time data analysis which was not the intention of the use of a microprocessor. Perhaps as the whole system becomes more complicated (in terms of control), the time required for display may become critical and a processor dedicated to the task of monitoring may become desirable.

Acknowledgements

I would like to thank Professor J D Whitehead who supervised this project. Also I am grateful to Mr. W R (Fred) From for his assistance throughout this project, Mr. G Cooper for his help with the PDP11, Dr Hainsworth and Mr. R Dennis for their help with the microprocessors and the office staff in the preparation of this report.

References

Brownlie, Digital phase shifter performance. Electronics Letters, May (1973) vol 9 no 10.

Brownlie, Dryburgh and Whitehead, Measurement of the velocity of waves in the ionosphere: A comparison of the ray theory approach and diffraction theory. J. Atmos. Terr Phys (1973), vol 35 pp 2147–2162

Brownlie, PhD thesis, University of Queensland (1974)

Dryburgh, PhD thesis, University of Queensland (1974)

Goosard and Hooke, Waves in the Ionosphere, Elservier 1975.

Hines, Internal atmospheric waves at ionospheric heights. Can. J. Phys. , no 38 pp 1441–1481.

Hines,Paghis,Hartz and Fejer, Physics of Earths Upper Atmosphere, Prentice-Hall 1965.

Motorola, The complete microcomputer data library (1978).

Pfister, The wavelike nature of inhomogeneties in the E-region, J. atmos. Terr. Phys. (1971) vol 33

Simplify video-display design by using versitile IC controller. Electronic Design ,no 13 June 1979.

Whitehead, The formation of the sporadic-E layer in temperate zones. J. Atmos. Terr. Phys. (1961) vol 28 pp 49–58

Appendices

Appendix 1

The screen

The Arrow character set

Appendix 2 Bribie program listings

(1) MMSYS (display control)

(2) VDUSUB (general support)

(3) TABLE (CRTC table generator)

(4) sample run and output

Appendix 3 Arrow program listings

(1) MAIN (general)

(2) (subroutine) INCSET (input character set)

(3) (sub) PLTLN (plots a line of dots)

(4) (sub) DISPLN (displays a line on terminal)

(5) (sub) SETCHR (reads character set from disk)

(6) CHANGE (arrow editor)

(7) SETALL (automatic editor)

(8) PLTALL (performs plots from a plot file)

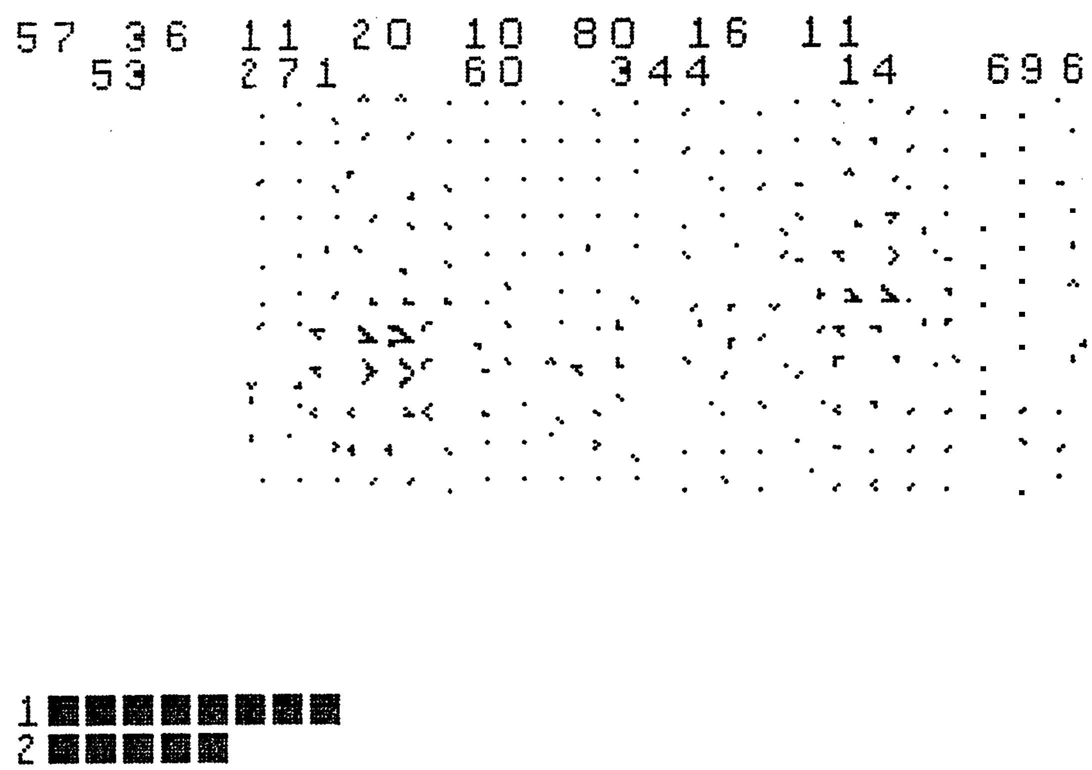
(9) BIGBIN (converts ASCII to binary)

(10) INVERT (inverts each character in set)

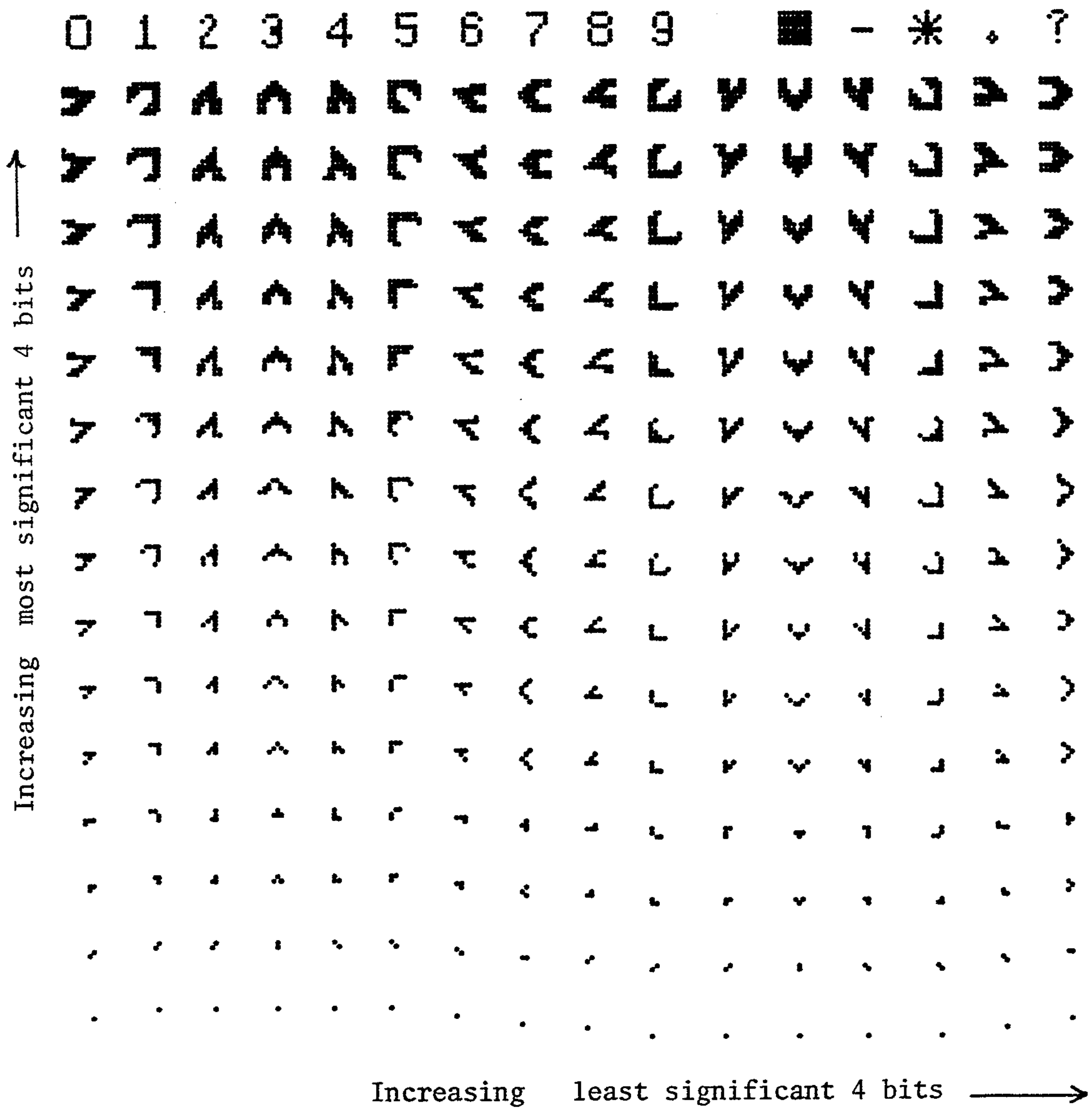
(11) (macro) BYTWRD (copies word to byte)

(12) (macro) REVERS (reverses bits in a byte)

(13) MAK44S (produces PLOT file to simulate screen)



VDU display



"Arrow" character set

```
        NAM MMSYS
* version 16-Oct-80
* THIS ROUTINE OUTPUTS THE RASTER PATTERN SOUTH TO THE BOTTOM
* WEST TO THE RIGHT.
* THIS REQURIES THE MEMORY MAPPED VDU HARDWARE TO BE INITIALIZED.
* G2Y $300 , G2X $0400, G1Y $0500, G1X $0600
* G1HT $0700 , G2HT $0800 , THE CHARACTER SUMMARY $0200.
UTACK   EQU $EB00
DEVICES EQU $E000
TP0     EQU $EB02       ;Temporaries
TP1     EQU $EB04
TP2     EQU $EB06
TP3     EQU $EB08
TP4     EQU $EB0A
TP5     EQU $EB0C
TP6     EQU $EB0D
CRTC    EQU $E010       ;CRT controller registers
G1S     EQU $0500       ;Gate 1 start
G2S     EQU $0300       ;Gate 2 start
XCORD   EQU $B8 ;TXB
YCORD   EQU $B9 ;RXB
MT      EQU $77 ;(RECEIVE)SCAN SIZES
NT      EQU $76 ;(TRANSMIT)
HTFLAG  EQU $EB0F
STOB    EQU $6200       ;Reset
HEXBCD  EQU $8400
* THE BOOK-KEEPING IS DECODED AND PRINTED FIRST.
* THE PROGRAM JUMPS BACK TO THE ARRAY CONTROL PROGRAM AT $A200
* THIS ROUTINE REQUIRES THE ROUTINE HEXBCD TO RUN .
* HEXBCD CONVERTS 2 WORDS TO BCD.
* FIRSTLY THE BOOK-KEEPING IS OUTPUT.
MMEM    EQU DEVICES+$400        ;1K of memory map memory
MMPNT   EQU $EB76       ;Pointer to current position in memory map
MMPNTL  EQU $EB77       ;pointer to low byte of above
BOTTIM  EQU $0FF3       ;bottom of data summary
DATTOP  EQU $0FFF       ;top of "
RESTK   EQU $8010       ;organises data
*
VORG    EQU $860D
DEMOUT  EQU VORG+$80    ;write no. to MM screen (no leading 0)
TOPSCN  EQU VORG+$CA    ;go to top of MM screen
CLRSCN  EQU VORG+$D1    ;clear MM screen
DEMOTZ  EQU VORG+$A3    ; write no. to MM screen (leading 0)
OUTCHM  EQU VORG+$9A    ;write char in "A" to MM screen
MMUPLN  EQU VORG+$DF    ;go up one line
BMESOT  EQU VORG+$A7    ;write message along bottom of screen
MMLINE  EQU VORG+$F5    ;GO TO NEXT LINE OR TOP
LNTBL   EQU $87B4       ;table of MM line starts
LN2     EQU LNTBL+4
LN12    EQU LNTBL+24
LN13    EQU LNTBL+26
LN14    EQU LNTBL+28
LN15    EQU LNTBL+30
LN16    EQU LNTBL+32
LN18    EQU LNTBL+36
LN19    EQU LNTBL+38
LN20    EQU LNTBL+40
LN21    EQU LNTBL+42
```

```
LSTLIN  EQU LNTBL+46
STP10   EQU 7           ;COLUMN WHERE THE ARRAY DISPLAY STARTS
STP11   EQU 6
STP16   EQU 2
*******************************
*
* CHARACTER CALCULATOR SCRATCH
PHASE   EQU $EB52
AMPL    EQU $EB53               ; THIS +^ INFO. HELD IN ARROW CHAR.
CURCHR  EQU $EB54               ; THE CURRENT CHAR.
INPHAS  EQU $EB55               ; THE INPHASE COMPONENT
QUADR   EQU $EB56               ; THE QUADRATURE COMP.
QUAD    EQU $EB57               ;QUADRANT OF THE PHASOR
XTEMP   EQU $EB58               ; TEMPORARY FOR INDEX $ 18&19
STPOS   EQU $EB5A       ;NUMBER OF LEADING BLANKS
SCRHT   EQU $EB5B       ;SCREEN HEIGHT
SCRWTH  EQU $EB5C       ;SCREEN WIDTH
LSTSIZ  EQU $EB5D       ;A CHECK FOR LAST SCAN SIZE
*
* ORDINARY CHARACTERS IN ARROW SET ($F?)
ZERO    EQU 240         ; "O"
SPACE   EQU 250
BLOCK   EQU 251         ; EVERY DOT IN MATRIX SET
HYPHEN  EQU 252
ASTERX  EQU 253         ; "*"
FSTOP   EQU 254         ; "."
QSTMRK  EQU 255         ; "?"
*
        ORG $7800
MOTRST  LDU #UTACK
        LDA MT  ;GET HEIGHT & WIDTH AND
        INCA    ;CONVERT TO COUNT FROM 1 RATHER THAN 0
        STA SCRHT
        LDA NT
        INCA
        STA SCRWTH
        ADDA SCRHT      ;CHECK IF SCAN SIZE CHANGED
        CMPA LSTSIZ
        BEQ SAMSIZ
        STA LSTSIZ
        LBSR CLRSCN
SAMSIZ  LBSR CURSE      ;SET CURSOR LOC
        LBSR PARVS      ;DISPLAY ARVS
        LBSR TOPSCN
        LDX #DATTOP+1
        PSHU X
        LDA #6  ; 12 BYTES OF TIME AND DATE
        STA TPO
NXTIM PULU X
        LDA ,-X
        LDB ,-X
        PSHU X
        ASLA
        ASLA
        ASLA
        ASLA
        PSHU B  ;ORA B
        ORA ,U+
```

```
        LBSR DEMOUT        ;output time to screen
        INC MMPNTL         ;blank
        DEC TPO
        BNE NXTIM
* NOW OUTPUT ARV'S.
        PULU X
        LEAX -2,X          ;OUTPUT ARV1 FIRST.
        PSHU X
        LDB ,X
        CLRA
        LBSR HEXBCD        ;convert
        TFR B,A
        LBSR DEMOUT
        INC MMPNTL
        LDX ,U
        LDB 1,X
        CLRA
        LBSR HEXBCD
        TFR B,A
        LBSR DEMOUT
        INC MMPNTL
        LBSR MMLINE        ;GO TO NEXT LINE
*********************
* NOW OUTPUT THE GATES.
        PULU X
        LEAX -12,X
        PSHU X
        LBSR GTPOUT                ;OUTPUT G1 POSITION FIRST.
        LBSR OUTGT
        STA TP1            ;TP1 IS USED TO STORE THE G1 WIDTH TILL THE HEIGHTS A
        TFR A,B
        CLRA
        LBSR WDTYP
* NOTE THE OUTPUT IS IN 4 CH. DECIMAL.
        LBSR GTPOUT        ;cascades into WDTYP
        LBSR OUTGT
        STA TP2
        TFR A,B
        CLRA
        LBSR WDTYP
        LBSR OUTGT
        TFR A,B
        CLRA
        ASLB
        ROLA
        LBSR WDTYP
        LBSR OUTGT
        TFR A,B
        CLRA
        ASLB
        ROLA
        ASLB
        ROLA
        LBSR WDTYP
        BRA HTCHCK
GTPOUT  PULU X
        LDA 1,X
        LDB #0
```

```
        LSRA
        RORB
        LSRA
        RORB
        ORB ,X++          ;move on
        LSRA
        RORB
        LSRA
        RORB
        PSHU X
WDTYP LBSR HEXBCD
        PSHS B
        LBSR DEMOUT              ;NO LEADING ZERO
        PULS A
        LBSR DEMOTZ              ;LEADING ZERO
        INC MMPNTL
        RTS
OUTGT PULU X
        LDA 1,X
        RORA
        RORA
        RORA
        ANDA #$C0
        ORA ,X++          ;move on
        PSHU X
        RTS
BYTYP LBSR DEMOUT
        INC MMPNTL
        RTS
HTCHCK LBSR RESTK       ; THIS ORGANISES THE DATA
        TST HTFLAG
        BEQ DATAOT
***** CHECK FOR HEIGHT ERRORS AND PRINT OUT IF THERE ARE ANY
        CLR TP6
        LDX #$0700
        LDA TP1
        LSRA
COMP1 CMPA ,X+
        BGE INCR
        INC TP6
INCR    CMPX #$0764
        BCS COMP1
        LDX #$0800
        LDA TP2
        LSRA              ;HEIGHTS ARE DIVIDED BY 2.
COMP2 CMPA ,X+
        BGE INCR2
        INC TP6
INCR2   CMPX #$0864
        BCS COMP2
        LDB TP6
        BEQ CLRNOW
        CLRA
        LBSR WDTYP
        BRA DATAOT
CLRNOW LDA #SPACE
        LBSR OUTCHM
        LBSR OUTCHM
```

```
        LBSR OUTCHM
        LBSR OUTCHM
**************************************************
* NOW THE DATA IS OUTPUT NORTH TO THE TOP.
DATAOT  LDA SCRHT       ;FIND LOWER LEFT CORNER
        LDX LN12,PCR
        CMPA #10        ;?10 LINES
        BLE GSCRLN      ;GOT SCREEN LINE THEN
        LDX LN13,PCR
        CMPA #11        ;?11 LINES
        BLE GSCRLN
        LDX LN16,PCR    ;ASSUME 16 THEN
GSCRLN  STX MMPNT
        LDX #G2S
        PSHS X
DONW    LDA SCRHT       ;NO. LINES/MATRIX
        ASLA
        STA TP1
LINE LDA SCRHT  ;NO. OF COLUMNS/MATRIX
        STA TP2
        PULS X
        CMPX #G1S
        BLT GT1
        CMPX #G1S+$100
        BLT GT2
        LBRA ERROR
**********************
* BOTH GATES  ARE OUTPUT SIDE BY SIDE.
GT2 LEAX -$200,X        ;GATE 2 DATA
        NEGA    ;SCREEN WIDTH STILL IN ACC A
        LEAX A,X        ;GET THIS ROWS DATA NOT THE NEXT ROWS
        NEGA    ;RESTORE A
        PSHS X  ;parameter to PNT
        LDA #SPACE      ;A SPACE BETWEEN MATRICES
        LBSR OUTCHM
        BRA PNT
GT1 LEAX $200,X
        PSHS X  ;parameter to PNT
        LBSR MMUPLN     ;NEXT LINE UPWARDS.
        LDB STPOS       ;OUTPUT A FEW SPACES AT THE START OF THE LINE
        LDA #SPACE
        LDX MMPNT
HEADER  STA ,X+ ;save calling OUTCHM many times in a loop by copying code
        DECB
        BGT HEADER
        STX MMPNT
PNT     PULS X
        LDA ,X+         ; LOAD UP IN PHASE COMP.
        SUBA #$40               ; % 1000000 CONVERT TO 2'S COMP.
        STA INPHAS              ; SAVE
        LDA $FF,X               ; LOAD QUADRATURE COMP. & CONVERT
        PSHS X
        SUBA #$40
        STA QUADR
* FROM ABOVE INFORMATION FIND CORRECT
* CHARACTER CODE FOR AMPLITUDE AND PHASE (0-14,0-15)
*
* FIRST FIND QUADRANT
```

```
        TST INPHAS
        BLT XNEG                ; IF INPHASE NEG. THEN BRANCH
        CLRA                ; QUADRANT FIRST ENCODED 0,-8,8,-16 LATER FIX
        TST QUADR               ; INITIALLY ASSUME QUADR >0 => QUADRANT 1
        BGE ENDQAD              ; QUADRATURE +VE THEN FINISH QUAD. SEARCH
        LDA #$F0                ; QUAD 4 CODE IS -16
        NEG QUADR               ; MAKE POSITIVE FOR REST OF CODE
        BRA ENDQAD              ; FINISHED SEARCH
XNEG NEG INPHAS             ; ALSO SET POSITIVE
        LDA #$F8                ; (-8) INITIALLY ASSUME QUAD +VE =>QUAD 3
        TST QUADR
        BGE ENDQAD              ; ASSUMPTION CORRECT
        LDA #$8
        NEG QUADR               ; ALSO SET POSITIVE
ENDQAD STA QUAD             ; STORE ENCRIPTED QUADRANT
* NOW FIND ANGLE IN FIRST QUADRANT
* CODINGS POSSIBLE 0 TO 14 (0), 14 TO 34 (1), 34 TO 56 (2)
* 56 TO 76 (3), 76 TO 90 (4)
* TRY MIDDLE FIRST
        LDA #$2
        STA PHASE
* BEFORE FURTHER PROCESSING TEST THAT DATA LEGAL
* PREVIOUS ROUTINE HAS SET DATA POSITIVE SO COMPARISON EASY
        LDA QUADR
        CMPA #$39               ; COMPARE AGAINST MAX LEGAL VALUE
        BGT DATERR              ; IF GREATER JUMP TO HANDLER
        LDA INPHAS              ; NOW COMPARE INPHAS COMPONENT
        CMPA #$39               ;
        BGT DATERR              ; ERROR HANDLER WILL PRINT ? INSTEAD
        TFR A,B             ; HAVE INPHAS COMPONENT IN A. SAVE IT IN B
        CMPA QUADR              ; SEE IF GREATER OR LESS THAN 45 DEGREES
        BGE XGTY                ; BRANCH TO INPHAS > QUADR
* QUADR > INPHASE INPHAS IN A , AMPLITIUDE = QUADR + INPHAS/2
        ASRA                ; INPHAS/2
        ADDA QUADR
        STA AMPL                ; SAVE AMPLITUDE
* CALCULATE 1.5 INPHAS TO TEST 45 DEG TO 56 DEG RANGE
        TFR B,A             ; INPHAS SAVED IN ABOVE
        ASRA                ; INPHASE /2
        ADDA INPHAS             ; 1.5 INPHAS
        CMPA QUADR              ; COMPARE 1.5 INPHAS <=> QUADR
        BGE NDPHASE             ;45 TO 56 DEG. RANGE PHASE = 2 (ASSUMED)
* CALCULATE 4X QUADRAT. FOR 56 TO 76 DEG RANGE
        INC PHASE               ; NEW PHASE 3
        ASLB                ; COPY NO LONGER NEEDED
        ASLB                ; MULTIPLY BY 4
        CMPB QUADR              ; COMPARE 4X INPHASE <=> QUADRAT.
        BGE NDPHAS
        INC PHASE               ; HENCE PHASE IS 4
        BRA NDPHAS
* INPHASE > QUAD. AMPLITUDE = INPHAS+ QUADRAT./2
XGTY LDA QUADR              ; LOAD UP QUAD. COMP.
        TFR A,B             ; SAVE COPY IN B
        ASRA                ; QUADRAT./2
        ADDA INPHAS             ; 1.5 QUADR.
        STA AMPL                ; SAVE AMPLITUDE
* CALCULATE 1.5 QUAD. FOR 34-45 DEG RANGE.
```

```
        TFR B,A                 ; RETRIEVE COPY OF QUAD.
        ASRA                    ; QUAD/2
        ADDA QUADR                      ; 1.5 QUAD.
        CMPA INPHAS
        BGE NDPHAS                      ; 34-45 DEG RANGE PHASE =2
        DEC PHASE
        ASLB                    ; 4 QUAD FOR 14-34 DEG RANGE
        ASLB
        CMPB INPHAS                     ; COMPARE 4 QUAD. <=> INPHASE COMP
        BGE NDPHAS                      ; PHASE =1
        DEC PHASE                       ; HENCE PHASE =0
        BRA NDPHAS
*
* ERROR HANDLER
DATERR LDA #QSTMRK                      ; '? ILLEGAL DATA OUTPUT
        STA CURCHR
        BRA OUTCC                       ; OUTPUT CURRENT CHAR.
*
* GENERATE FULL ANGLE
NDPHAS LDA QUAD                 ; LOAD UP ENCODED QUADRANT
        ADDA PHASE                      ; ADD ANGLE IN QUAD. 1
        BGE NOADJ                       ; TEST IF ENCODED FORM IS NEG.
        NEGA
NOADJ ANDA #$F                  ;% 1111 DECODE 0-15
        STA PHASE
*
* FROM EARLIER WE HAD ILLEGAL DATA IF > 57
* SO MAX AMPLITUDE IS 57 + 57/2 =85
* WE NOW MAP THIS TO 0-14
        LDA AMPL                        ; LOAD UP AMPLITUDE
        CLRB                    ; TOP HALF OF B IS AMPLIT. CODING
        CMPA #$1                        ; INITIALLY ASSUME SMALL AMPL. <=> 0
        BLE NDAMP
        INCB
        CMPA #$4                        ;^D4 =>1
        BLE NDAMP
        INCB
        CMPA #$C                        ;^D10 =>2
        BLE NDAMP
        INCB
        CMPA #$10                       ;^D16 =>3
        BLE NDAMP
        INCB
        CMPA #$16                       ;^D22 =>4
        BLE NDAMP
        INCB
        CMPA #$1C                       ;^D28
        BLE NDAMP
        INCB
        CMPA #$22                       ;^D34
        BLE NDAMP
        INCB
        CMPA #$28                       ;^D40
        BLE NDAMP
        INCB
        CMPA #$2E                       ;^D46 =>8
        BLE NDAMP
        INCB
```

```
        CMPA #$34               ;^D52
        BLE NDAMP
        INCB
        CMPA #$3A               ;^D58
        BLE NDAMP
        INCB
        CMPA #$40               ;^D64
        BLE NDAMP
        INCB
        CMPA #$46               ;^D70
        BLE NDAMP
        INCB
        CMPA #$4C               ;^D76 =>13
        BLE NDAMP
        INCB
        CMPA #$55               ;^D85 =>14 LAST STEP A BIT BIGGER
        BLE NDAMP
* IF WE FELL THRU. ERROR CASE
* LDX #ERROR MESSAGE
* LBSR WRITE MESSAGE TO SCREEN
*
        LDB #$F         ; WE WILL OUTPUT A '? INSTEAD
        STB PHASE               ; OVERWRITE PHASE INFO.
*               ; AND FALL INTO ORDINARY EXIT
NDAMP ASLB              ; PUT L.S. HALF TO MOST SIG. HALF
        ASLB            ; SINCE AMPL. IN HIGH HALF AND PHASE IN LOW
        ASLB
        ASLB
        ORB PHASE               ; INCLUDE THE PHASE INFO.
        STB CURCHR
OUTCC   LDA CURCHR       ; NOW OUTPUT THE ARROW
        LBSR OUTCHM
        DEC TP2
        LBGT PNT
        DEC TP1
        LBGT LINE
FIN PULS X
        LBRA STOB
ERRMS1 FCB QSTMRK,QSTMRK,QSTMRK,QSTMRK,HYPHEN,ZERO          ;????-0
        FCB 4
ERROR PSHU X
        LEAX ERRMS1,PCR
        LBSR BMESOT
        JMP $F000               ; RESET
*
* TABLE OF MAPS OF AMPLITS. TO DISPLAY ARRAY FOR CURSOR
*   the unusual placement is due to bus in 6809 assembler
TBL10   FCB 0,0,0,0,1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,8,9,9,9,9,9
*33 LINES IN THE TABLE
* TABLE for 11x11 matrix
TBL11   FCB 0,0,1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6
        FCB 7,7,7,8,8,8,9,9,9,10,10,10,10          ;+SENTRY
*
* TABLE for 16x16 matrix
TBL16   FCB 0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10
        FCB 10,11,11,12,12,13,13,14,14,15,15,15 ;+SENTRY
*
*
```

```
* CALCULATE CURSOR LOC
CURSE   PSHS X,A,B         ;SAVE REGS
        LDB #$45           ;BLINK 1/16, CURSOR START 5(%01000101)
        LDA #10            ;CURSOR START REG
        STA CRTC
        STB CRTC+1
        LDA #11            ;CURSOR END REG.
        STA CRTC
        STB CRTC+1
        LDA YCORD          ;RANGE 0-255
        LSRA               ;0-31
        LSRA
        LSRA
        LDB SCRHT          ;GET HEIGHT OF SCREEN
        LEAX TBL10,PCR     ;GET TABLE OF DISPLAY POSITIONS
        CMPB #10           ;?10 LINES
        BLE GOTTAB         ;GOT CORRECT TABLE
        LEAX TBL11,PCR
        CMPB #11           ;?11 LINES
        BLE GOTTAB
        LEAX TBL16,PCR     ;ASSUME 16 THEN
GOTTAB  LDA A,X ;FIND POSITION IN DISPLAY
        SUBA SCRHT         ;CONVERT YCORD UP-SCREEN TO DOWNSCREEN
        INCA
        NEGA
        LEAX LN2,PCR       ;POINT TO LINE 3(FROM LN0)
* (0,1 ARRAY DATA, 2 ARROW)
        ASLA    ;each element of LNTBL is 2 bytes
        LDX A,X            ;MOVE DOWN Y LINES
        PSHS X             ;SAVE FOR A MOMENT
*
        LDB XCORD          ;0-255 ACROSS SCREEN
        LSRB    ;0-31
        LSRB
        LSRB
        LDA SCRWTH         ;GET SCREEN WIDTH
* NOW AS ABOVE TO FIND THE HORIZONTAL POSITION OF CURSOR
        LEAX TBL10,PCR     ;GET POS ON ARRAY
        CMPA #10           ;?10 COLUMNS/MATRIX
        BLE GOTAB2         ;GOT CORRECT TABLE
        LEAX TBL11,PCR
        CMPA #11           ;?11 COLS.
        BLE GOTAB2
        LEAX TBL16,PCR     ;16 THEN
GOTAB2  LDB B,X
        PULS X             ;RETRIEVE POSITION
        ABX                ;ADD B TO X, MOVE ACROSS SCREEN
        LEAX -MMEM,X       ;CURSOR STARTS @0 NOT @MMEM
        LDA SCRWTH         ; NO COLS/MATRIX
        LDB #STP10         ;TRY 10 FIRST
        CMPA #10
        BLE GOTHED         ;GOT LENGTH OF HEADER
        LDB #STP11         ;?11
        CMPA #11
        BLE GOTHED
        LDB #STP16         ;ASSUME 16 THEN
GOTHED  STB STPOS          ;TELL REST OF PROGRAM
        LEAX B,X           ;DISPLAY IS SHIFTED BY STPOS
```

```
*                               ;ALSO DISPLAY IS SHIFTED ACROSS
        TFR X,D                 ;COPY TO A+B REG
*CURSOR LOCS HIGH REG14,LOW REG15  ,BELOW OVERWRITES OTHER OF REG
        LDB #14                 ;SET HIGH HALF
        STB CRTC
        STA CRTC+1
        TFR X,D                 ;GET ANOTHER COPY
        LDA #15                 ;SET LOW HALF
        STA CRTC
        STB CRTC+1
        PULS X,A,B              ;RESET REGS
        RTS
*
*       displays magnitudes of arvs
PARVS   PSHS X,A,B              ;SAVE REGS
        LDX MMPNT               ;SAVE CURRENT CHAR. POSITION
        PSHS X
        LDX LN19,PCR            ;line past end of arrows
        STX MMPNT
        LDB DATTOP-13           ;GET ARV1
        LSRB
        LDA #ZERO+1             ;"1"
        BSR PBLKS               ;print B block chars
        LDB DATTOP-12           ;get ARV2
        LDX LN20,PCR
        STX MMPNT
        LSRB
        LDA #ZERO+2             ;"2"
        BSR PBLKS
        PULS X                  ;retrieve char pos.
        STX MMPNT
        PULS X,A,B
        RTS
*
*       prints B block chars then pads with spaces till col 22
PBLKS   PSHS B                  ;save for padding
        LBSR OUTCHM             ;output char "1" or "2"
        LDA #BLOCK
        TSTB
PBLK1   BLE PBLK1E              ;IF <=0 END THIS PRINT BLOCK
        LBSR OUTCHM
        DECB
        BRA PBLK1               ;MORE BLOCKS
PBLK1E  PULS B
        SUBB #40
        LDA #SPACE
PBLK2   LBSR OUTCHM
        INCB
        BLT PBLK2
        RTS
*
        END
```

```
        NAM VDUSUB
*       Version 18-SEP-80
VORG    EQU $860D
* THIS ROUTINE DOES THE OUTPUT TO THE VDU OR TTY FOR
* MOST STONDARD TYPES.
XOUTCH  EQU $F018
PDSSBY  EQU $F03E
CRTC    EQU $E010
MMPNT   EQU $EB76
MMPNTL  EQU $EB77
MMEM    EQU $E400
MMSIG   EQU $E4
*****************
SPACE   EQU 250
ZERO    EQU 240
NCW     EQU 44  ;NO. C IARS DISPLAYED ACROSS SCREEN
*****************
        ORG VORG
* THIS ROUTINE OUTPUTS 2 CHAR BCD DATA LEADING ZEROES SUPPRESED.
DECOUT PSHS     A
        BSR OUTHH
        PULS    A
        BSR OUTHL
        LDA #$20 SPACE IN ASCII
        BSR OUTC1
        RTS
OUTHH LSRA
        LSRA
        LSRA
        LSRA
        BNE ONW
        LDA #$20
        BRA OUTC1
OUTHL ANDA #$0F
ONW ADDA #$30
OUTC1 JSR XOUTCH
        RTS
* SIGNED HEX OUT WITH LEADING ZEROES SUPPRESSED
SHEXOT TSTA
        BMI NEGOT
        PSHS    A
        BITA #$F0
        BEQ NOZERO
        LDA #$20
        BSR OUTC1
        PULS A
        JSR PDSSBY
        RTS
NOZERO LDA #$20
        BSR OUTC1
        LDA #$20
        BSR OUTC1
        PULS    A
HALFOT ANDA #$F
        ADDA #$30
        CMPA #$39
        BLS *+4
        ADDA #7
```

```
        BSR OUTC1
        RTS
NEGOT NEGA
        PSHS    A
        BITA #$F0
        BEQ NEGNZ
        LDA #$2D
        BSR OUTC1
        PULS A
        JSR PDSSBY
        RTS
NEGNZ LDA #$20
        LBSR OUTC1
        LDA #$2D
        LBSR OUTC1
        PULS    A
        BSR HALFOT
        RTS
******* NOW FOLLOWS THE MEMORY MAPPED OUTPUT ROUTINES.
        ORG VORG+$80
DEMOUT PSHS     A BCD OUTPUT WITH LEADING ZERO AS BLANK.
        BITA #$F0
        BNE LEADCH
        LDA #SPACE
        BSR OUTCHM
        BRA SECCH
LEADCH LSRA
        LSRA
        LSRA
        LSRA
        ADDA #ZERO
        BSR OUTCHM
SECCH PULS      A
        ANDA #$0F
        ADDA #ZERO
OUTCHM LDX MMPNT
        STA ,X+
        STX MMPNT
        RTS
******** BCD WITH LEADING ZERO.
DEMOTZ PSHS     A
        BRA LEADCH
********************
*& HEXOTZ WONT OPERATE CORRECTLY WITH ARROW CHR SET
*HEXOTZ PSHS    A HEXIDECIMAL WITH LEADING ZERO.
*       LSRA
*       LSRA
*       LSRA
*       LSRA
*       BSR ADDON
*       PULS    A
*       ANDA #$0F
*ADDON ADDA #$30
*       CMPA #39
*       BLE OUTCHM
*       ADDA #7
*       BRA OUTCHM
***************************
```

```
BMESOT PSHS U THIS OUTPUTS A MESSAGE ACROSS THE BOTTOM
* OF THE SCREEN. ON ENTRY 'X' SHOULD CONTAIN THE BOTTOM
* ADDRESS OF THE MESSAGE.'A' IS DESTROYED.'04'STOPS.
        TFR     X,U
        LDX LSTLIN,PCR  ;GET LAST LINE
MESLP PULU A
        CMPA #4
        BEQ FINMES
        STA ,X+
        CMPX #MMEM+$400
        BEQ RTNMES
        BRA MESLP
FINMES CLR ,X+ THE RESTOF THE LINE IS CLEARED.
        CMPX #MMEM+$400
        BEQ RTNMES
        BRA FINMES
RTNMES PULS U
        RTS
*************************
TOPSCN LDX #MMEM
        STX MMPNT
        RTS
CLRSCN LDX #MMEM
        LDD #$FAFA                ; SPACE*100 +SPACE
CLRLP STD ,X++
        CMPX #MMEM+$400
        BCS CLRLP
        RTS
MMUPLN  PSHS A,B,X       ;GO UP ONE LINE
        LDD MMPNT        ; GET CURRENT POSITION
        LEAX LNTBL,PCR   ; POINT TO LINE START ADDRESS TABLE
FLY1    CMPD ,X++        ;COMPARE WITH POINTER AND MOVE ON
        BHS FLY1         ;(D-@X) SKIP UNTIL X POINTS TO NEXT LINE
        LDX -6,X         ;CORRECT FOR OVERSHOOT +MIDDLE LINE + UP ONE LINE
        STX MMPNT        ;STORE IT
        PULS A,B,X
        RTS
*
*THIS ROUTINE MOVES THE POINTER TO START
*OF NEXT LINE (WRAPS AROUND TO TOP LINE)
MMLINE PSHS A,B,X
        LDD MMPNT ;ALMOST AS ABOVE
        LEAX LNTBL,PCR
FLY2 CMPD ,X++
        BHS FLY2
        LDX -2,X                ;GET IT
        CMPX #$FFFF       ;HAVE WE HIT SENTRY
        BNE NTTOP
        LDX LNTBL,PCR
NTTOP   STX MMPNT ;MULTIPLE ENTRY POINT
        PULS A,B,X
        RTS
* MOVE TO BOTTOM OF PAGE
BOTPGE PSHS A,B,X
        LDX LSTLIN,PCR
        BRA NTTOP
* MOVE TO TOP OF PAGE
TOPAGE PSHS A,B,X
```

```
         LDX LNTBL,PCR
         BRA NTTOP
         ORG $87B0
LNM1     FDB MMEM        ;SECOND SENTRY
LN0      FDB MMEM               ;SENTRY
LNTBL    FDB MMEM               ;LINE 0
LN2      FDB NCW+MMEM           ;LINE 1
LN2      FDB NCW*2+MMEM         ;LN2
LN3      FDB NCW*3+MMEM         ;LN3
LN4      FDB NCW*4+MMEM         ;LN4
LN5      FDB NCW*5+MMEM         ;LN5
LN6      FDB NCW*6+MMEM         ;LN6
LN7      FDB NCW*7+MMEM         ;LN7
LN8      FDB NCW*8+MMEM         ;LN8
LN9      FDB NCW*9+MMEM         ;LN9
LN10     FDB NCW*10+MMEM        ;LN10
LN12     FDB NCW*11+MMEM        ;LN11
LN12     FDB NCW*12+MMEM        ;LN12
LN13     FDB NCW*13+MMEM        ;LN13
LN14     FDB NCW*14+MMEM        ;LN14
LN15     FDB NCW*15+MMEM        ;LN15
LN16     FDB NCW*16+MMEM        ;LN16
LN17     FDB NCW*17+MMEM        ;LN17
LN18     FDB NCW*18+MMEM        ;LN18
LN19     FDB NCW*19+MMEM        ;LN19
LN20     FDB NCW*20+MMEM        ;LN20
LN21     FDB NCW*21+MMEM        ;LN21
LN22     FDB NCW*22+MMEM        ;LN22
LSTLIN   FDB NCW*23+MMEM        ;LN23
         FDB $FFFF              ;SENTRY
         END
```

```
C
C GENERATES A M680? FORMAT TABLE FOR CRTC
C OUTPUT IS TO FILE TABLE.DAT
C VARS  CRTC REGS =IR?? ;NO.DOTS/CHAR IDPC
C TOTAL NO. OF CHARS ACROSS SCREEN =ICAS  ;NO. LINES/CHAR=ILPC
C TOTAL NO OF ROWS=INR   ;FREQ. ADJUST=IFAQJ  ;TEMPORARIES ?TEMP?
C POM = CLOCK FREQ/MAINS FREQ
 1010   FORMAT(I3)
 1020   FORMAT(I5)
C STANDARD INTEGER INPUT
 1050   FORMAT(I7)
C CLOCK INPUT FREQ FORMAT
 2010   FORMAT(7I6)
C STANDARD OUTPUT FORMAT
C       OPEN(UNIT=1,NAME='TABLE.DAT')
        CALL ASSIGN(1,'TABLE.DAT')
 999    TYPE 20
        TYPE 25
 20     FORMAT(17H MAINS 20MS(50HZ))
 25     FORMAT(26H CLOCK FREQ IN KHZ(13500)?$)
        ACCEPT 1050,ICLKF
        CLKF=ICLKF
        POM=CLKF * 20.0
        TYPE 30
 30     FORMAT(31H DOTS PER CHAR(HARDWARE 9)(I3)?$)
        ACCEPT 1010,IDPC
C       NO OF CHARS =40mS x CLOCK FREQ /(625 lines on screen X dots/char)
        TEMP1=625*IDPC
        ICAS=(40.0 * CLKF)/TEMP1
        TYPE 40,ICAS
 40     FORMAT(41H HENCE TOTAL NO CHARS ACROSS SCREEN(R0+1), I4)
        TYPE 50
 50     FORMAT(31H NO. LINES PER CHAR (R9+1)(I3)?$)
        ACCEPT 1010,ILPC
        ITEMP= IDPC*ICAS*ILPC
        TEMP1= ITEMP
        INR=  POM /TEMP1
        TEMP1= ICAS * IDPC
        TEMP2=  POM /TEMP1
        TEMP1= INR*ILPC
        IFADJ= TEMP2- TEMP1 +0.5
C
        IR4=INR-1
        IR5=IFADJ
        IR0=ICAS-1
        IR9=ILPC-1
C
        TYPE 60,INR
 60     FORMAT(16H HENCE NO ROWS= ,I4)
        TYPE 70,IFADJ
 70     FORMAT(16H AND FREQ ADJ = ,I4)
        TYPE 80
 80     FORMAT(45H NO CHARS DISPLAYED ACROSS SCREEN(R1 48)(I3)?$)
        ACCEPT 1010,IR1
        TYPE 90,IR1,IR0
 9      FORMAT(23H HORIZ. SYNC. POS (R2),I4,1H-,I4,2H ?$)
        ACCEPT 1010,IR2
        IR3= ICLKF/(200 * IDPC)
        TYPE 100,IR3
```

```
C
 100    FORMAT(31H DOTS PER H.S.PULSE (R3)EQUALS: ,I3)
C NO ROWS DISPLAYED = TOTAL NO OF CHARS (1K) / CHARS/LINE DISPLAYED
        IR6= 1024/IR1
        TYPE 110,IR6
 110    FORMAT(32H NO. ROWS DISPLAYED (R6) EQUALS:,I4)
        TYPE 120
 120    FORMAT(31H VERTICAL SYNC POS (R7 29)(I3)?$)
        ACCEPT 1010,IR7
        TYPE 130
 130    FORMAT(22H INTERLACE (R8 0)(I3)?$)
        ACCEPT 1010,IR8
        TYPE 140
 140    FORMAT(27H CURSOR START (R10 64)(I3)?$)
        ACCEPT 1010,IR10
        TYPE 150
 150    FORMAT(24H CURSOR END (R11 1)(I3)?$)
        ACCEPT 1010,IR11
        TYPE 160
 160    FORMAT(24H START ADDR (R12 0)(I5)?$)
        ACCEPT 1020,IR12
        TYPE 170
 170    FORMAT(32H CURSOR LOCATION (R14 1023)(I5)?$)
        ACCEPT 1020,IR14
        TYPE 2010,IR0,IR1,IR2,IR3,IR4,IR5,IR6
        TYPE 2010,IR7,IR8,IR9,IR10,IR11,IR12,IR14
        TYPE 200
 200    FORMAT(12H FIX(1=YES)?$)
        ACCEPT 1010,ITEMP
        IF ( ITEMP .EQ. 1) GOTO 999
C
        WRITE(1,5010)ICLKF
        WRITE(1,5020) IDPC
 5010   FORMAT(16H* CRTC TABLE FOR,I7,3HKHz)
 5020   FORMAT(19H* NO. DOTS PER CHAR,I4)
        WRITE (1,6000)IR0
        WRITE(1,6010)IR1
        WRITE(1,6020)IR2
        WRITE(1,6030)IR3
        WRITE(1,6040)IR4
        WRITE(1,6050)IR5
        WRITE(1,6060)IR6
        WRITE(1,6070)IR7
        WRITE(1,6080)IR8
        WRITE(1,6090)IR9
        WRITE(1,6100)IR10
        WRITE(1,6110)IR11
        WRITE(1,6120)IR12
        WRITE(1,6140)IR14
        WRITE(1,6150)
C
 6000  FORMAT(10HTABLE FCB ,I5,19H  ;R0 NO CHARS/LINE)
 6010   FORMAT(5H FCB ,I5,24H  ;R1 DISPLAYED PER LINE)
 6020   FORMAT(5H FCB ,I5,18H  ; HORIZ SYNC POS)
 6030   FORMAT(5H FCB ,I5,23H  ; DOTS/HOR SYNC PULSE)
 6040   FORMAT(5H FCB ,I5,16H  ;TOTAL NO ROWS)
 6050   FORMAT(5H FCB ,I5,11H  ;FREQ ADJ)
 6060   FORMAT(5H FCB ,I5,17H  ;ROWS DISPLAYED)
 6070   FORMAT(5H FCB ,I5,16H  ;VERT SYNC POS)
```

```
C
 6080   FORMAT(5H FCB ,I5,12H  ;INTERLACE)
 6090   FORMAT(5H FCB ,I5,13H  ;LINES/CHAR)
 6100   FORMAT(5H FCB ,I5,15H  ;CURSOR START)
 6110   FORMAT(5H FCB ,I5,13H  ;CURSOR END)
 6120   FORMAT(5H FDB ,I5,13H  ;START ADDR)
 6140   FORMAT(5H FDB ,I7,13H  ;CURSOR LOC)
 6150   FORMAT(22H FDB 0      ;LIGHT PEN)
        STOP
        END
```

```
C
>RUN TABLE
MAINS 20MS(50HZ)
CLOCK FREQ IN KHZ(13500)?8300
DOTS PER CHAR(HARDWARE 9)(I3)?9
HENCE TOTAL NO CHARS ACROSS SCREEN(R0+1)  59
NO. LINES PER CHAR (R9+1)(I3)?10
HENCE NO ROWS=    31
AND FREQ ADJ =     3
NO CHARS DISPLAYED ACROSS SCREEN(R1 48)(I3)?
HORIZ. SYNC. POS (R2),  44-  58 ?53
DOTS PER H.S.PULSE (R3)EQUALS:  4
NO. ROWS DISPLAYED (R6) EQUALS:  23
VERTICAL SYNC POS (R7 29)(I3)?27
INTERLACE (R8 0)(I3)?
CURSOR START (R10 64)(I3)?
CURSOR END (R11 1)(I3)?1
START ADDR (R12 0)(I5)?
CURSOR LOCATION (R14 1023)(I5)?1023
    58    44    53     4    30     3    23
    27     0     9     0     1     0  1023
FIX(1=YES)?
TTO  -- STOP


 * CRTC TABLE FOR   8300KHz
 * NO. DOTS PER CHAR   9
 TABLE FCB     58  ;R0 NO CHARS/LINE
  FCB    44   ;R1 DISPLAYED PER LINE
  FCB    53   ; HORIZ SYNC POS
  FCB     4   ; DOTS/HOR SYNC PULSE
  FCB    30   ;TOTAL NO ROWS
  FCB     3   ;FREQ ADJ
  FCB    23   ;ROWS DISPLAYED
  FCB    27   ;VERT SYNC POS
  FCB     0   ;INTERLACE
  FCB     9   ;LINES/CHAR
  FCB     0   ;CURSOR START
  FCB     1   ;CURSOR END
  FDB     0   ;START ADDR
  FDB    1023   ;CURSOR LOC
  FDB 0       ;LIGHT PEN
```

```
C
C       MAIN.FTN
C       THE orginal program that did everything but make coffee
C       (although rather clumsily)
C       later programs use this as a basis
C
C
        DIMENSION NCHARS(256,8) ! THE CHARACTER 9:T
        COMMON NCHARS
 1      WRITE (5,10)            !RESET
        WRITE (5,11)
 10     FORMAT (1X,'WHAT DO YOU WANT TO DO?')
 11     FORMAT (1X,'1=INPUT CHARS, 2=PLOT CHARS, 3:DRAW, 9=EXIT')
        READ (5,15) I
 15     FORMAT (I5)
        IF (I.EQ.1) GOTO 100    ! JMMP TO APPROPRIATE CODE
        IF (I.EQ.2) GOTO 200
        IF (I.EQ.3) GOTO 300
        IF (I.EQ.9) GOTO 1000
        WRITE (5,18)
 18     FORMAT (1X,'ERROR')
        GOTO 1
C
C       CALLS INPUT CHARACTER SET SUBROUTINE
C
 100    CALL INCSET             !ALL DONE BY SUBROUTINE
        GOTO 1
C
C       CALL PLOTTER ROUTINE
C
 200    WRITE (5,201)
 201    FORMAT (1X,'DO YOU WANT TO PLOT? (1=YES) ')
C               historical (time to turn on plotter)
        READ (5,202) I
 202    FORMAT (I5)
        IF (I.NE.1) GOTO 1
        CALL ASSIGN(4,'PB:')    !INITIALISE PLOTTER
        CALL PLOTS (3,4,4)
        CALL LIMIT (0.,6.,0.,7.)
        CALL SETIN
        CALL LOCATE (0.0,6.0,0.0,6.0)
        CALL SETCHR
        WRITE (5,210)
 210    FORMAT (1X,'USER UNITS: XMIN, XMAX, YMIN, YMAX (WITH COMMAS)')
        READ (5,220) XMIN,XMAX,YMIN,YMAX
 220    FORMAT (4F10.7)
        CALL MA:UO (XMIN,XMAX,YMIN,YMMX)
 230    WRITE (5,240)
 240    FORMAT (1X,'CHAR NO., X, Y (ALL INTEGERS)')
        READ (5,250) ICH, IX, IY
 250    FORMAT (3I8)
        IF (ICH) 290,260,260    !NEGATIVE EXIT
 260    DO 270 ICR=1,8
        IRCH=NCHARS(ICH,ICR)    !GET A ROW OF CHARACTER
        IYY=IY+8-IRCH
        WRITE(5,265)IRCH,IX,IYY !TRACE
 265    FORMAT(3(2X,I8))
        CALL PLTLN (IRCH,IX,IYY) !PLOT IT
 270    CONTINUE
```

```
C
        GOTO 230
 290    CALL NEWPEN(0)           !RELEASE PLOTTER
        CALL PLOT(0.0,0.0,999)
        CALL CLOSE(4)
        GOTO 1
C
C       SAME AS PLOTTER PROGRAM EXCEPT THAT SINGLE CHAR
C       IS DISPLAYED ON SCREEN
C
 300    CALL SETCHR
 305    WRITE (5,310)
 310    FORMAT(1X,'CHAR NUMBER ')
        READ (5,320) ICH
 320    FORMAT (I8)
        IF (ICH) 380,330,330             !NEGATIVE EXIT
 330    DO 340 ICR=1,8                   !ROW BY ROW
        IRCH=NCHARS(ICH,ICR)
        CALL DISPLN(IRCH)        !DISPLAY LINE
 340    CONTINUE
        GOTO 305
 380    CONTINUE
        GOTO 1
 1000   STOP
        END
```

```
C
C
C
C       The character inputting routine
C       Characters are input line by line, character by character in
C       the order char. 1 char. 2 etc. until the first 48 characters
C       have been input.(48=3x16 i.e. all arrows pointing North starting
C       amplitude 1,2..15,blank then North-East and finally NNE. This is
C       because SETALL will decode the other angles for me)
C
C
        SUBROUTINE INCSET
        DIMENSION LINE(8)
C       READS UP TO 256 CHARACTERS
C       ROW BY ROW IN FORMAT
C
C       00010000
C
C       IF ONE NUMBER OF THE EIGHT IS A 8 OR 9
C       THEN THE REST OF THE CHARACTERS ARE
C       ASSUMED TO BE BLANK.
C
        CALL ASSIGN(1,'CHRSET.DAT')
        ICHRCT=0                    ! CURRENT CHARACTER NUMBER
 10     ICHRCT=ICHRCT+1
        IROWCT=0                    ! ROW COUNTER
        WRITE(5,15)ICHRCT
 15     FORMAT(1X,'CHR NO   ',I8)
 20     IROWCT=IROWCT+1
        NCRIPT=0                    ! ENCRIPTION OF ROW AS INTEGER
        WRITE(5,25)IROWCT
 25     FORMAT(1X,'ROW NUMBER   ',I8)
        IBLANK=0                    ! END OF CHR. SET FLAG
C       INPUT ROW AS 8X1 DIGIT NUMBERS BACK TO BACK
        READ(5,40)(LINE(I),I=1,8)
 40     FORMAT(8I1)
        DO 70 I=1,8
C       RUN THRU 8 TIMES ;I NOT USED; IBIT -CURRENT BIT
        IBIT=LINE(I)
        IF (IBIT-8)60,50,50
 50     IBLANK=1                    !SET FLAG
        IBIT=IBIT-8
 60     NCRIPT=NCRIPT*2+IBIT        !ENCODE AS BINARY NUMBER
 70     CONTINUE
C       WRITE ENCRIPTED FORM TO FILE
        INDEX=ICHRCT*10+IROWCT   !A CHECK
        WRITE(1,80)NCRIPT,INDEX
 80     FORMAT(1X,I6,2X,I6)
        IF(ICHRCT-256)90,120,120            ! IF LAST CHAR
 90     IF (IBLANK)100,100,110
 100    IF (IROWCT-8)20,10,10               ! IF LAST ROW OF CHR.
C
C
 110    NCRIPT=0
        INDEX=0
        DO 120 I=ICHRCT,256
        DO 120 J=1,8
        WRITE(1,80)NCRIPT,INDEX             !SAME FORMAT
```

```
C
 120     CONTINUE
C
         CALL CLOSE(1)
         RETURN
         END
C
C
C
C
C
C        This routine will appear in many other programs(both documented & no
C
         SUBROUTINE PLTLN (ICH,IX,IY)
         DIMENSION ICHMAT(8)
C
C        PLOTS  THE ROW OF A CHARACTER ENCODED AS AN INTEGER
C
         IBYTE1=ICH
         DO 20 J=8,1,-1                              !DECODE ENCODED FORM
         IBYTE2=IBYTE1/2            !GET EACH BIT
         ICHMAT(J)=IBYTE1-IBYTE2*2
         IBYTE1=IBYTE2
  20     CONTINUE
         CALL NEWPEN(3)             !GRAB BLUE PEN
         Y=IY                       !HP ROUTINES REQUIRE REAL NOS.
         DO 30 J=1,8                                 !RUN ALONG LINE
C*
         WRITE(5,25)IX,IY,ICHMAT(J),J     !TRACE TO TTY
  25     FORMAT(4(2X,I8))
         IF (ICHMAT(J)) 30,30,40                     !SEE IF DOT
  40     X=IX+J
C*
         WRITE(5,35)X,Y,J
  35     FORMAT(1X,F10.8,F10.8,I8)
         CALL PLOT(X,Y,1)
  30     CONTINUE
         CALL PENUP                 !PUT PEN AWAY
         CALL NEWPEN (0)            !SINCE ONE AT A TIME
         RETURN
         END
```

```
C
C       This subroutine is also used frequently (intended for tracing)
C
C
        SUBROUTINE DISPLN(ICH)
        DIMENSION LINEN(8)                      !LINE OF NUMBERS
        DIMENSION LINEC(8)                      !LINE OF CHARS
C
C       DISPLAYS A LINE OF CHARACTERS FROM ENCODED FORM IN ICH
C
        IBY1=ICH
        DO 10 J=8,1,-1
        IBY2=IBY1/2
        LINEN(J)=IBY1-IBY2*2                    !DECODE ICH
        IBY1=IBY2                       !PUT EACH BIT INTO ARRAY
 10     CONTINUE
        DO 20 I=1,8                             !SET CHAR LINE
        IF (LINEN(I).EQ.0) LINEC(I)= "20056       ! EQUAL TO ' .'
        IF (LINEN(I).EQ.1) LINEC(I)= "20130       ! EQUAL TO ' X'
 20     CONTINUE
        WRITE (5,30) (LINEC(J),J=1,8)
 30     FORMAT (1X,8A1)
        RETURN
        END
C
C
C
C       This subroutine is the most important of all.
C       SETCHR sets up an array of 256 characters in groups
C       of 8 lines (per char) each line is an encoded form
C       of the 8 dots per line of the character (in a binary
C       form since this packs best and transferrs to ROM easily)
C
C
        SUBROUTINE SETCHR
        DIMENSION NCHARS(256,8)
        COMMON NCHARS
        CALL ASSIGN (1,'CHRSET.DAT')
        INDEOF=0
        DO 40 ICHRCT=1,256
        DO 40 IROWCT=1,8
        IF (INDEOF.EQ.1) GOTO 30                !IF END OF INPUT FILE
        READ (1,10) NCRIPT,INDEX                !GET CHAR + CHECK NUMBER
 10     FORMAT (1X,I6,2X,I6)
        IF (INDEX.EQ.0) GOTO 20                 !=END OF INPUT FILE
        IF (ICHRCT*10+IROWCT.NE.INDEX) GOTO 50 !IS FILE OK?
        GOTO 30
 20     NCRIPT=0
        INDEOF=1
 30     NCHARS (ICHRCT,IROWCT)=NCRIPT   !STORE IN ARRAY
 40     CONTINUE
        GOTO 70                         !EXIT
 50     WRITE (5,60) ICHRCT,IROWCT,NCRIPT,INDEX
 60     FORMAT (1X,'SEQUENCE ERROR  CHRCT',I8,'ROWCT',I8,
     1  'ENCRIPT',I8,'INDEX',I8)
 70     CALL CLOSE(1)
        RETURN
        END
```

```
C
C       The arrow editing program.
C       This is necessary because it is impossible to input 8x8x48
C       numbers correctly (or 256 instead of 48 if SETALL didn't exist)
C       CHANGE is also a fairly early program and has some annoying
C       little command sequences. Since the advent of SETALL the only
C       edit features commonly used were NEW and SWAP (mainly typing
C       errors in MAIN).
        DIMENSION ICHM(8,8)       !A COMPLETE CHARACTER
        DIMENSION LINE(8)         !A LINE IN CHAR
        DIMENSION NCHARS(256,8)
        COMMON NCHARS
C
C       CONTROLLER
C
        CALL SETCHR               !GET CURRENT CHAR SET
 10     WRITE(5,15)               !RESET
 15     FORMAT(' REQUEST (1=COPY,2=SWAP, 3=DISPLAY,
     1  4=ROTATE, 5=NEW, 9=EXIT)')
        READ (5,20)I
 20     FORMAT(I8)
        IF(I.EQ.1) GOTO 100
        IF(I.EQ.2) GOTO 200
        IF(I.EQ.3) GOTO 300
        IF(I.EQ.4) GOTO 400
        IF(I.EQ.5) GOTO 500
        IF(I.EQ.9) GOTO1000
        GOTO 10                   !TWIT!!
C
C
C       COPY
C
 100    WRITE(5,110)
 110    FORMAT(' SOURCE, DESTINATION')
        READ (5,120)M,N
 120    FORMAT(2I8)
        DO 130 I=1,8
 130    NCHARS(N,I)=NCHARS(M,I) !OVERWRITE
        GOTO 10
C
C
C       SWAP
C
 200    WRITE(5,210)
 210    FORMAT(1X,'CHAR1,CHAR2')
        READ (5,220)M,N
 220    FORMAT(2I8)
        DO 230 I=1,8
        IDUMMY = NCHARS(N,I)
        NCHARS(N,I) = NCHARS(M,I)
        NCHARS(M,I) = IDUMMY
 230    CONTINUE
        GOTO 10
C
C
C       DISPLAY
C
 300    WRITE(5,310)
```

```
C
 310    FORMAT(' CHAR. NUMBER')
        READ(5,320)ICH
 320    FORMAT(I8)
        DO 330 ICR=1,8
        IRCH = NCHARS(ICH,ICR)           ! GET LINE
        CALL DISPLN(IRCH)                ! DISPLAY LINE
 330    CONTINUE
        GOTO 10
C
C
C       ROTATE
C
C
 400    WRITE(5,405)
 405    FORMAT(' CHAR. NUMBER')
        READ(5,410)ICH
 410    FORMAT(I8)
        DO 420 I=1,8
        IB1=NCHARS(ICH,I)        !SET UP CHAR IN 8x8 ARRAY
        DO 420 J=8,1,-1
        IB2=IB1/2
        ICHM(I,J) = IB1-IB2*2
        IB1=IB2
420     CONTINUE
        WRITE(5,425)
 425    FORMAT(1X,'ROTATION AXIS (1=E,2=N,3=NE,4=NW)')
        READ(5,430)I
 430    FORMAT(I8)
        IF(I.EQ.1)GOTO 440       !NORTH IS TOP OF SCREEN
        IF(I.EQ.2)GOTO 450
        IF(I.EQ.3)GOTO 460
        IF(I.EQ.4)GOTO 470
        WRITE (5,435)
 435    FORMAT(' NO CHANGE')
        GOTO 480
C
C       ROTATE ABOUT EAST AXIS
 440    DO 445 I=1,4
        DO 445 J=1,8
        IA = ICHM(I,J)   !SWAP AROUND WITHIN ARRAY
        ICHM(I,J)=ICHM(9-I,J)
 445    ICHM(9-I,J)=IA
        GOTO 480
C
C       ROT. ABOUT NORTH AXIS
 450    DO 455 I=1,8
        DO 455 J=1,4
        IA= ICHM(I,J)
        ICHM(I,J) = ICHM(I,9-J)
 455    ICHM(I,9-J)=IA
        GOTO 480
C
C       ROT. ABOUT N.E. AXIS
 460    DO 465 I=1,7
        DO 465 J=1,8-I
        IA=ICHM(I,J)
        ICHM(I,J)=ICHM(9-J,9-I)
 465    ICHM(9-J,9-I)=IA
```

```
C
        GOTO 480
C
C       ROT. ABOUT N.W. AXIS
 470    DO 475 I= 1,7
        DO 475 J= (I+1),8
        IA= ICHM(I,J)
        ICHM(I,J)= ICHM(J,I)
 475    ICHM(J,I)= IA
C
C       DECODE BACK AGAIN
C
 480    DO 490 I=1,8
        NCRIPT=0
        DO 485 J=1,8
 485    NCRIPT=NCRIPT*2 + ICHM(I,J)
        NCHARS(ICH,I)=NCRIPT
 490    CONTINUE
        GOTO 10
C
C
C       CREATE A NEW CHARACTER
C
 500    WRITE(5,505)
 505    FORMAT(' CHAR. NUMBER')
        READ(5,510)ICH
 510    FORMAT(I8)                !SAME AS INCSET
        DO 525 I=1,8
        WRITE(5,515)I
 515    FORMAT(' ROW NUMBER',I8)
        READ(5,520)(LINE(J),J=1,8)
 520    FORMAT(8I1)
        NCRIPT=0
        DO 522 J=1,8
 522    NCRIPT=NCRIPT*2+LINE(J)
 525    NCHARS(ICH,I)=NCRIPT
        GOTO 10
C
C
C       EXIT
C           CLEAN UP
C
 1000   CALL ASSIGN(1,'CHRSET.DAT')      !WRITE BACK EDITED SET
        DO 1020 ICHRCT=1,256
        DO 1020 IROWCT=1,8
        INDEX= ICHRCT*10 + IROWCT
        WRITE(1,1010) NCHARS(ICHRCT,IROWCT), INDEX
 1010   FORMAT(1X,I6,2X,I6)
 1020   CONTINUE
        CALL CLOSE(1)
        STOP
        END
C       SETCHR and DISPLN are as above.
```

```
C
C       SETALL
C           This prosram builds a complete 240 character arrow
C       set from a minimium arrow set of 45 (+3 dummy chars for my ease).
C       This means less typins and hence less errors and less time wasted.
C
C       Firstly we copy the current arrow set into a second array and
C       then proceed to pluck out an arrow from the minimal set and rotate
C       it to the required ansle by 0,1 or 2 rotations (IA1&IA2 5= do nothins
C       The new array is then written back and chars. 241 to 256 inserted
C       by CHANGE.
C
C
        DIMENSION ICHM(8,8)
        DIMENSION LINE(8)
        DIMENSION NCHARS(256,8)
        DIMENSION NCHAR2(256,8)
        COMMON NCHARS,NCHAR2
C
C       CONTROLLER
C
        CALL SETCHR
        CALL SETCH2
        WRITE(5,15)
 15     FORMAT('$ CHECK ONE BY ONE? (1=YES) ')  !TRACING ONLY
        READ(5,17)IAUTO
 17     FORMAT(I1)
        DO 999 IA = 1,15          !15 AMPLITUDES &
        DO 999 IP = 1,16          !16 DIRECTIONS
        ISRC = 32                 !HERE ON IS REALLY A TABLE
        IF((IP.EQ.4).OR.(IP.EQ.8).OR.(IP.EQ.12).OR.(IP.EQ.16))ISRC=0
        IF((IP.EQ.2).OR.(IP.EQ.6).OR.(IP.EQ.10).OR.(IP.EQ.14))ISRC=16
        ISRC = ISRC+IA
        IDEST =(IA - 1)*16 + IP          ! AMPL=1;PHASE=1 => 1  MAP IS TO 000
        IF(IP.NE.1)GOTO 101
        IA1=3              !TWO ROTATIONS MAY BE NECESSARY
        IA2=5
        GOTO 200
 101    IF(IP.NE.2)GOTO 102
        IA1=5
        IA2=5
        GOTO 200
 102    IF(IP.NE.3)GOTO 103
        IA1=5
        IA2=5
        GOTO 200
 103    IF(IP.NE.4)GOTO 104
        IA1=5
        IA2=5
        GOTO 200
 104    IF(IP.NE.5)GOTO 105
        IA1=2
        IA2=5
        GOTO 200
 105    IF(IP.NE.6)GOTO 106
        IA1=2
        IA2=5
        GOTO 200
```

```
C
106     IF(IP.NE.7)GOTO 107
        IA1=3
        IA2=2
        GOTO 200
107     IF(IP.NE.8)GOTO 108
        IA1=4
        IA2=5
        GOTO 200
108     IF(IP.NE.9)GOTO 109
        IA1=4
        IA2=5
        GOTO 200
109     IF(IP.NE.10)GOTO 110
        IA1=4
        IA2=5
        GOTO 200
110     IF(IP.NE.11) GOTO 111
        IA1=1
        IA2=2
        GOTO 200
111     IF(IP.NE.12)GOTO 112
        IA1=1
        IA2=5
        GOTO 200
112     IF(IP.NE.13)GOTO 113
        IA1=1
        IA2=5
        GOTO 200
113     IF(IP.NE.14)GOTO 114
        IA1=1
        IA2=5
        GOTO 200
114     IF(IP.NE.15)GOTO 115
        IA1=3
        IA2=1
        GOTO 200
115     IF(IP.NE.16)GOTO 116
        IA1=3
        IA2=5
        GOTO 200
116     WRITE(5,120)
120     FORMAT(' HELP IMPOSSIBLE PHASE ANGLE')
        STOP
200     CONTINUE
C
C?
        IF(IAUTO.NE.1) GOTO 208
        WRITE(5,205)IA,IP,IA1,IA2,ISRC,IDEST
 205  FORMAT(' A',I4,'P',I4,'A1',I4,'A2',I4,'SRC',
        1  I6,'DEST',I6,'CON(O=Y)')
        READ(5,207)III
 207  FORMAT(I1)
        IF(III.NE.0)GOTO 1000    !EXIT
 208    CONTINUE
C
C
C       the rest of the program is copied from CHANGE
C
```

```
C
C       COPY
C
        DO 210 I=1,8
 210   NCHARS(IDEST,I)=NCHAR2(ISRC,I)      !MAKE A COPY
        WRITE(5,211)(NCHARS(IDEST,IDDT),IDDT=1,8)
 211   FORMAT(1X,8I8)
C
C
C       ROTATE
C
C
        ICH=IDEST
        IF1=1
 430   DO 420 I=1,8
        IB1=NCHARS(ICH,I)
        DO 420 J=8,1,-1  !INTO ARRAY
        IB2=IB1/2
        ICHM(I,J) = IB1-IB2*2
        IB1=IB2
 420    CONTINUE
        IF(IF1.EQ.1)I=IA1
        IF(IF1.EQ.2)I=IA2                  !  FIRST OR SECOND OF THE 2 ROTATION
        IF(IF1.GT.2)GOTO 600               ! EXIT ROTATE SECTION
        IF1=IF1+1
        IF(I.EQ.1)GOTO 440         !ROTATION AXES
        IF(I.EQ.2)GOTO 450
        IF(I.EQ.3)GOTO 460
        IF(I.EQ.4)GOTO 470
        IF(I.EQ.5)GOTO 480
        WRITE (5,435)
 435   FORMAT(' NO CHANGE')
        GOTO 480
C
C       ROTATE ABOUT EAST AXIS
 440   DO 445 I=1,4
        DO 445 J=1,8
        IDUMY = ICHM(I,J)
        ICHM(I,J)=ICHM(9-I,J)
 445   ICHM(9-I,J)=IDUMY
        GOTO 480
C
C       ROT. ABOUT NORTH AXIS
 450   DO 455 I=1,8
        DO 455 J=1,4
        IDUMY= ICHM(I,J)
        ICHM(I,J) = ICHM(I,9-J)
 455   ICHM(I,9-J)=IDUMY
        GOTO 480
C
C       ROT. ABOUT N.E. AXIS
 460   DO 465 I=1,7
        DO 465 J=1,8-I
        IDUMY=ICHM(I,J)
        ICHM(I,J)=ICHM(9-J,9-I)
 465   ICHM(9-J,9-I)=IDUMY
        GOTO 480
C
C       ROT. ABOUT N.W. AXIS
```

```
C
 470   DO 475 I= 1,7
         DO 475 J= (I+1),8
         IDUMY= ICHM(I,J)
         ICHM(I,J)= ICHM(J,I)
 475   ICHM(J,I)= IDUMY
C
C        DECODE BACK AGAIN
C
480      DO490 I=1,8
         NCRIPT=0
         DO 485 J=1,8
 485   NCRIPT=NCRIPT*2 + ICHM(I,J)
         NCHARS(ICH,I)=NCRIPT
 490   CONTINUE
         GOTO 430
C
C
C        DISPLAY CHARACTER
C
 600   ICH=IDEST
         WRITE(5,610)ICH
 610   FORMAT('0 CHAR.NO.:',I6)
         DO 630 ICR=1,8
         IRCH=NCHARS(ICH,ICR)
         CALL DISPLN(IRCH)
C  YES IT CAN BECOME ANNOYING WATCHING EACH CHARACTER BUT THATS HISTORY
 630   CONTINUE
C
C        END OF MAIN LOOP
999      CONTINUE
C
C        EXIT
C                 CLEAN UP
C
 1000 CALL ASSIGN(1,'CHRSET.DAT')
         DO 1020 ICHRCT=1,256
         DO 1020 IROWCT=1,8
         INDEX= ICHRCT*10 + IROWCT
         WRITE(1,1010) NCHARS(ICHRCT,IROWCT), INDEX
 1010 FORMAT(1X,I6,2X,I6)
 1020 CONTINUE
         CALL CLOSE(1)
         STOP
         END
C
C        THIS SUBROUTINE COPIES INITIAL CHARACTER SET TO  A TEMPORARY AREA
         SUBROUTINE SETCH2
         DIMENSION NCHAR2(256,8)
         DIMENSION NCHARS(256,8)
         COMMON NCHARS,NCHAR2
         WRITE(5,10)
 10      FORMAT('          ENTERING SETCH 2')
         DO 20 I=1,256
         DO 20 J=1,8
         NCHAR2(I,J)=NCHARS(I,J)
 20      CONTINUE
         RETURN
         END
```

```
C
C       PLTALL
C       Here we plot characters as specified in the file 'PLT.DAT'
C       this was initially for generality so that any series of characters
C       could be plotted, but at time of writing the only serious PLOT file
C       that is used is the one set up in another part of the program.
C       A program that reads actual data from Bribie is being written so
C       that the actual screen can be represented.
C       A plot file is terminated by a negative character.
C
        DIMENSION NCHARS(256,8)
        COMMON NCHARS
 1      WRITE (5,10)
        WRITE (5,11)
 10     FORMAT (1X,'WHAT DO YOU WANT TO DO?')
 11     FORMAT (1X,'2=PLOT CHARS, 3=DRAW, 4=SET UP PLOT, 9=EXIT')
        READ (5,15) I
 15     FORMAT (I5)
        IF (I.EQ.2) GOTO 200
        IF (I.EQ.3) GOTO 300
        IF (I.EQ.4) GOTO 400
        IF (I.EQ.9) GOTO 1000
        WRITE (5,18)
 18     FORMAT (1X,'ERROR')
        GOTO 1
C
C       CALL PLOTTER ROUTINE
C
 200    CALL ASSIGN(3,'PLT.DAT')          !OPEN "PLOT" FILE
        CALL ASSIGN(4,'PB:')              !INITIALIZE PLOTTER
        CALL PLOTS (3,4,4)
        CALL LIMIT (0.,12.,0.,12.)
        CALL SETIN
        CALL LOCATE (1.0,8.0,1.0,8.0)
        CALL SETCHR
        CALL MAPUU (0.,320.,0.,320.)      !320 IS A MAGIC NUMBER
        DO 270 II=1,256            ! RUN THRU ENTIRE SET =256=16X16
        READ (3,250) ICH, IX, IY
 250    FORMAT (3I8)
        IF (ICH) 290,260,260
 260    DO 270 ICR=1,8
        IRCH=NCHARS(ICH,ICR)
        IYY=IY+8-ICR
C*
        CALL PLTLN (IRCH,IX,IYY)
 270    CONTINUE
 290    CALL NEWPEN(0)
        CALL PLOT(0.0,0.0,999)
        CALL CLOSE(4)
        CALL CLOSE(3)
        GOTO 1
C
C       A SINGLE CHARACTERIS DISPLAYED ON SCREEN
C
 300    CALL SETCHR
 305    WRITE (5,310)
 310    FORMAT(1X,'CHAR NUMBER ')
        READ (5,320) ICH
```

```
C
 320     FORMAT (I8)
         IF (ICH) 380,330,330              !NEGATIVE EXIT
 330     DO 340 ICR=1,8                    !ROW BY ROW
         IRCH=NCHARS(ICH,ICR)
         CALL DISPLN(IRCH)                          !               DISPLAY LINE
 340     CONTINUE
         GOTO 305
 380     CONTINUE
         GOTO 1
 400     CALL ASSIGN(3,'PLT.DAT')
         ICH=0
         DO 450 IY=1,16
         DO 450 IX=1,16
         ICH=ICH+1
         WRITE (5,410)IX,IY,ICH
 410     FORMAT(' X=',I3,'   Y=',I3,'  CHARACTER NO.=',I4)
         IXX=(IX-1)*16                     ! POSITION X,Y NOTING 8X8 /CHAR
         IYY=(IY-1)*16
         WRITE (3,430)ICH,IXX,IYY
 430     FORMAT(3I8)
 450     CONTINUE
         CALL CLOSE(3)
         GOTO 1
 1000    CONTINUE                !EXIT
         STOP
         END
C
C
         SUBROUTINE PLTLN (ICH,IX,IY)
         DIMENSION ICHMAT(8)
C
C        PLOTS  THE ROW OF A CHARACTER ENCODED AS AN INTEGER
C
         IBYTE1=ICH
         DO 20 J=8,1,-1                              !DECODE ENCODED FORM
         IBYTE2=IBYTE1/2
         ICHMAT(J)=IBYTE1-IBYTE2*2
         IBYTE1=IBYTE2
 20      CONTINUE
         CALL NEWPEN(4)
         Y=IY
         DO 30 J=1,8                                 !RUN ALONG LINE
C*
         IF (ICHMAT(J)) 30,30,40                     !SEE IF DOT
 40      X=IX+J
         CALL PLOT(X,Y,1)
 30      CONTINUE
         CALL PENUP
         RETURN
         END
```

```
C
C       BIGBIN
C               BIGBIN is an enlarged (improved?) version of
C       the BIN program. It writes a file of encoded arrows (CHRSET.DAT)
C       to a binary file CHRSET.BIN which is composed of bytes with
C       each bit in the file representing a dot of a character.
C       This enables the character set to be transferred to ROM via
C       paper tape and a program on Dr. Hainsworth's M6800 that writes ROMs
C       (Because of the very good packing density of the binary file comparec
C       with the ascii file (about 14 to 1) intermediate version can be store
C       without wasting space.)
C
C               The reversing of the process is complicated by PDP11s
C       sign-extending bytes when unpacked to words hence the macro-11 routir
C
        DIMENSION NCHARS(256,8)
        BYTE BIT(2048)
        COMMON NCHARS
        TYPE 1             !WHICH DIRECTION OF TRANSFER?
1       FORMAT(' BINARY TO ASCII?(1=YES)',$)
        ACCEPT 2,IFLAG
2       FORMAT(I1)
        IF (IFLAG .EQ. 1)GOTO 1000
        CALL SETCHR        !READ IN ASCII VERSION
        DO 10 I=1,256
        DO 10 J=1,8
        N=(I-1)*8+J        !MAPPING 2D ARRAY TO 1D
        BIT(N)=NCHARS(I,J)
 10     CONTINUE
        CALL ASSIGN(3,'CHRSET.BIN')
        M="177777          ! 2 BYTES OF 1'S
        WRITE(3)M,(BIT(N),N=1,2048),M    !WRITE THE LOT
        CALL CLOSE(3)
        GOTO 2000          !STOP RUN
C
C
1000    CALL ASSIGN(3,'CHRSET.BIN')
C       M is a dummy integer
        READ(3)M,(BIT(N),N=1,2048),M     !READ THE LOT
        CALL CLOSE(3)
        CALL ASSIGN (1,'CHRSET.DAT')
        DO 1010 I=1,256
        DO 1010 J=1,8
        N=(I-1)*8+J        !2D -> 1D MAP
C       A MACRO-11 SUBROUTINE TO COPY A BYTE TO A WORD NO SIGN EXTEND
        CALL BYTWRD(BIT(N),NCHARS(I,J))
 1010   CONTINUE
C       NOW WRITE TO ASCII FILE
        DO 1140 ICHRCT=1,256
        DO 1140 IROWCT=1,8
        INDEX=ICHRCT*10+IROWCT
        WRITE(1,1110)NCHARS(ICHRCT,IROWCT),INDEX
 1110   FORMAT(1X,I6,2X,I6)
 1140   CONTINUE
        CALL CLOSE(1)
C
        GOTO 2000          !STOP RUN
 2000   STOP
        END
C
C
C
```

C

```
C
C       Now follow some other routines which had their uses as fixes
C
C       INVERT
C       READS FROM 'CHRSET.BIN'
C       TO REVERSE BIT PATTERNS TO REVERSE EVERY CHAR. (CALLS A MACRO PROG)
        BYTE BIN(2048)
        CALL ASSIGN(3,'CHRSET.BIN')
        READ(3)M,(BIN(I),I=1,2048),M      !READ THE LOT
        CALL CLOSE(3)
        DO 10 I=1,2048
        CALL INV(BIN(I))
 10     CONTINUE
        M="177777         ! ALL SET
        CALL ASSIGN(3,'CHRSET.REV')
        WRITE(3)M,(BIN(I),I=1,2048),M     !WRITE THE LOT
        CALL CLOSE(3)
        STOP
        END


;       Now the two MACRO-11 programs

;       COPY A BYTE TO A WORD (BYTWRD)
        .TITLE BYTWRD   ;COPY A BYTE TO A WORD NO SIGN EXTEND
BYTWRD::        MOV(R5)+,R0     ;RUN PAST NO ARGS
        MOVB @(R5)+,R1  ;GET BYTE
        MOV R1,R2       ;MOVE TO OUTPUT WORD
        BIC #177400,R2  ;REMOVE TOP HALF
        MOV R2,@(R5)    ;RETURN WORD
        RTS PC
        .END


;       REVERSE THE ORDER OF BITS IN A BYTE (INV.MAC)
        .TITLE INV
INV::   MOV (R5)+,R0    ; RUN PAST NO. ARGS.
        MOVB @(R5),R1   ;COPY THE BYTE
        MOV #8.,R0      ; SET COUNTER
LOOP:   ASRB R1         ; LSB OFF TO CARRY
        ROLB R2         ; CARRY TO LSB HENCE REVERSE ORDER
        SOB R0,LOOP     ;8 TIMES
        MOVB R2,@(R5)   ; PUT BACK
        RTS PC
        .END
```

```
C
C
C       MAK44S  Make a screen width 44 screen
C       Program reads the screen format from either terminal or file using
C       characters " ","0"-"9",".","?","*","-","B" for block and "A" for
C       arrow. The arrow information is also read from either terminal
C       or file. Other programs not included read real data from disk and
C       write the screen format and arrow data in required format. The
C       result of running these programs is to produce a PLOT file that
C       is understood by PLTALL and was used to produce all the screen plots.
C
C
        BYTE NAME(32)
        BYTE INLIN(44),N0,N1,N2,N3,N4,N5,N6,N7,
        1N8,N9,BLOCK,HYPHEN,ASTERX,DOT,QUEST,ARROW,EXIT
C       command line from tty
        DATA N0,N1,N2,N3,N4,N5,N6,N7,N8,
        1N9/'0','1','2','3','4','5','6','7','8','9'/
        DATA BLOCK,HYPHEN,ASTERX,DOT,QUEST,
        1ARROW,EXIT/'B','-','*','.','?','A','E'/
C Characters possible on screen "ARROW" is further decoded
        WRITE(5,1)
1       FORMAT(' ARROW DATA FROM: (TI:  OR FILE.EXT)?'$)
        READ(5,1020)INLIN
        CALL ASSIGN(2,INLIN)
        WRITE(5,5)
        READ(5,1020)INLIN
5       FORMAT(' Screen data from TI: or FILE.EXT? '$)
        CALL ASSIGN(4,INLIN)
        CALL ASSIGN(3,'PLT.DAT')
C       Plotter characteristics
        WRITE(3,1000)0.0,11.0,0.0,8.0    !LIMIT
        WRITE(3,1000)1.0,10.0,1.0,7.0    !LOCATE
        WRITE(3,1000)0.0,460.0,150.0,460.0      !USER UNITS
        DO 100 LINE = 1,44        ! 44x44 screen is hardwired into prog
        WRITE (5,10)LINE
        WRITE (5,11)
10      FORMAT(' Line number',I5)
11      FORMAT(' 12345678901234567890123456789012345678901234')
        READ(4,1020)INLIN
        IARCT=0              ! Arrow count
        DO 100 I=1,44
        ICH=251 !DEFAULT
        IF (INLIN(I) .EQ. N0) ICH=241
        IF (INLIN(I) .EQ. N1) ICH=242
        IF (INLIN(I) .EQ. N2) ICH=243
        IF (INLIN(I) .EQ. N3) ICH=244
        IF (INLIN(I) .EQ. N4) ICH=245
        IF (INLIN(I) .EQ. N5) ICH=246
        IF (INLIN(I) .EQ. N6) ICH=247
        IF (INLIN(I) .EQ. N7) ICH=248
        IF (INLIN(I) .EQ. N8) ICH=249
        IF (INLIN(I) .EQ. N9) ICH=250
        IF (INLIN(I) .EQ. BLOCK) ICH=252
        IF (INLIN(I) .EQ. HYPHEN) ICH=253
        IF (INLIN(I) .EQ. ASTERX) ICH=254
        IF (INLIN(I) .EQ. DOT) ICH=255
        IF (INLIN(I) .EQ. QUEST) ICH=256
        IF (INLIN(I) .EQ. EXIT) GOTO 500        !EXIT ="E"
```

```
C
      IF (INLIN(I) .NE. ARROW) GOTO 90
      IARCT=IARCT+1
      WRITE(5,20)IARCT
20    FORMAT(' Arrow',I5,'Amplit. , Phase(0-15) '$)
      READ(2,1030) IAMP,IPHAS
      ICH=IAMP*16 + IPHAS +1
90    CONTINUE
      LINFT=440-LINE*10          ! line one at top instead of bottom
      WRITE(3,1010)ICH,I*10,LINFT
100   CONTINUE
500   WRITE(3,1010)-1,0,0       !EOF
      CALL CLOSE(3)
1000  FORMAT(4F16.8)
1020  FORMAT(44A1)
1010  FORMAT(3I8)
1030  FORMAT(2I5)
      STOP
      END
```