Defence Science Journal, Vol 44, No 2, April 1994, pp 147-159

PC-Based Display for Sonar Applications : A Design Experience

G. Miny, Anney Morris, M.A. Jayendran and A. Unnikrishnan
Naval Physical & Oceanographic Labotatory, Cochin-682 004

ABSTRACT

An attempt was made to standardise the hardware and the software used for the sonar display systems. The idea of standardise hardware suggested itself in using an IBM-compatible PC AT which configures with an Intel 80386 CPU and a standard VGA display adapter card. The software consists of Intel assembly procedures embedded in a PASCAL (turbo 4.0) environment.

1. INTRODUCTION

Sonars find extensive applications in anti-submarine warfare. Using the acoustic energy radiated by a surface or submerged target, a sonar system detects, tracks and classifies targets in real time. A sonar converts acoustic signals to electrical signals, processes the electrical signals employing signal-processing techniques and finally presents the results as visual displays (in addition to the aural presentations).

Sonar display systems usually show the output from digital signal processors in various formats, spread over many pages. The types of information to be displyed are[1] : (i) energy of all the beams to aid detection; and (ii) spectral distribution of a few bearings to aid classification.

In the case of non-PPI displays, the display format is usually of two types[1,2] : (i) ASCAN, which shows the amplitude verses bearing (or frequency if the information is spectral). Every update overwrites the previous one and is useful in picking the significant peaks, and (ii) WATERFALL, which shows the time history of many previous updates. The waterfall format keeps the most relevant information on the topmost line of the display window, while the previous information displayed will be pushed one line below at every update. The amplitude is encoded as the pixel intensity. with the bearing (or frequency) being shown along the X-axis. The format thus helps refer to the history of information displayed, because over a time span, the updates are available in the different levels of waterfall.

Usually, the ASCAN update is shown just above the waterfall updates[2]. While the energy information is derived from a direct energy computation averaged over a time constant, the spectral information is as a consequence of an FFT computation for every beam in various sub-bands of a broadband of frequencies. If there are b bands over which the spectrum is computed, B beams and p points per spectral band, then every update would require pbB locations of memory. This number could multiply to an objectionably large figure if the waterfall history is also required to be stored. It is therefore imperative that the display be organised into various pages and windows within a page for getting related information[3].

The display system deals with the entire information in different pages and each page is again classified into independent windows. While flipping through pages one should necessarily take care of the information on other pages. This is required because all the pages are active, although one page at a time is visible. That is, the data from the post-signal processor may come to any of the

Received 2 November 1992

windows (in any page) at any time. If this occurs, then the most relevant data should be received and properly channelled to the corresponding window[3].

The sonar display design should put out all formats in various pages and window and viewports within a page. In addition to the display of information it should be possible to control the display system to : (i) scan through all the formats, (ii) select a page to be displayed, (iii) read out the value of information displayed by a free cursor, (iv) designate track around a bearing, (v) modify the parameters like threshold, time constant, beam selected for narrowband analysis, etc., and (vi) grab any of the displayed pages in secondary memory.

The display design should also realize suitable hardware interface to the external hardware to collect the information for the various pages and windows.

The sections that follow discuss how the requirements for a sonar display mentioned above are realized on an IBM-compatible PC/AT with a VGA analog monitor. Section 2 summarizes the salient features of the design realized, while Section 3 elaborates the realizations of specific functions on the VGA card[4], and summarizes the primitives realized. Section 4 examines the design of the display system, defining the data structures used at the higher level and explaining the interfaces to the primitives. The hardware interface of the PC to an external hardware to collect the data required for various windows is discussed in Section 5. The last section lists some typical performance figures, discusses the limitations of the design and the suggest the avenues to be explored to enhance the design.

2 MAIN FEATURES OF THE DESIGN

There are three pages in the display system : (a) broadband page, (b) narrowband page and (c) hidden page.

2.1 Broadband Page

The broadband page is divided mainly into two windows.

The energies of all the beams are handled in one window. This window has two viewports. The data are displayed in the form of ASCAN (amplitude form) in one viewport and as colour levels (16 colours) in another viewport. The data shown in colour levels build a waterfall.

In the broadband page, the enegy window is defined at the left of the screen (Fig. 1) and the corresponding ASCAN is displayed just above the window. The cursor in this window is positioned at the top border of the window and has only a horizontal movement. The broadband window takes 256 pixels/update, can grow up to 250 lines in waterfall, and is saved while the broadband page is turned into any other page.

In the broadband page itself, a second window on the right side has the window space further subdivided into eight viewports. This window, which displays the data in ASCAN, gives the spectrum of all bands corresponding to eight consecutive beams in one format. The data in each band for the beam are proportionately compressed to 320 points and displayed. A free cursor provided here can move across the band (horizontally) and between beams (vertically). The cursor is used to select the beam and band for the narrowband display (second page). The spectrum has yet another display formate, called waterfall-on-spectrum, in which a selected set of four beams (out of the eight selected for ASCAN display) can be viewed both in ASCAN form and in waterfall form (Fig.2). These two formats in the spectrum window can be toggled using a softkey. If the waterfall on spectrum is switched on, then while turning the broadband page the waterfall information for the spectrum is also saved to the hidden page. The waterfall-on-spectrum has been provided to facilitate a cursory glance at the compressed spectral data, before ones goes into a detailed narrowband observation. The operator positions a cursor on any of the beams in any band and designates as many as eight beams at a time for narrowband analysis.

2.2 Narrowband Page

The Second (narrowband) page consists of eight windows and each window displays data from different beams/bands in waterfall format (Fig. 3). Each window keeps 30 instants' previous data, with 640 points/instance. These windows can be independently dropped and reused by means of the softkeys provided. A free cursor moves across the windows and in each window the cursor moves horizontally along the top border of the window. The spectrum readout corresponding to the cursor position is given separately in each window. These windows take 640 data at a time for all bands. That is, each band gets stretched to 64 data in a narrowband.

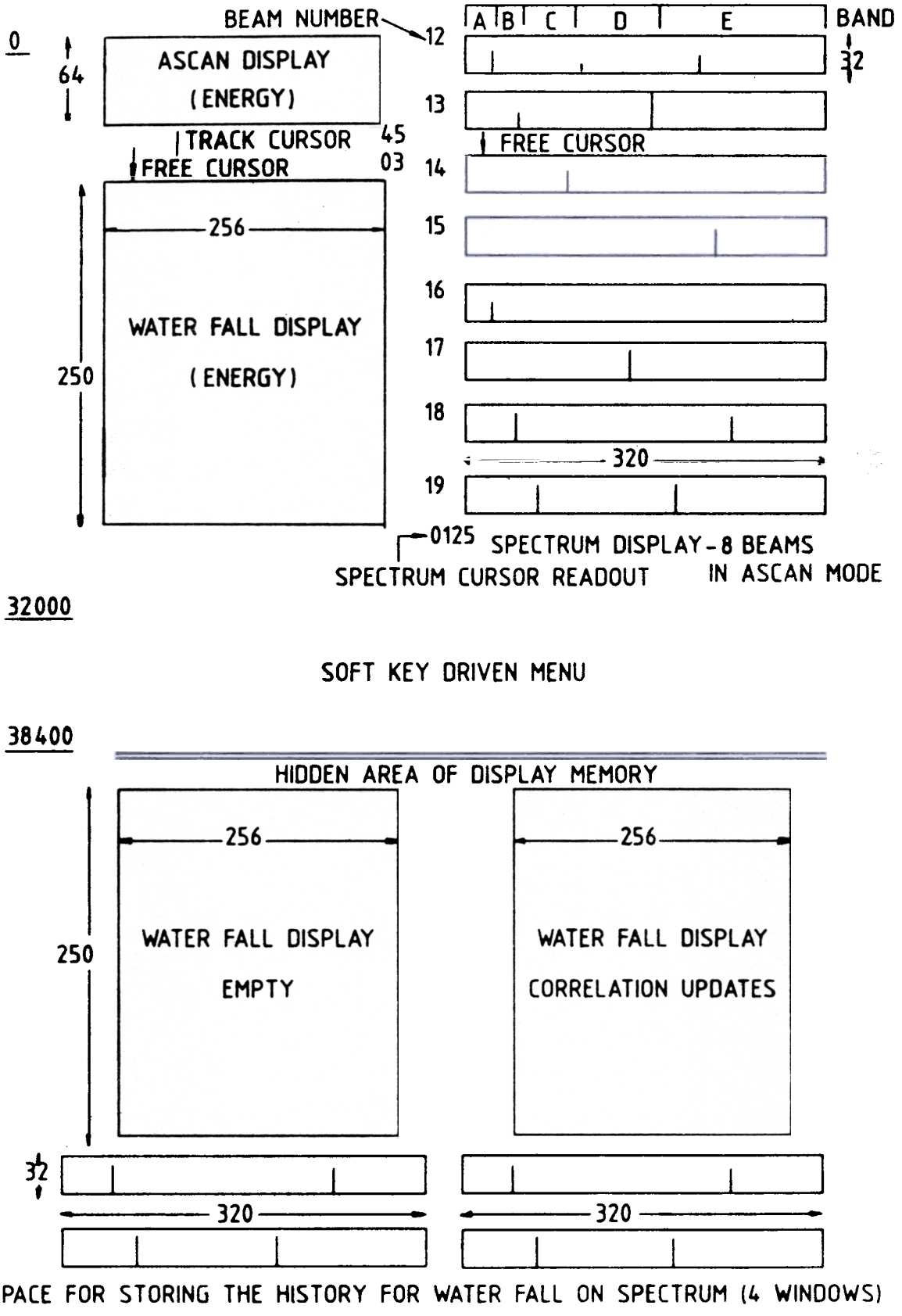
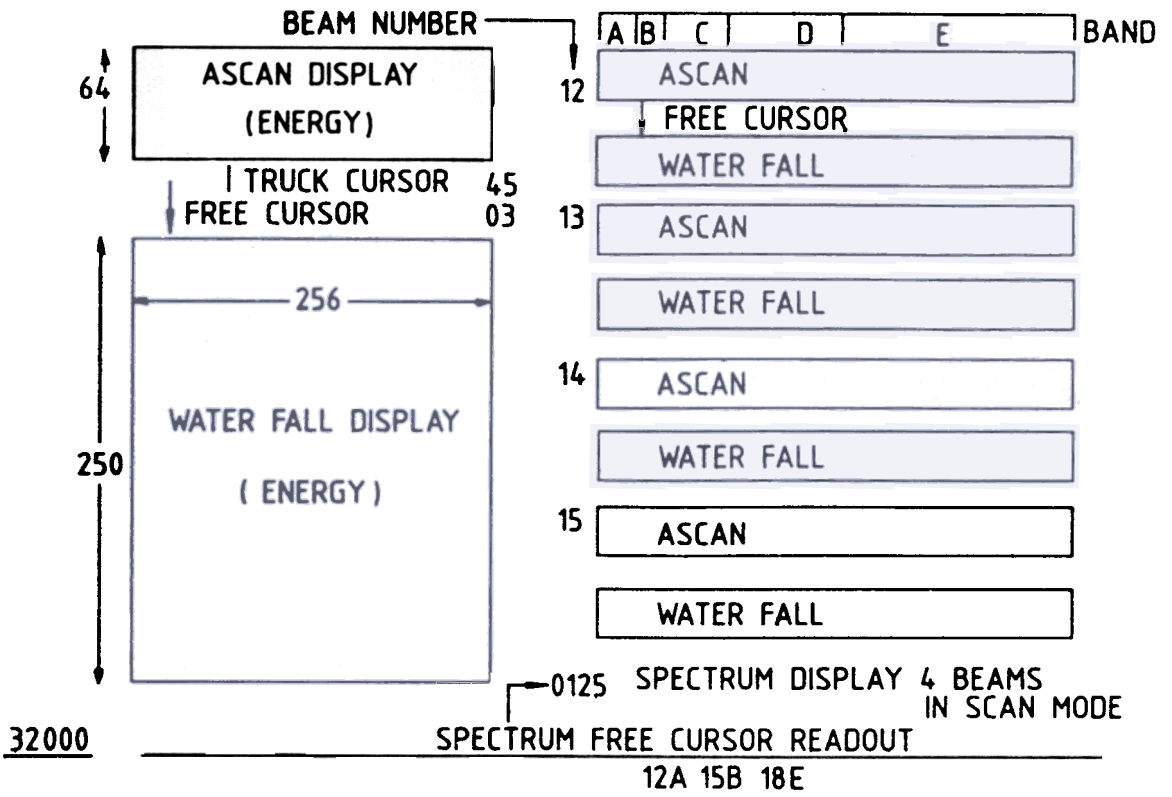
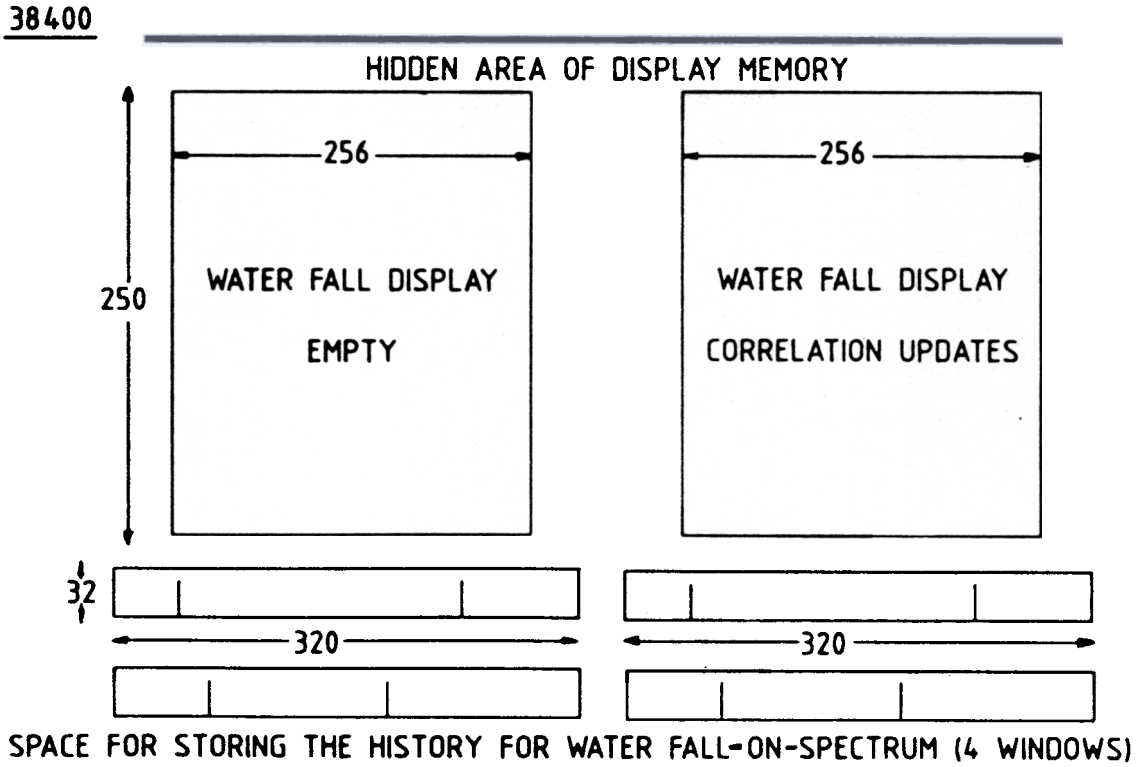


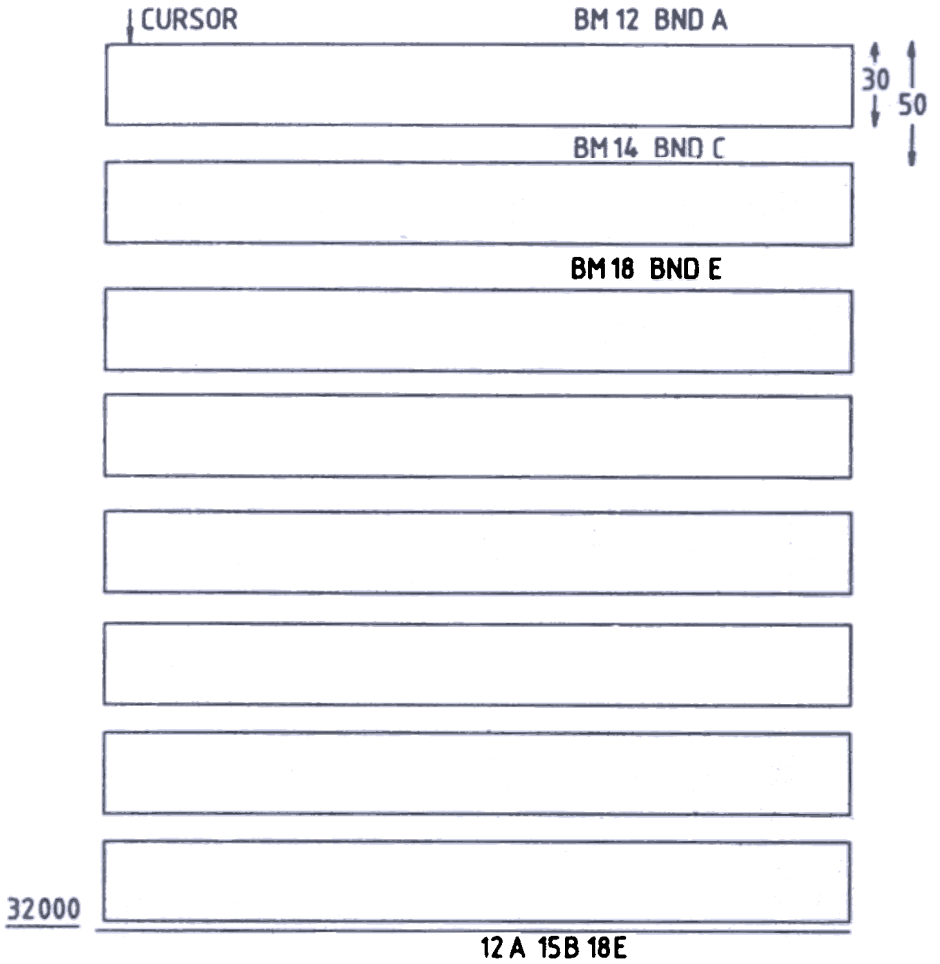
Figure 1. Broadband page



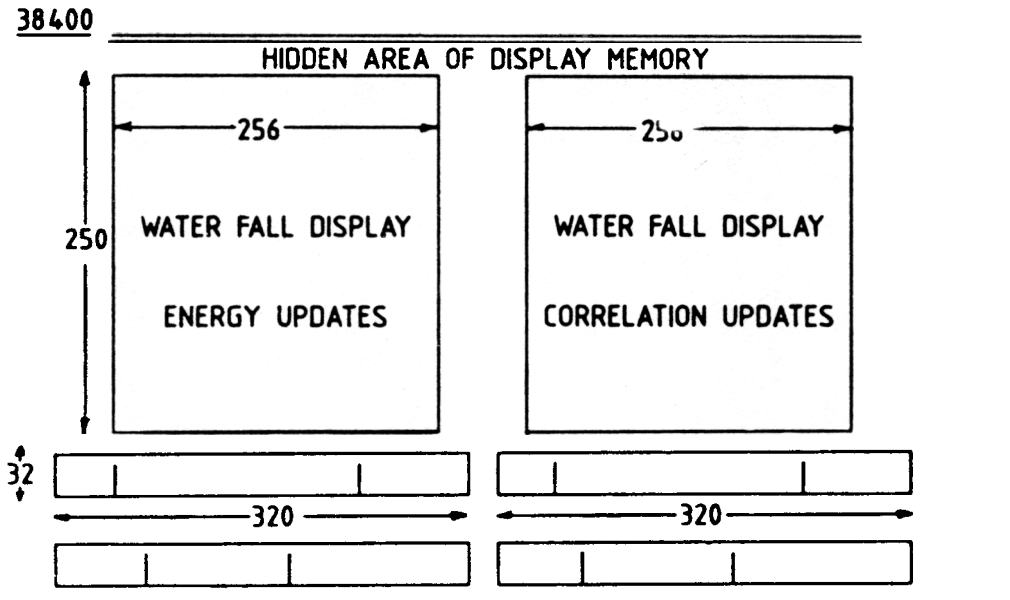
SOFT KEY DRIVEN MENU



SPACE FOR STORING THE HISTORY FOR WATER FALL-ON-SPECTRUM (4 WINDOWS)



SOFT KEY DRIVEN MENU



SPACE FOR STORING THE HISTORY FOR WATER FALL ON SPECTRUM (4 WINDOWS)

2.3 Hidden Page

While most of viewable information is kept in the display memory in 80×640 byte locations starting at $0A0000, a hidden page is established in the hidden region of the display memory from the address location 38400 (relative to $0A0000) over 25600 bytes. While flipping from the broadband page, the entire broadband page is retained in this hidden page; so this page is used as an-active reserve with the update taking place in the hidden space also. The attraction of using such a page (in the hidden region) instead of the main memory for keeping the data is that it avoids the necessity to rebuild the entire broadband display in four planes when required again. The display system takes the correlator updates also in a window similar in size to that of energy information in the hidden page. The correlator updates (on hidden page) can be swapped with the energy updates at a fast rate across all planes of display.

The hidden page therefore has three windows. One window always holds data coming from the correlator and another is kept for keeping the enegy data when the broadband page is turned. The third window keeps the spectrum updates of the broadband page switches with the waterfall-on-spectrum formate on.

The hidden page is used to hold the data when the broadband page is turned. The bulk of the narrowband data is held in the main memory dynamically in a rolling buffer. The entire data in the eight windows in the narrowband are kept in RAM. Even when the narrowband is made visible on the screen, the data get duplicated in RAM.

A third window, which is common for all pages (except in hidden page), is the softkey-driven menu window—a text window that handles parameters for post-processor and display. This window will be receptive to the intervention of the operator, who realizes the control functiones outlined in Sec. 1 through a hierarchically presented menu and function keys of the IBM-compatible keyboard. The relevant parameters (whether operator-selected or default) are displayed on the screen. The display helps in knowing the environment of the signal processor specifically.

Each window has a cursor and the cursor control passes from window to window and from page to page. At a time the cursor will be active only in one window. A softkey is provided for toggling the cursor control whenever necessary. The cursor within the active window is slaved to the arrow key of the IBM-compatible keyboard.

3. DESIGN DETAILS

The entire display is done in the VGA graphics environment[4], which allows window manipulations like : (i) illuminating pixels, (ii) scrolling the updates to realize waterfall effect, (iii) moving the display data across all four planes for fast switching, (iv) providing alphanumeric annotations, and (v) frame grabbing.

3.1 Main Features of the VGA Controller[4]

The software takes advantage of the VGA resolution 640×480 (in a 14-inch monitor) and four-bit planes with a display memory of 256 K and starting at address 0A0000(hex). Each consecutive set of 8 pixels in a row is illuminated by addressing successive byte locations starting at 0A0000(hex). Therefore, though there are 256 kB of storage making the display memory, because of the organization in four-bit planes, the directly addressable region spans over 64 kB only.

The visible region of the display memory takes up about 150 kB of storage. The rest, 106 kB, can be used for the hidden page given a span of about 640 pixels per scan line and 320 scan lines. This is the VGA 12(hex) mode and is set by giving the mode number [in this case 12(hex)] in the AX register and invoking the ROM BIOS interrupt 10(hex). Setting the mode clears the screen also.

Figure 4 shows the organization of the VGA system in relation to the display memory. Most of the display functions are controlled through a set of registers grouped into : (a) sequence registers, (b) graphic controller registers, and (c) attribute controller registers.

Most of the VGA registers are set directly through the assembly programs, and these registers are accessed through ports. These registers are eight-bit wide and are segmented into one to eight fields. The indirect addressing technique is used to access the registers in each set. For this, two I/O port locations, are used. The address register is used to point to one of the actual VGA registers within a selected group and the value written in the data register by the host goes directly to the selected register.

3.1.1 The Sequence Registers

The sequencer register set consists of six registers. All these registers are accessed through two registers.

viz sequencer index register with port address 3c4(hex) and sequencer data register with host port address 3c5(hex). The map mask register, one among the sequencer registers, is used here to set (enable/disable) the four-bit planes of an addressed byte. The index 2 selecting map mask register is moved into port 3c4(hex) and the bit plane pattern is moved into the least significant four-bits of the data register at port 3c5(hex), thus setting the colour for the addressed byte or pixel.

3.1.2 The Graphic Controller Registers

The address applied is to a whole byte (consisting of 8 pixels). Therefore, whatever the colour set to this applied address it is attained by all pixels in that byte. To access a pixel or a group of pixels within a byte without disturbing the contents of the other pixels in that group. It is necessary to mask the undesired pixels. To do so, the bit mask register (one of the graphics controller registers) is used. The graphics controller register group contains 9 registers along with the bit mask register. These registers are indexed from 0 to 8 and the index register port is 3ce(hex) and the data register port address is 3cf(hex). The undesired pixels can be masked from the write operation by setting the corresponding bits to 0 in the bit mask register, which is indexed at 08(hex).

The mode register, one among the graphics controller registers, handles specific EGA/VGA write modes to facilitate : (i) illuminating a pixel, and (ii) moving the displayed data fast across the 32-bit bus of the display memory.

The mode register, idexed at 05(hex), has a field to select three write modes. Of these modes, mode 0, which is the default mode, permits the operator to

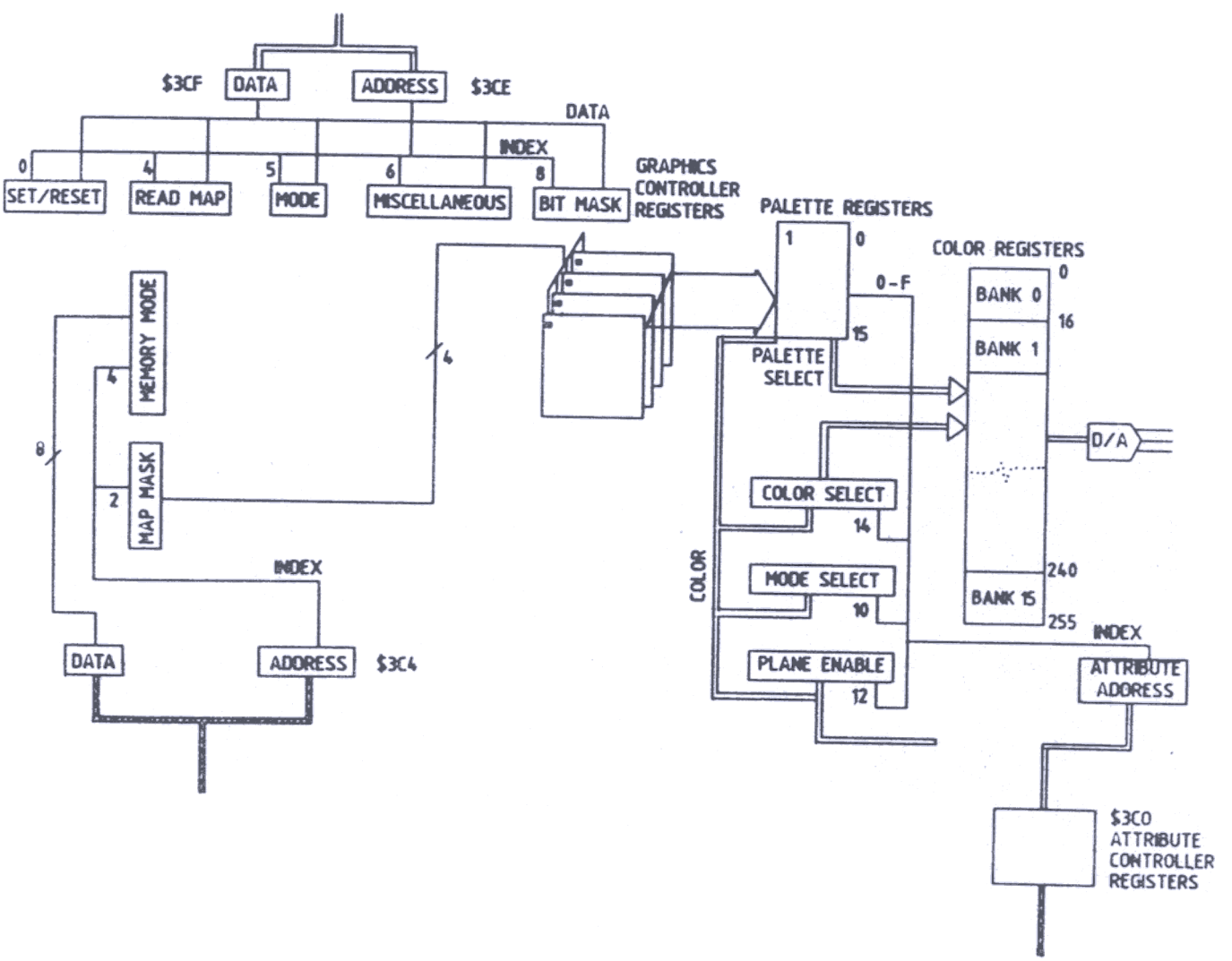


Figure 4. VGA organisation.

illuminate pixel by ensuring the colour information in the map mask register and pixel position in the bit map register. The write mode 1 facilitates fast transfer of displayed data across the 32-bit bus of the display memory. This mode thus helps one to swap two pages (across the viewable portion and the hidden portion) or to push a line of displayed data to realize a scrolling effect.

The mode register also has a field, which, when set, permits one to read the contents of the VGA memory locations. The plane number to be read is given in the read map select register available in the graphics controller chip, indexed 04(hex). The read mode helps the operator to read the display memory as is required as part of a frame-grabbing operation.

3.1.3 Attribute Controller Registers

The registers of this set are used to control the static attributes of the display like palette controls, colour register group selected, etc. These static attributes help switch from colour to black and white and also various colour combinations. The attribute registers reside at host port addresses 3c0(hex) in such a way that alternate read and write operations toggle the data path to the attribute address register and the selected attribute register. In Fig.4 the contents of the palettes in the VGA address a set of colour select registers, so that the output of the latter decides the actual colour to be given to the addressed pixel. (However in the EGA, the output of the palettes directly decides the colour to be assigned to the illuminated pixel.) Bits c45 and c65 of the colour select register, indexed at 14 hex, decide which one of the 16 banks is to be selected for deciding the colour to the addressed pixel. The present design selects bank 0 (which is the EGA colour emulation on VGA) for colour display and bank 1 is the monochrome multilevel intensity display. When bank 1 is selected, the colours selected are summed up to a gray level with the BIOS call :

AH		
10(hex)	AL=1B hex	sum colour values immediately to gray scale
12(hex)	AL=33 hex	enable/disable summing to gray scale at mode set.

The mode control register, indexed at 10(hex), decides the mode of operation (alphanumeric/graphic) and type colour register grouping – (i) 16 groups, each group having 16 registers or (ii) 4 groups, each group having 64 registers. The 16 palette registers, indexed at 0-f(hex), can also be modified so that a colour lookup function may be implemented across the data loaded into display memory and the colour selected for the display. (This lookup function is used for realizing a smooth colour grading without adrupt variations).

3.2 Alphanumeric Annotations

Most of the alphanumeric annotations in graphic mode are handled by using BIOS calls for writing characters on display area. All these calls are invokec through the interrupt 10(hex). The parameters involvec are the string and the row and the column where the string starts. As expected, there will be a bit of overheac when using the alphanumeric BIOS calls too often. The overhead may be ignored in cases where the menu window is annotated but is considerable in cases like cursor readouts. Therefore, in the present design fo alphanumerics of cursor readouts, the characters whic are preloaded on one line of the hidden area are copie (by setting write mode one through mode register c graphics controller registers) to the visible region, thu avoiding the overhead.

3.3 Realization of Basic Primitives

This section elaborates the realization of th following basic primitives which are later used t implement the display : (i) illuminating a line of pixe from an array of bytes, (ii) scrolling a rectangular patc of illuminated pixels to realize a waterfall effect, (ii generating the ASCAN display from an array of byte (iv) making a cursor of a defined pattern, (v) annota an addressed location (addressed by row number ar column number) with the charaters in a string, and (v grab a line of pixels from various lines.

All the primitives are declared in pascal as exterr procedures and realise in assembly language, taking t parameters passed from the stack location [SP(st pointer) + 4] onwards. Table 1 lists all the realiz primitives.

(i) The procedure, as defined in pascal belo illuminates a line of pixels :

Procedure DSPLY2W(varbuff:bufftype;

start_addr:**word**;count:**integ**

The procedure writes the pixel in write mode starting at the byte addressed by **start_addr**. T

parameter count indicates the number of pixels to be illuminated. The procedure uses the bit mask register to select individual pixels in a byte. The colour information contained in the array of bytes buff is assigned to the corresponding pixels using the map mask register.

Table 1 Summary of primitives

DSPLY2W **arguments :** **buff:bufftype,** **start_addr:word,** **count:integer);**	Displays a set of pixels with the colour information given in array buff
SCRLL2W **arguments :** **start_addr :word,** **c_line : integer,** **count : integer);**	scrolls a set of lines
BIT64 **arguments :** **max_array:asc_array,** **curr_array:buff,** **start_addr :word,** **count : integer);**	displays the data in ASCAN format
CU_JMP **arguments :** **x:str_buf,** **curs_pos : word,** **prev_addr: word,** **strt_addr:word,** **spos : integer;**	handles all cursor manipulations
ALPHA **arguments :** **strng:string [10]** **row, colum :integer,** **colour, length:integer;**	displays text information in graphic environment
SAVE	copies a page of information from display memory into secondary storage device

(ii) Scrolling is achieved using the procedure.

Procedure SCRLL2W(start-addr :**word**: c_line, count : **integer**)

The pixels are moved across byte addresses saparated by 80 bytes by writing in the VGA write mode 1. c_ line and **count** define dimensions of the patch to be moved. This procedure can be suitably modified to swap the data displayed across two viewports.

(iii) The ASCAN display is accommodated using the procedure described below :

Procedure BIT64(**var** max_ array:asc_**array:**

var curr_ array:buff; start_addr : **word** : count : **integer**);

The arguments **max_array** and **curr_array** contain amplitude information of previous and current updates for each pixel starting from address given in the parameter **start_addr**. The integer partameter **count** contains the number of pixels in which amplitude information is displayed.

(iv) The cursors are defined using the procedure with the following format :

Procedure CU_JMP(**var** × : str_buf; curs_pos : **word**; preve_addr, strt_addr : **word**; spos : **integer**);

This procedure handles all the cursors in different windows and pages. The procedure blanks the previous position of the cursor and remakes the cursor in the new position. The parameters **prev_addr** and **start_addr** hold the previous and current positions of the cursor and parameter **spos** gives the exact pixel position of the cursor in the addressed byte. The variable **curs_pos** is the location at which the cursor position readout (contained in the parameter ×) is to be placed.

(v) Alphanumeric annotations in graphics mode are realised using the procedure :

Procedure ALPHA(**var** strng : string [10]; row, column, colour, length : **integer**);

The procedure dispalys the characters from the string array **strng** in a 16 × 8 matrix at the location addressed by row and column with the specified colour, using the BIOS call explained in sec. 3.2. The argument length specifies the number of characters in the string.

(vi) A full page of the display memory (usually from visible area) is grabbed to a secondary storage devices, using the procedure :

Procedure SAVE :

The procedure takes the data from the four bit planes of each byte and stores it in a RAM buffer. A Pascal routine then transfers the buffer into secondary storage device. An assembly procedure sets the mode register from among the graphic control registers in read mode 0, and by selecting each bit plane through the read map select register, the colour codes in each byte are transferred into the RAM buffer.

4. DISPLAY DESIGN

The design of a typical display system inyolves writing higher level programs to fill data into various

swindow in each page, using the primitives outlined in Section 3. As observed in Section 1, a typical sonar display should display information in (i) energy window to aid detection, and (ii) spectrum window to display the spectral information from a few selected beams, to aid classification.

The energy window is defined as :

```
lines_per_window = 250; max_no_bytes = 32
window_desc = record
                start_addr  word
                curr_line_no
                max_no_line : 0 .. lines_per_
                              window;
                max_array : array [0 .. max_no_
                bytes] of byte;
                end;    end;
```

The **start_addr** locates the window in row and column at the offset address specified in start_addr. The absolute address in VGA is then A0000h:start_addr. The start_addr takes a value respectively between 0 and 38400(dec) and between 38400 (dec) and 64000(dec), to position the window in the visible area and hidden area. (In a typical design meant to display the output from an energy processor in the visible area and correlation processor in the hidden area, the start_addr was initialised to 642(dec) and 39762(dec) respectively. A modified version of the procedure SCRLL2W makes it possible to swap displayed the information fast, in waterfall formate, in between the visible and hidden areas of displayed memory). The **curr_line_no** indicates the depth of the waterfall history and is incremented by one on each update and is never allowed to grow beyond **max_no_lines**.

In a typical design, upon receipt of a buffer of intensity values destined for a window [buffer length = 8 * **max_no_bytes** and an intensity value which lies between 0 and f(hex)] the procedure SCRLL2W is invoked by passing **curr_line_no** also as a parameter. The SCRLL2W procedure moves the existing data one line down. The topmost line of the window is now free for display. If **curr_line_no** = max_no_lines, the earliest data (bottommost line of the window) gets overwritten. Then the procedure DSPLY2W [(Sec. 3.3(i)] is invoked to display the current data on the top line of the window, thus building a waterfall. The variable **curr_line_no** gets incremented each time and steadies when it reaches **max_no_lines**.

The same data received are displayed in amplitude form also just above the waterfall window. The **max_array** stores the max value in a group of eight pixel intensities (one byte) to facilitate the bargraph display of the ASCAN information [(Section 3 (iii)].

The spectrum window displays the spectral information of selected eight consecutive beams in ASCAN format. The information per beam covers all processing bands thus making the resolution low and still facilitating a cursory glance over the spectral distribution in different beams.

Each beam description is kept as follow :

```
max beam    = 8;        fft_count = 40;
 spec_desc    record
              beam       byte;
              spec_addr  : word;
              spec_max   : array (0..fft_count) of
              byte;
              curr_line_no : integer;
              end;
```

The **spec_addr** gives the location where the spectral information of a particular beam (specified by the parameter beam) is to be displayed. The absolute address is then calculated as A0000h:spec_addr where A0000h is the VGA display memory start address. The beams for which the spectral information is displayed are selected from the energy window. The **spec_max** array keeps the previous update information to clear each byte for current update.

Since eight beams can be selected for spectral display, the records are kept in an array.

spec_desc_array = **array**(1..maxbeam) of spec_desc.

The data received at an instant will be (fft_count * 8) per beam thus giving a total of (fft_count * 8 * max_beam) spectral information.

The spectrum window can be reconfigured to display four selected beams in ASCAN format as well a waterfall format. The **curr_line_no** is used here to mark the depth of the waterfall as in energy data display.

As mentioned earlier, the narrowband information of eight selected beam/bands is displayed in the second page as eight window. This feature makes it possible to have a finer view of the computed spectrum, as a single

band of a given beam stretched across a full line of display.

Each narrowband window is defined as :

```
nb_desc = record
    beam, band                  : byte;
    start_addr                  : word;
    cur_line_no, nb_curr_line   : integer;
    buf_full, update            : boolean;
    end;
```

The **start_addr** locates the window in row and column. In other words, **start_addr** is the offset address of each window from the display memory start address. The beam and band parameters work as a reference for the window. **curr_line_no**, as in the case of energy window, gives the depth of the waterfall in each window.

Cursor positions in broadband and narrowband pages are updated using the procedure CU_JMP [(Sec. 3.3.(iv)]. A global variable **active_ctr l** taking the values (left,right,nb) decides whether the cursor is active in energy window, spectrum window (both in broadband page) or narrowband window (in narrowband page). In order to ensure that the cursor is independent in each window, three instances of the cursor definition, given below, are kept in an array indexed by **active_ctr l :**

```
curs_desc = record
        start_addr,prev_addr : word;
        limit                : word;
        pos                  : integer;
        end;
```

The **start_addr** and **prev_addr** respectively give the byte positions of the cursor display memory at the current update and previous update while limit gives the bound for cursor movements. **Pos** indicates the actual cursor readout. **Prev_addr** is used to blank the previous cursor position following an update. The **active_ctr l** ensured to take appropriate values, as the active display switches between the narrowband/ broadband pages, in order to resume proper control of the cursor in the page selected for display.

5. HARDWARE INTERFACE TO EXTERNAL PROCESSOR[5]

In a typical design, the data destined for various windows and pages come from a hardware external to PC. Also, the data meant for energy, spectrum and narrowband pages come from independent processors, each raising a flag to mark the availability of data. The PC interface hardware therefore uses an encoder to identify the interrupting external processor. The encoder output interrups the PC on the 10th interrupt vector line available in PC bus and the encoded value is read by the interrupt service routine from the IO port 310(hex) to choose the relevant processing option in the service routine. The processing in the relevant option includes transferring the data from the sharable memory of the external processor to the appropriate buffer, clearing the external interrupt by pulsing a flag, and finally raising a software flag to the background display program. Since the different external processors were bussed together on the data and address line, the host processors were mapped to 4 KB address space from d000:0000(hex). The transfer of data could be through DMA or by reading under program control.

Figure 5 shows the block schematic of the hardware interface. The other possible option of mapping the external memory to extended address space was not attempted for the design realised. Also, in the design realised, the processing parameters, destined for the external processors and collected during the interactive sessions with operator, are flushed out through the IO port 304(hex), just before servicing any interrupt.

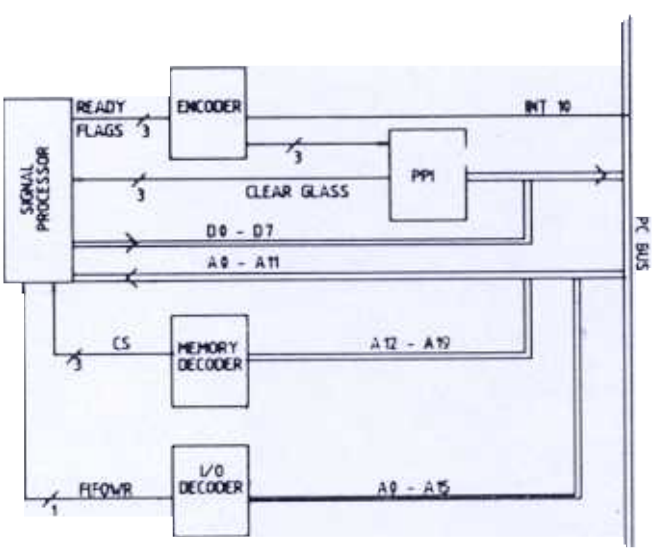


Figure 5. Hardware interface.

6. CONCLUSION

The display system realised, as discussed in the previous sections, was interface to signal processors,

by illuminating individual pixels. Because of the requirement to illuminate individual pixels, the tranfer typically took 340 ms per narrowband window, with the waterfall history full. During the narrowband display, updates were carried out both in the RAM area and in the display memory.

computing spectrum, energy and correlation data. Figures 6-8 show typical snapshots of display of the broadband and narrowband pages. Fig. 6, read along with Fig. 1, shows the broadband page displayed after nearly 200 updates. The Peak in the ASCAN window of the energy display (refer to Fig. 1 also) is shown as an intensified stretch in the waterfall window, thus showing the history of nearly 200 updates. The spectral information on broadband page (right window in Fig. 6, refer to Fig. 1 also) shows the instantaneous display of 8 channels across different bands A,B,C,D and E. In Fig. 7, the spectral information in 4 channels (out of 8 channels shown in Fig. 6) is displayed in waterfall format also (refer Fig. 3). Figure 8 shows the datailed narrowband spectrum in waterfall format (refer to Fig. 2) for the bands specified for selected channels in Fig. 6.

With regard to the performance of the display system, a single update in energy window along with waterfall scrolling took 320 ms for 256 points per update. The swapping of data between hidden and visible regions of the display memory, with the entire waterfall history area completely filled, hardly took any time. Since the space available in hidden memory was not sufficient to hold the narrowband update also, the narrowband waterfall was maintained in a set of rolling buffers held in the RAM area. During a switchover to the narrowband display, the waterfall information was transferred in bulk from RAM area to display memory,

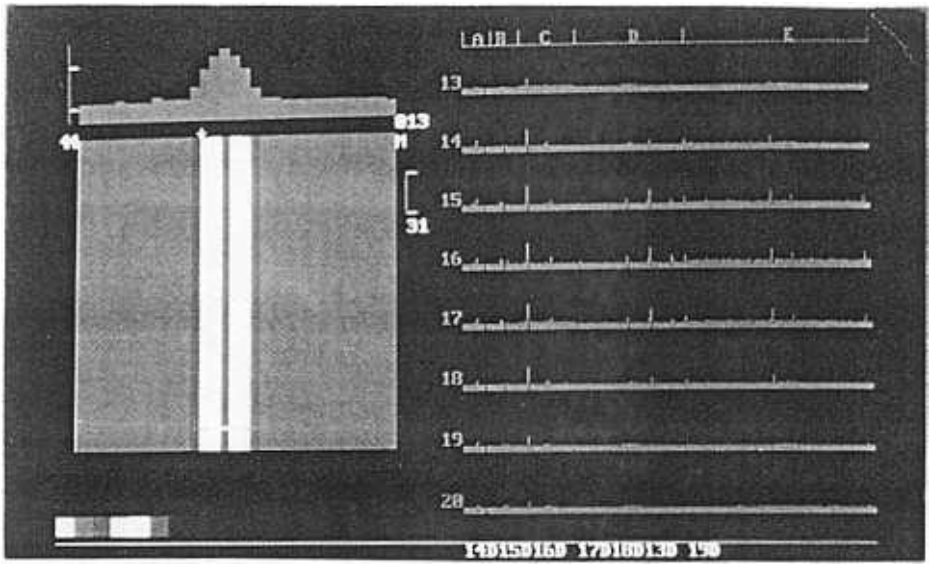


Figure 6. Broadband page.

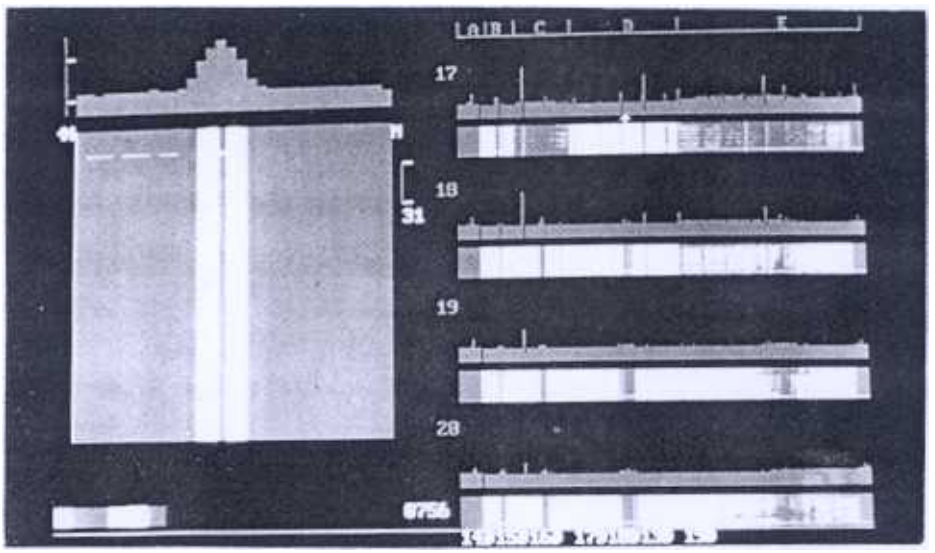


Figure 7. Broadband page with waterfall-on-spectrum.

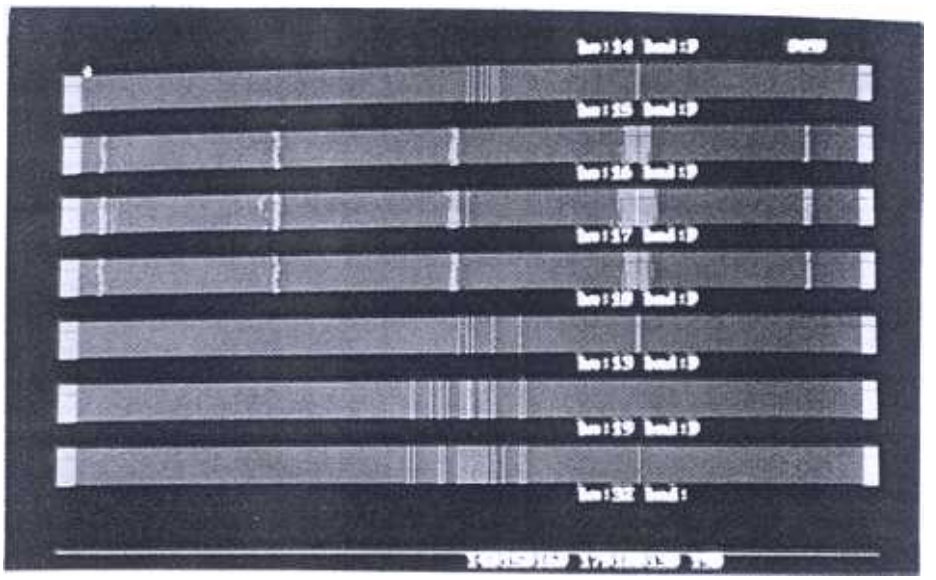


Figure 8. Narrowband page.

The updating of various windows was so sequenced that the key depressions were serviced in between the updating process of any two windows. Therefore the system responses specially relating to the cursor movements, were quite fast.

The design also provided a facility, viz. FRAME GRAB, to take a snapshot of the displayable area of memory using SAVE primitive explained in sec 3.3 (vi). The four display planes were read as consecutive bytes into an array and transferred to secondary memory as a binary file. A grab of 480 lines with 640 pixels per line was complete in less than 3 s. Grabbed pictures can be replayed, restoring a full page in 3 s.

Further improvement in design could incorporate better resolution on a 19 inch monitor with the super-VGA controller card. Grabbing time also can be reduced by temporarily storing the grabbed data in the extended memory. Advanced display devices incorporating LCD and plasma displays could also help ensure higher reliability[6].

ACKNOWLEDGEMENTS

The authors would like to thank the Director, National Physical & Oceanographic Laboratory, Cochin, and Ministry of Defence, Government of India, for giving permission to submit the paper for publication. Immense gratitude is also due to their senior colleagues Dr. V.K. Aatre, Shri V. Chander, Shri R.C. Agarwal and Shri B. Lalmohan for their constructive criticisms and helpful suggestions throughout the work.

REFERENCES

Delisle, R.B. & Kroenert, J.T. An analytical model for the detection performance of multiple channel time-history display formats, *J. Accous. Soc. Am.*, 1983, **73**(6). 2065-70.

HS 312 TSM 8241, Technical Book, 4/417, Feb 1991 Thomson Sintra.

3. Oldham, S.J. Human interaction with next generation sonar systems, FCSL, Birdhall Lane, Cheadle Heath, Stockport, SK3 OXQ.
4. Ferraro, R.F. P: ɔgrammer's guide to the EGA and VGA cards, Addison Wesley, 1988.
5. Personal computer hardware technical reference library AT.
6. Display Device 1990, No.2 — Electronic display device and application technology, Dempa Publications, Inc.,1990.