Video Digitizer Using VHDL

Project Report 1998-99

Submitted by

ANANTH .R
GURUMOORTHY .G
NIRMAL KUMAR .V
SENTHIL .G
SHIVAKUMAR .B

Guided by

Asst Prof. S. GOVINDARAJU M.E,

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF **BACHELOR OF ENGINEERING IN ELECTRONICS & COMMUNICATION ENGINEERING** OF THE BHARATHIAR UNIVERSITY COIMBATORE

Department of Electronics & Communication Engineering
Kumaraguru College of Technology
Coimbatore-641 006

CONTENTS

ACKNOWLEDGEMENT

It is our foremost duty to thank our Principal, **Mr.K.K.Padmanabhan**, B.Sc.(Engg.), M.Tech., Ph.D., for his kind patronage. We owe a lot to the management for the interest and encouragement given in making this project a success.

We are always thankful to our beloved Head of the Department, **Prof. M.RAMASAMY** M.E, whose consistent support helped us a great deal and his enthusiastic involvement needs special mention. We are greatly indebted to our beloved guide **Mr. S.GOVINDARAJU**, Assistant Professor, Dept. of ECE for his invaluable guidance and timely support for the successful completion of our project. As a token of our esteem and gratitude, we honour him for his assistance towards this cause.

In this regard we would also like to thank our class advisor **Mr.K RAMPRAKASH**, Assistant Professor, Dept. of ECE for his invaluable advises.

Last but not the least , we would also like to thank the teaching and non teaching staffs for their technical assistance.

SYNOPSIS

The video digitizer for PC's is an Addon card, which is used to digitize video signal. It is designed to handle colour images and allows the T.V images on the computer screen. The card forms a perfect link with several graphic workshop programs. It is compatible with 24-bit color files.

A signal from the video source is the input to this card. The card is made out of VHDL software. This signal is digitized and fed into the computer with the help of software. The software is developed in TURBO® C language acts as the master, which drives the card. This software packs the digitized color information into a TIFF(Tagged Image File Format) file. This file can be processed further by atmost any graphics program that runs under MS-DOS ® or MS® WINDOWS.

Once the picture is digitized and brought on the PC screen, it opens new vistas of application which would be impossible with an analog image. Many commercially available software packages such as ADOBE® PHOTOSHOP, COREL®PHOTOPAINT, Z-SOFT® PHOTOCREATOR etc., makes image editing realistic. There are

numerous applications associated with this digitizer and depends on the creativity of the user.

INTRODUCTION

The call for more and better tools to integrate text,images and sound on PC's has had a new impulse with the widespread introduction of the Microsoft® Windows Graphical User Interface (GUI). While there are tremendous advantages of having an artist draw an image in freehand based on his or her imagination or trace a photograph on the bit pad, another less time-consuming approach is to use a television camera to input a live image into the computers memory. This is precisely the function of a video digitizer sometimes known as a Frame Grabber, since it captures a video images and then converts it into a series of digital values. Once the image has been digitized and stored as a bitmapped display, it can be manipulated in exactly the same way as an original image created by an artist.

The digitizer circuit is designed using VHDL synthesis software. The main purpose of using VHDL is to bring compactness and modifiability in the digitizer circuit for future expansion.

The input to the digitizer is video signal. The circuit should be designed in such a way that it co-ordinates with the input signal.

Therefore there is the necessity to understand what it is. Therefore a study on it has been made. The basic principle of the video signal and TV systems have been explained in INPUT VIDEO SIGNAL.

To digitize the video signal some basic building blocks such as ADC, splitter, counters, etc., are needed. Further the sampling of the signal should be done to match the hardware. Therefore scanning has to be done in such a way that it is reproduced intelligently. DIGITIZER BASICS deals with how actually the scanning takes place along with the basic building blocks of the circuit.

This digitizer is designed as an insertion card, which is to be connected to the peripheral slot of the PC. Therefore the hardware should be designed in a way that it is compatible with the computer's bus structure. Further some vital information from the input video signal has to be used in the process. Details regarding the PC interface, the process of digitizing, writing it to the RAM and retrieving of the stored information have been widely covered in HARDWARE DESCRIPTION.

The master to this project is the software. The insertion card's control signal and all other preset value are governed by the software. The digitized information also has to be stored in a specific format, which is achieved by the software. DIGITIZER SOFTWARE gives you the idea of the software with the help of flow charts. It also covers how the file formats (TIFF) are used to store the image along with the digitizers software.

Now comes the main question, What can be done with this digitizer? APPLICATIONS shall provide you with the answer. Here the various applications of it have been elaborated.

THE INPUT VIDEO SIGNAL

The basic input to the digitizer is the standard video signal. This video signal consists of picture information and timing information, further there are different methods of scanning such as interlaced scanning and non-interlaced scanning. The time period of the input video signal plays very important role in the process of digitizing. Since the input video signal contains the time information of the time period between each line as well as the time period between each frame, it is necessary to make a detailed study of the input video signal. This chapter deals with the details of the basic video signal.

PICTURE INFORMATION

The picture information is optional in character and may be thought of as an assemblage of a large number of tiny areas representing picture details. These elementary areas representing picture details. These elementary areas into which picture details may be broken up are known as "picture elements" or "pixels" which when viewed together represent visual information of the scene.

As an example, suppose that we want to transmit an image of black cross on a white background, at the left in figure 2.1 to the right side. The picture is divided into elementary areas of black and white, and the pixels forming the cross are black. When each picture element is transmitted to right of the figure and reproduced in the original position with its shades of black or white, the image is duplicated.

Thus at any instant there are almost an infinite number of pieces of information that need to be picked up simultaneously for transmitting picture details. However, simultaneous pick-up of all the pixels in one frame is not practicable because it is not feasible to provide a separate signal path (channel) for the signal obtained from each picture element. In practice, this problem is solved by a method known as "scanning" where conversion of optical information to electrical form is carried out element by element, one at a time and in a sequential manner to cover the entire picture.

SCANNING

Scanning is the process by which an electron beam spot is made to move across a rectangular area so as to cover it completely. This rectangular area may be the target surface in a television camera or the screen of a picture tube in a television receiver. Former is the characteristic of transmitter scanning whereas the latter represents receiver scanning.

HORIZONTAL SCANNING

The movement of electron beam spot from left to right and back so as to start a new line in the same direction is termed as horizontal scanning. For illustration see figure 2.2. Horizontal scanning frequency is defined as the number of lines scanned per second. Obviously in 625 line, 50Hz system where 25 frames are transmitted per second, the horizontal scanning rate is 15,625 lines per second. One cycle makes the spot scan from left extreme to right extreme (during trace) and back (during retrace). Figure 2.3 illustrates the process.

VERTICAL SCANNING

Vertical scanning is the movement of the electron beam spot in the vertical direction. It is clear from figure 2.3 that the spot gradually under goes atilt in the vertical direction while scanning horizontally so that when the spot retraces back to the right extreme, it does not overlap the precading line and scans a line other than the one previously scanned. If there is no vertical tilt the spot would scan the same horizontally. For illustration see figure 2.4.

The figure 2.4 clearly emphasizes the necessity of vertical scanning in addition to horizontal scanning. We can notice how in the absence of vertical scanning, the raster which would have been a rectangular area covering the entire picture tube screen is reduced to a horizontal line. On similar line, the absence of horizontal scanning with vertical scanning present would result in a single dot.

LINES PER FRAME

To get finer details of any picture the number of scanning lines for one complete picture should be large. However, factors like the need for larger channel bandwidth with increase in number of lines, smallest possible thickness of the scanning beam and limitation of resolving capability of the human eye restrict the number of lines into which picture height can be divided. Since increase in bandwidth causes limitations in number of channels that can be provided. Thus a compromise between quality and cost, total number of lines for a frame is chosen as 625 in CCIR and 525 lines in American system.

FRAMES PER SECOND

In television the still picture are captured and moved continuously at a certain speed, so that due to the persistence of vision, we feel that the picture is in motion. For attaining this level, the picture had to be moved at a rate of 25 frames/second, since in Indian system, there are 625 lines in one frame so there is a necessity to scan 15625(625 x 25) lines per second. This is the frequency at which horizontal scanning ha to be done. The time period needed for scanning one horizontal line is 64 micro seconds (1/15625), out of which active line period is 53.3 micro seconds and 10.7 micro seconds is the line blanking period. The beam return during this short interval to the extreme left side of the frame to start tracing the next line.

INTERLACED SCANNING

As already stated it's enough to move the picture at 25 frames per second to cause illusion. But to cause illusion they are not rapid enough to allow the brightness of one picture or frame to blend smoothly into the next through the time when the screen is blanked between successive frames. This results in a definite flicker of light that is very annoying to the observer when the screen becomes alternatively bright and dark.

This problem is overcome by using the concept of interlaced scanning. In interlaced scanning each frame is divided into two fields odd & even.Each field is scanned alternately. It reduces flicker to an acceptable level since the area of screen is covered at twice the rate.

In a 625 line TV system, for successful interlaced scanning, 625 lines of each frame or picture are divided into sets of 312.5 lines and each set is interlaced alternately to cover the entire picture area. To achieve this, vertical frequency is made at 50 Hz, so the flickering effect is eliminated without increasing the speed of scanning.Since the frequency of vertical trace is 50 Hz,the nominal duration is 20ms (1/50). Out of this period, 18.72ms is spent in bringing the beam from the top to bottom of the frame and the remaining 1.28 ms is taken by the beam to return back to the top to commence the next cycle.

Since horizontal and vertical oscillators operate simulataneously, 20 horizontal lines (1.28 ms/64 ms = 20 lines) get traced during each vertical retrace interval. Thus, 40 lines are lost per frame as blanked lines during the interval of the two fields. This leaves the active number of lines for scanning the picture to 585 ie., (625 – 40) per frame.

THE COMPOSITE VIDEO SIGNAL

In monochrome television the composite video signal consists of camera signal corresponding to light variations in picture, blanking pulses to make the retrace invisible and synchronizing(sync) pulse to keep te scanning in step with that in the camera at transmitting end. A horizontal sync pulse is needed at the end of each active line period, whereas the vertical sync pulse is required after each field of scanning. The amplitude of both horizontal and vertical sync pulses are kept the same to obtain higher efficiency of picture signal transmission but their duration(width) is chosen to be different for spreading them at the receiver. Since the sync pulses are needed consecutively and not simultaneously with the picture signals, so they are sent in a time division basis and thus form apart of composite video signal.

In colour television the video signal has additional information about colours in the scene and colours in the scene and colour sync (burst) to synchronize colour reproduction in received picture. Both the colour (chroma) signal and colour burst are contained within the sama channel width.

VIDEO SIGNAL DIMENSIONS

As shown in figure 2.5 the video signal is constrained to vary between certain amplitude limits. The level of video signal when the picture detail being transmitted corresponds to maximum whiteness to be handled, is referred to as peak white level. This is fixed at 10 to 12.5 percent of the maximum value of the signal while the black level corresponds to approximately 72 percent. The sync pulses are added at 75 percent level called the blanking level. The difference between the black level and the blanking level is known as pedestal. However in actual practice the two levels being very close tend to merge with each other as shown in figure 2.5. Thus the picture information may vary between 10 percent t0 about 75 percent of the composite video signal depending on relative brightness of the picture at any instant. The darker the picture, the higher will be the voltage within those limits.

At the receiver side the picture tube is so biased that the received video signal corresponding to about 10 percent modulation yields complete whiteness at the particular point on the screen.Similarly an analogous arrangement is made for the black level.

DC COMPONENT OF THE VIDEO SIGNAL

In addition to continuous amplitude variation for individual picture elements, the video signal has an average brightness of the scene. In the absence of the D.C. component the receiver cannot follow changes in brightness as the AC camera signal,say for grey picture elements on a black background wil then be same as a signal for a white area in a grey background. In figure 2.5 DC components of the signal for these lines have been identified each representing a different level of average brightness of any scene with the help of three lines in a separate frames because average brightness can change only from frame to frame and not from line to line.

PEDESTAL HEIGHT

Pedestal height is the distance between the pedestal level and average value (dc level) axis of the video signal. This indicates average brightness since it measures how much the average value differs from the black level

As illustrated in figure 2.6 the composite video signal contains horizontal and vertical blanking pulses to blank corresponding retrace intervals. The repetition rate of horizontal blanking pulse is therefore equal to the line scanning frequency (15625 Hz). Similarly the frequency of vertical blanking pulse is equal to field scanning frequency of 50 Hz.

DETAILS OF HORIZONTAL BLANKING

PERIOD	TIME (µseconds)
Total lines (H)	64
Horizontal blanking	12 ± 0.3
Horizontal sync pulse	4.7± 0.3
Front porch	1.5± 0.3
Back porch	5.8 ± 0.3
Visible line time	52

TELEVISION SYSTEMS

Three colour systems have been developed so far, for the transmission and reception of video signals. They are,

1. The American NTSC (National System Committee).
2. The German PAL (Phase Alteration Line by line).
3. The French SECAM (Sequential Colour A Memory).

While these three systems differ in several aspects, the basic features of all the three systems are almost the same. The basic features of these colour systems have been discussed below.

All the three systems describe the picture in terms of luminance and colour difference signals. It is the way in which colour difference signals are modulated that makes one system different from the other.

Out of these three systems, the PAL system, which is a refinement of NTSC, has been adopted in our country.

In all the three systems three signals pertaining to the three basic colours red, green and blue are obtained by scanning the picture by a colour camera.

These signals are then contributed to provide the luminance and chrominance signals. The three systems differ mainly in the method of processing the colour signals and in the method used for transmitting the chroma signals. The main feature of the PAL system is given below.

1. Three colour signals i.e., signals corresponding to the red, green and blue signals (or as R, G and B signals in short).
2. The luminance or 'Y' signal is obtained by combining the colour signals in the proportions given below.

$$Y = 0.3R + 0.596\ G + 0.11\ B$$

The luminance signal contains the brightness information and by itself it can reproduce black and white pictures.

3.Colour difference signals are then obtained by combining the colour signals with the luminance signal. Since the entire colour information can be contained in two colour difference signals (and luminance signal), only two colour difference signals (i.e.) R-Y and B-Y signals are obtained. These can be obtained from subtracting the luminance from the chroma signal and contain the colour information.

4.The R-Y and B-Y signals are weighted to obtain U and V signals. The weighting factors are

$$U = 0.493(B-Y)$$

$$V = 0.877(R-Y)$$

The weighting is done just to avoid over modulation of the carrier and except of the weighting, the U and V signals carry the same colour information as contained in the R-y and B-Y signals. The U and V signals form the final colour difference signals.The luminance signal is transmitted as it is and provides a high definition black and white picture. The chroma signals add colour to picture, advantage being taken here of the limited resolving power of the eye to colour.

5.The luminance and chroma signals have finally to be transmitted by modulating these on the vision carrier.

Standards of the video signal adopted in our country

No of lines per frame	625
Frame frequency	25 / second
Field frequency	50/second
Line interlace	2:1
Picture aspect ratio	4:3
Channel separation	70MHz
Picture carrier modulation	AM
Polarity of transmission	Negative
Separation of sound carrier to vision	5.5MHz
Sound modulation	FM
Line sweep frequency	15625Hz

• Chroma sub- carrier	4.43361875MHz

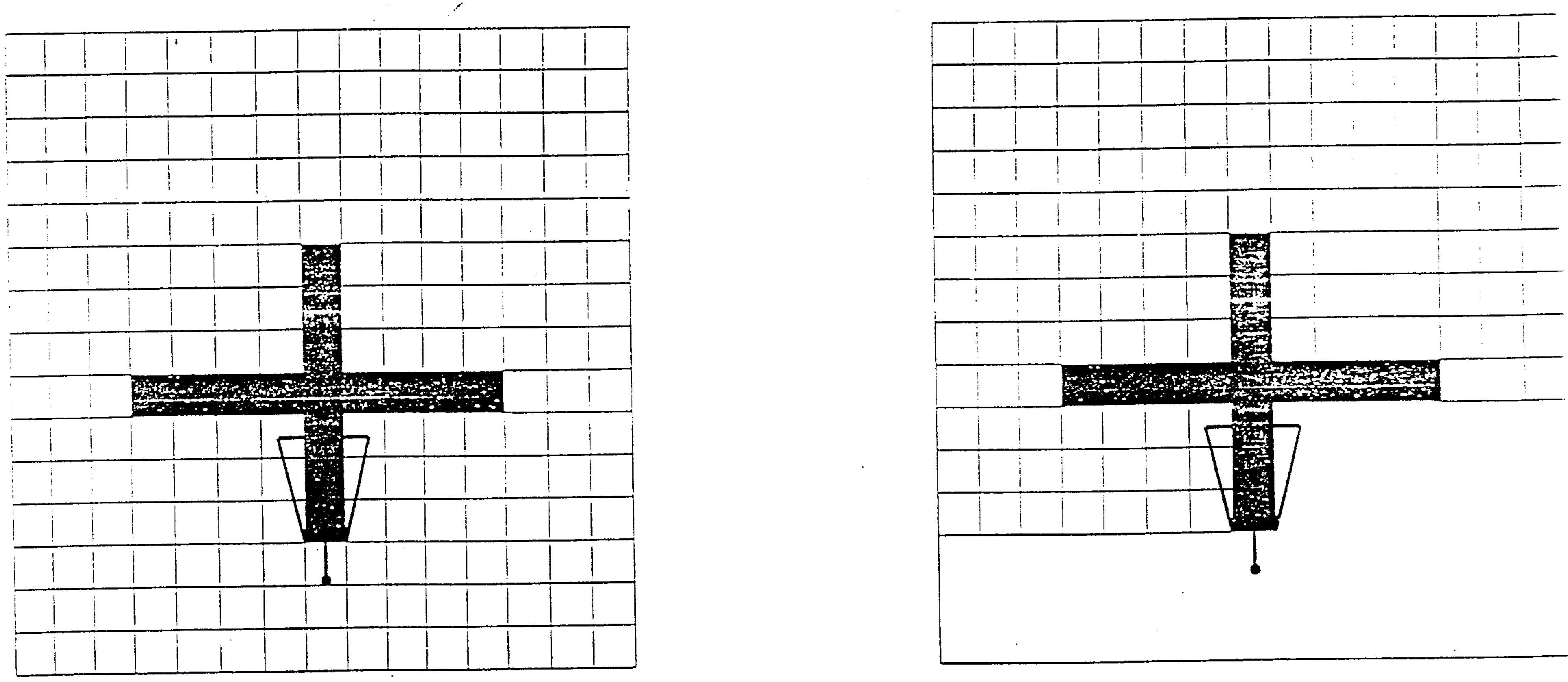


FIGURE 2.1 TRANSMISSION OF AN IMAGE OF BLACK CROSS ON A WHITE BACKGROUND (PIXEL BY PIXEL)

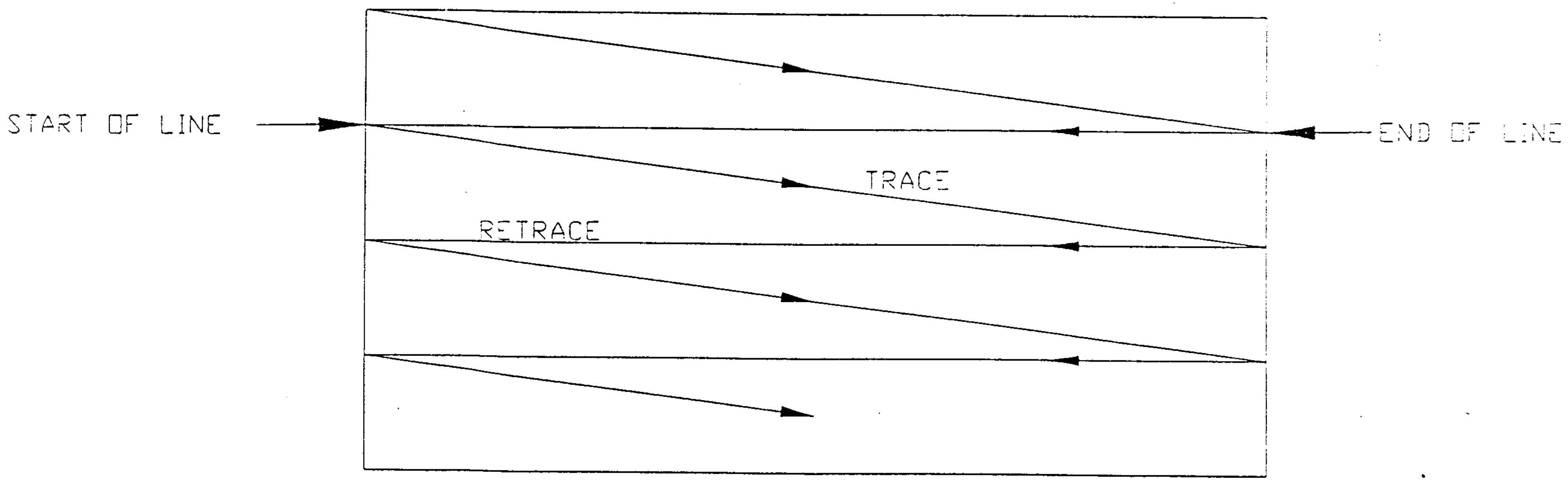


FIGURE 2.2 : SCANNING PROCESS

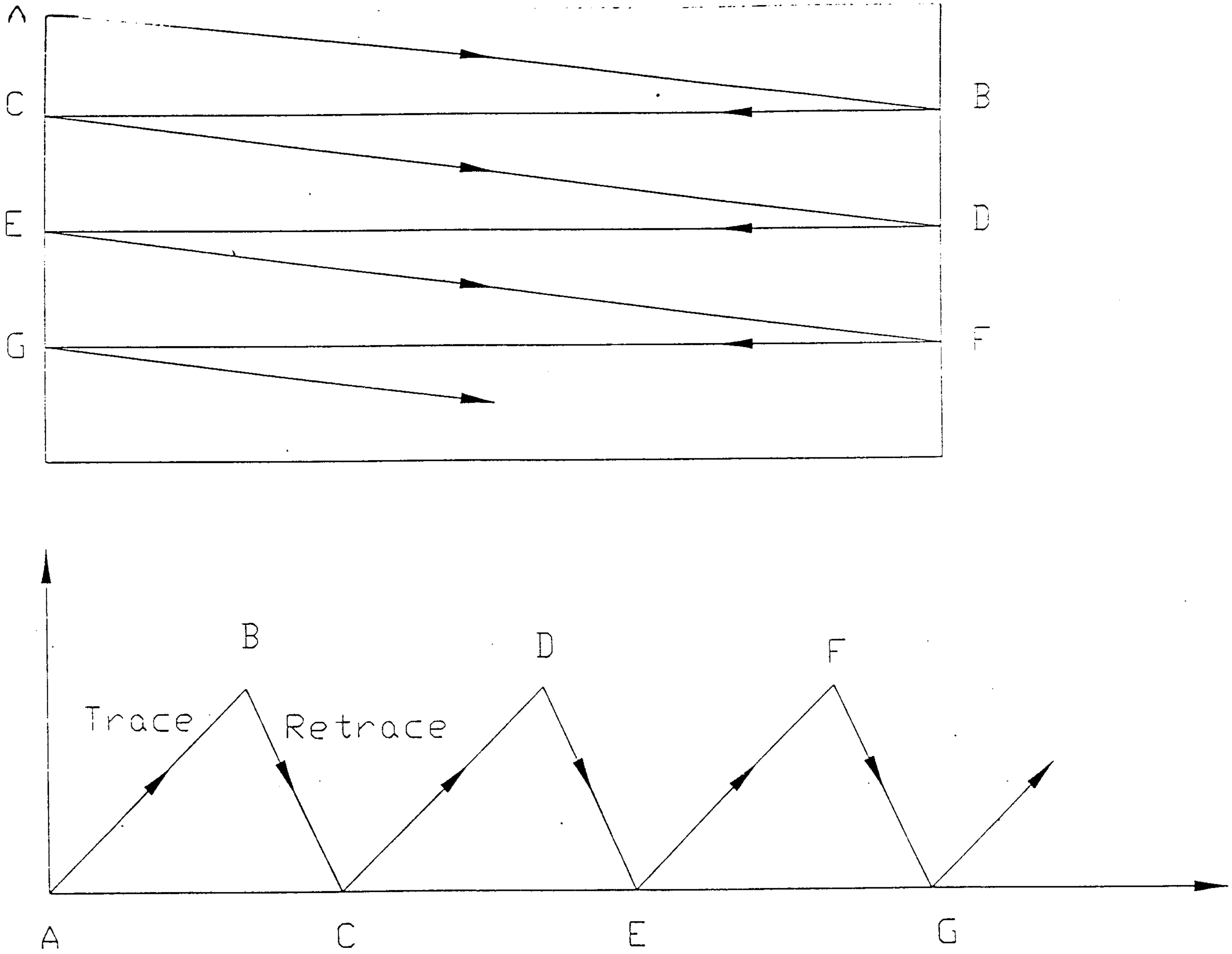
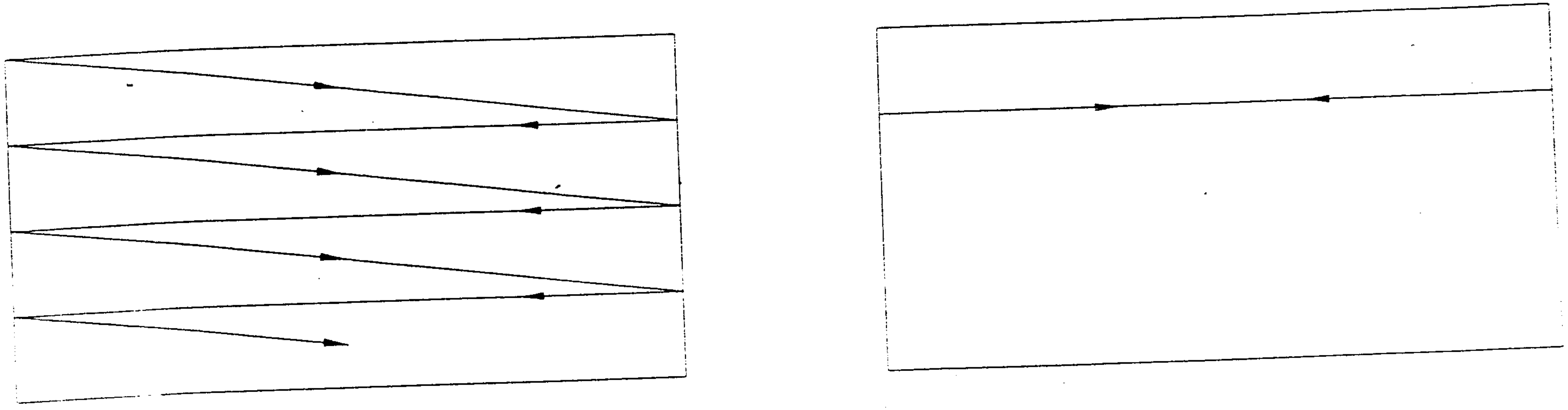


FIGURE 2.3 : HORIZONTAL SCANNING



VERTICAL SCANNING PRESENT

VERTICAL SCANNING ABSENT

FIGURE 2.4 : VERTICAL SCANNING

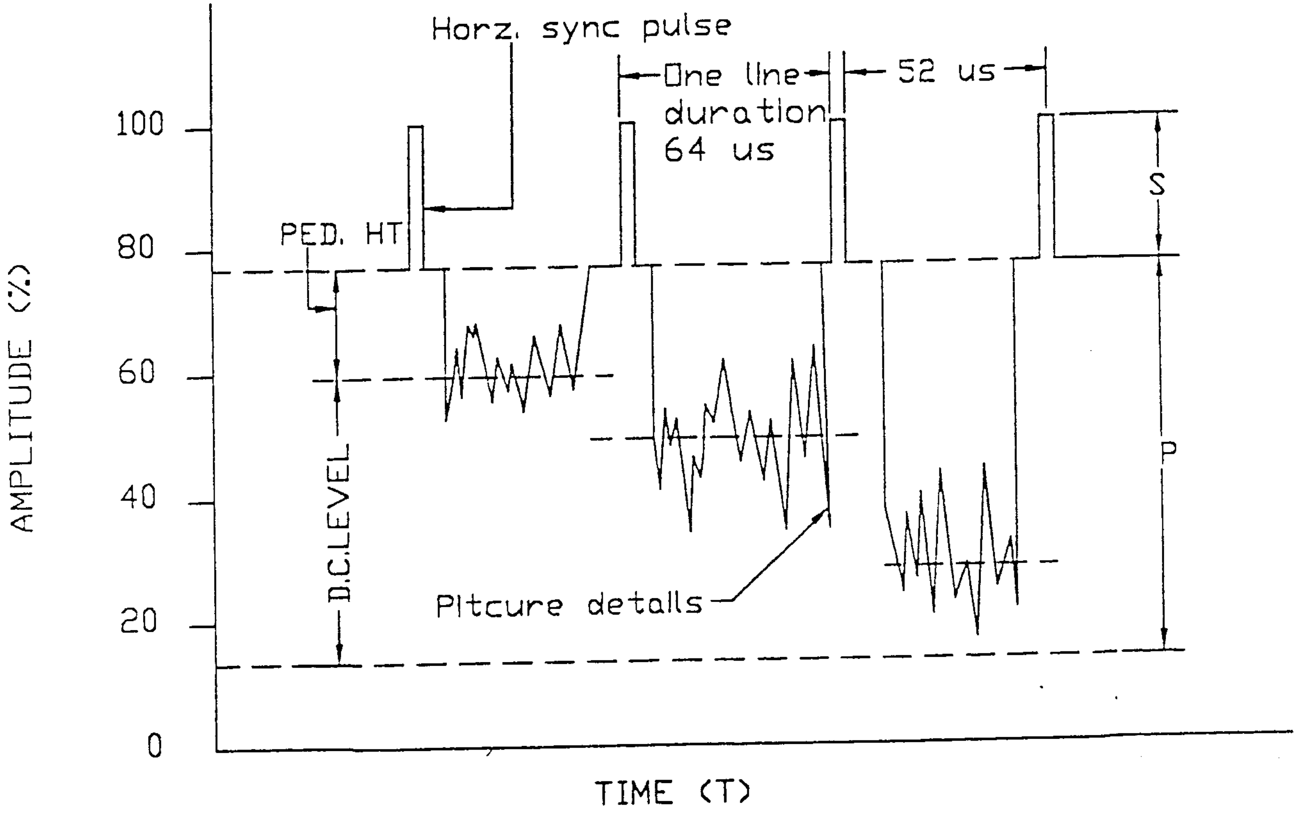


FIGURE 2.5 : VIDEO SIGNAL DIMENSIONS

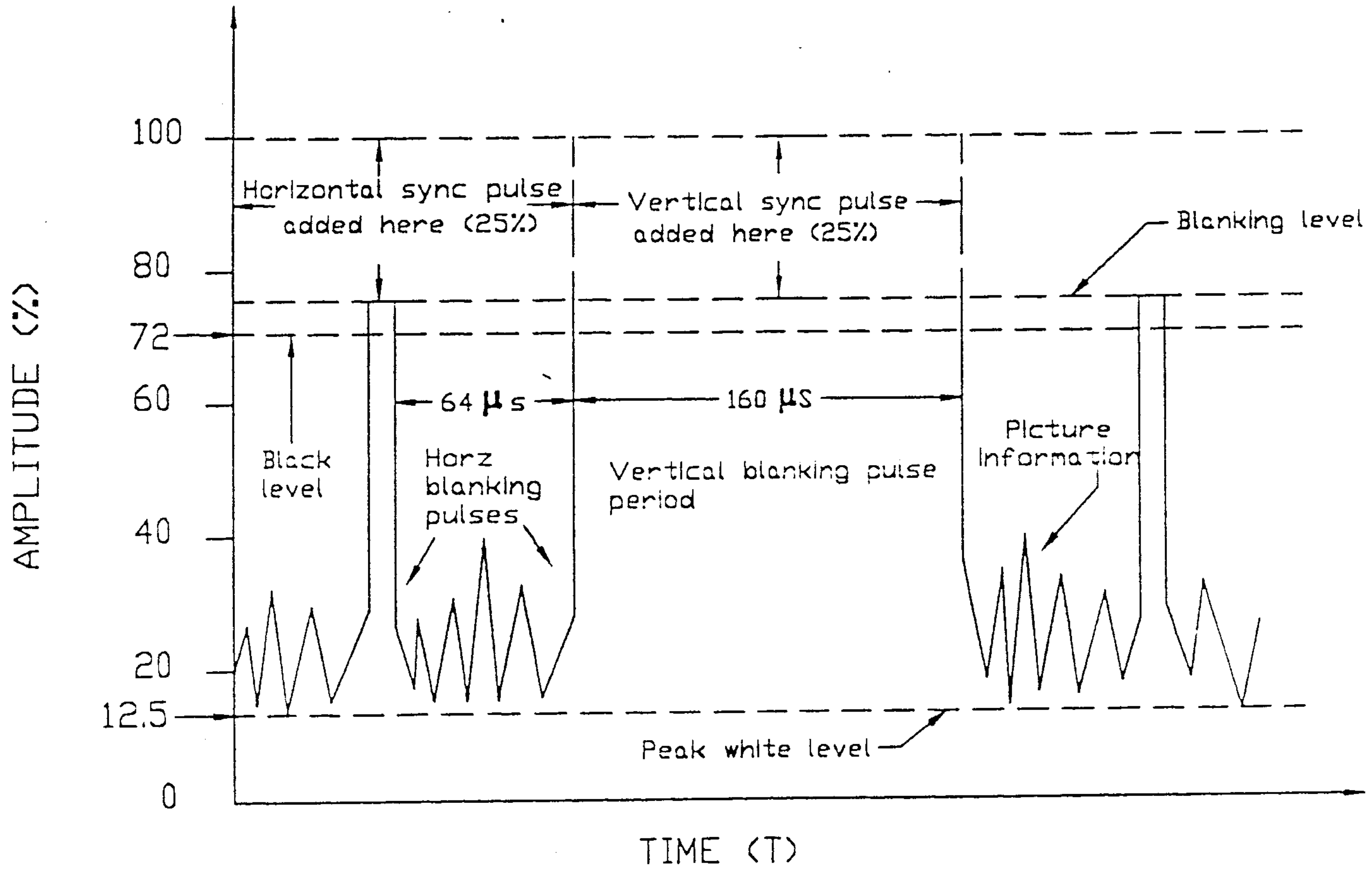


FIGURE 2.6 : VIDEO SIGNAL DIMENSIONS

DIGITIZER BASICS

What actually is a digitizer ? How does it work ? A block diagram of the digitizer and its description shall be able to give you the answer to these questions. The digitizer, as mentioned earlier requires an input image to be stilled for some time. This image is scanned and transferred to the computers main memory. The speed of the digitizer should be compatible with both the computer and the video signal. The timing of the whole process is crucial therefore the method of scanning should match with the above. The method adopted for scanning is explained here along with the block diagram description.

As shown in the figure 3.1 the input signal is the composite video signal. This signal is given to the video synch separator and is separator and separated into synchronous signal vertical synch pulse, burst and odd/even raster. These output information is used in coordinating the digitizing operation. The primary colour signals red, green and blue are given to the A/D converter through a 4 to 1 analog multiplexed. The control signal for the analog multiplexed is given from the computers data bus through a 2-bit latch. The red, green and blue signals are digitized one by one. There is a provision for monochrome signal(B/W) to be digitized.

The information(video signal) is digitized from various positions in the rame.In order to get back the picture in a meaningful way the storage had to be one properly. The digitized information is stored in a RAM is given using two resentable binary counter. One is the samples per line counter which counts the number of samples scanned in one line. The other counter is the line counter which counts the number of lines in a frame. Since the samples taken for digitizing from a single line at a time is ten, it's necessary to use atleast a 4 - bit counter. The lower address bus of RAM (A0-A3) is given by this counter value. The line counter counts the number of lines scanned in a single frame. Burst is used as the horizontal sync pulse which is used to reset the samples per line counter as well as the clock signal for the line counter. In interlaced scanning the maximum number of lines scanned in a single frame is 312.5, so it's needed to have at least a 9 – bit counter. Thiscounter will decide the upper nine bit of the RAM's address(A4 – A12). Selecting the particular address and making the write enable high, the digitized value is stored in that memory location. Data stored in the RAM is accessed to the computer's memory through a buffer. 24MHZ clock signal is used as a reference clock signal. By loading some preset value tothe counter the starting position of the frame is adjusted. Using a 3 to 8 decoder the chip select logic is used to select within the block of I.C's used in the block diagram.

SCANNING PRINCIPLE

The time required for scanning one line is 64 μ seconds of which 53.3 μ seconds is the active line period. The remaining 10.7 μ seconds is the horizontal retrace period. Considering a maximum resolution of 640 pixels in line, the time between successive pixel is 0.1 μ second. The analog to digital converter should have a conversion time less than the signal changing time. Since the cost of A/D converter is very high for a very low conversion time as well as other limitations we choose a method of scanning.

In a single frame there are 625 lines and in each line there are 640 pixels at maximum resolution. Each line is considered a division of 10 blocks. Each block has 64 pixels. So the time gap between a pixel in successive block will be 6.4 μ seconds. Now the first pixel from the first block is scanned and first pixel from second block and so on. After completing a frame, second pixel in all the blocks are scanned in the same manner. This procedure is shown in figure . Since there are 64 pixels in a block, it is necessary to can a frame 64 times to get the complete information in 64th frame the last pixel in each lock is scanned. This is shown in figure 3.2 . So it necessary to keep picture unchanged for atleast 64 frames. For black and white picture the minimum still time should be 2.5 seconds and for colour images it should be 15 seconds.

STILL TIME CALCULATIONS

Frame frequency	: 25
Number of frame needed	: 64
Still time for B/W	: (1/25) x 64 = 2.56 sec's

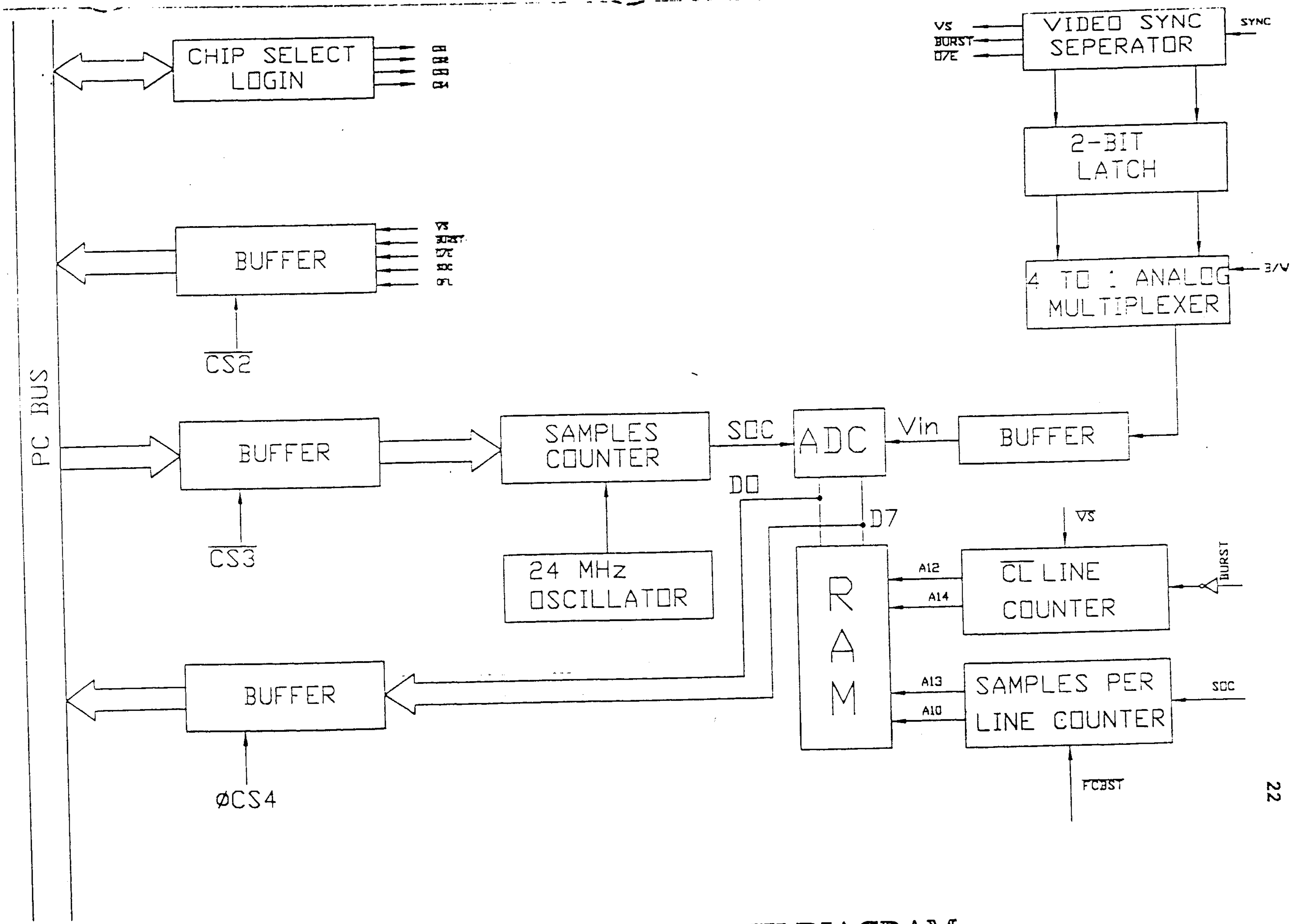
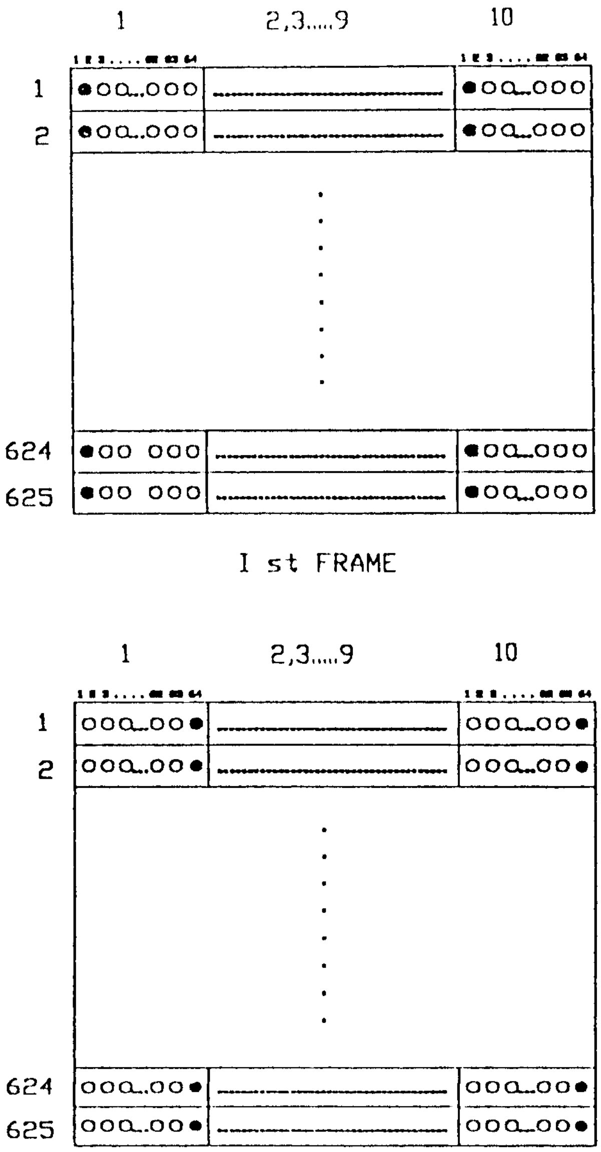


FIG 3.1 : VIDEO DIGITIZER BLOCK DIAGRAM



I st FRAME

64 th FRAME

○ PIXEL ● PIXEL DIGITIZED

FIGURE 3.2:SCANNING PROCESS

HARDWARE DESCRIPTION

Any circuit interfaced with the computer must form a perfect link with it. The linking between the card and the computer is carried out through the computer peripheral slot, so care should be taken when any circuit is interfaced with the computer. The signal lines from the computer play a vital role in this digitizer . It is necessary to know how actually it is interfaced, how the split signal is used to carry out the processes of digitizing this chapter deals with the computer interfacing and the process of digitizing.

INTERFACING PC AND BUS

When any system is interfaced with the computer the address and data bus of the system should match with the bus structure of the computer. So threshold should have thorough knowledge regarding the PCBUS. The PCBUS is connected to the environment through the peripheral connectors. All the signals inside the computer is available in the peripheral slots.There are 62 pins arranged as 31 in a row.

Figure shows the peripheral slot on IBM PC motherboard. A+ n front of a signal indicates that the signal is active high and a '-' indicates the signal is active low. A0-A19 on the connectors are the 20 demultiplexed address lines and D0-D7 are the eight data lines. IRQ2-IRQ7 are interrupt request lines which go to 8259A Priority Interrupt Controller so that peripheral boards can interrupt the 8088 if necessary. Some other signals are the power supply voltages: ∓5v, and ∓12v; thestandard ALE; MEMR, MEMW, IOR/W and some clock signals.DMA request pins DRQ1-DRQ3 allow peripheral board to request the use of buses, it lets the peripheral device or board know by asserting the appropriate DACK0-DACK3 signal. The AEN signal on the connector can be used to get the DMA address on the bus. When programmed number of bytes has been transferred , the T/C pin on the connector goes high to let the peripheral know that the transfer is complete. A peripheral board can assert the I/O CHRDY pin on the connector low to cause the 8088 to insert WAIT state until it is ready.

TO GENERATE SAMPLE SIGNAL

In the monochrome signal, it is enough to sample B/W signal. This is shown in digitizer's circuit diagram. The B/W signal is given to the ADC through the relay circuit. The relay is activated by using 74LS137(U11). This IC is a 3-8 decoder with a D flip-flop in the input side, so that when the latch enable (LE) goes low to high, the data present in the last but before transmission, is stored in the latch. Depending on the input value, the corresponding output is activated so that one of the relay is closed. The signal is sampled to ADC via a transistor buffer. The transistor is connected in a common collector mode, so that the driving capacity of the signal is increased.

TO SEPARATE THE SYNC SIGNAL

Naturally composite video signal consists of colour information as well as synchronous (timing) information. To get back the digitized colour information to meaningful one the synchronous signal is necessary. The synchronous signal consists of horizontal retrace,vertical retrace front porch, back porch. The separation of synchronous signal from composite video signal is done by using IC

LM1881 (U13). This IC is a video sync separator. It is an 8-pin IC with composite video signal as input. There are provision to get synchronous signal, vertical sync pulse and burst as outputs. Burst signal is used as horizontal sync pulse. There is also indication whether the frame is odd or even by producing a high pulse in pin 7. Pin 6 is the reset pin in which a resistor (100 K) and capacitor of 0.01μF is connected in parallel. This value corresponds to the line frequency of the normal video signal (15625 Hz). In case the line frequency on incoming video signal is not 15625 Hz, some other value of capacitor should is added.

THE DIGITIZING CIRCUIT

The digitizing part is the very important thing in the hardware. A/D Converter is the basic block in the digitizing unit. The ADC0820 (U5) is a simultaneous type A/D converter. This is also called flash converter. It is a 20-pin IC which has 8 output lines. The input power supply is +5v. This IC has internal track-hold arrangement eliminating the need of external sample and hold device. This type of

ADC is selected because this technique only has a very low conversion time. For this IC the nversion time is 1.2 μsecond s. Since the time between successive pixels scanned is 6.4 microsecond, this type of ADC can be used. Preset P1 is used to optimize the conversion by setting correct black & white level.

As mentioned earlier, the time gap between successive pixels scanned is calculated as 5.33 microsecond. This is achieved by giving a calculated pulse of period of 5.33 microsecond. The standard clock is generated using a crystal oscillator of 24 MHz frequency. The clock signal is given to an 8 bit counter (U8). When the preset value is loaded, U8 will produce a pulse at end of 16 clock pulses. The S3 pin of U8 is connected to the RD of ADC0820 and also it is connected to WE of RAM.

CALCULATION OF 5.33 MICRO SECOND

Clock frequency	: 24 MHz
Clock signal period	**: 1/24 MHz = (4.16667E-08) seconds**

Time period of the signal at RCO = (4.16667E-08) x 16 seconds Since RCO is connected to the second counter, for every 4 pulses there will be a change of state from low to high or vice versa, produced at s3 in of second counter of period 1.333E-06x4 (5.333 μseconds). So for every 5.33 μ second the ADC is enabled. At that time sample is digitized and stored in RAM. The signals which has to be digitized are primary colour signals red, green and blue.

When a vertical sync pulse is produced in pin3 of IC LM 1881 it indicates the end of frame. At the beginning of the next frame the scanning should not be done on the same pixel scanned already. This problem is avoided using software technique. At the end of every frame some preset value is loaded in the first counter. Depending on the preset value the pulse width for the first instant is varied, so that the time period of the signal produced at RCO pin is also varied. From the next pixel onwards, the gap is same as what is in previous case (5.33 μ second).

During the retrace period of the video signal there will not be any picture information. So during that period the digitizing should not be done, since it is waste in storing unwanted signal in RAM.

This is done by electronic switches (U7A) provided in the circuit. Signal is given in such a way that when burst occurs the switch get closed. When the switch is closed the signal to ADC is grounded.

THE DIGITIZING PROCESS

The A/D converter converts the analog signal to an 8-bit digital information. Since we are not scanning the pixels continuously, there is a need to give correct address location to store the information in the RAM. The RAM 6264 (U6) is a high speed CMOS RAM of 8KB capacity. It is a 28 pin package having common input and output data lines. The memory had a thirteen bit address bus (A0-A12).The address to the RAM is given using IC's in U8, where U8 consists of 4-bit counters .One counts the samples taken in a line at a time. The input to this counter is enable pulse for A/D converter. The maximum samples taken from a line at a time is ten. The other three are used to count the number of lines scanned in a frame. Burst signal generated in IC U13 (LM1881) is used as input signal to the IC's as well as it is used to clear the sample counter. Using this four IC's the correct address where the information is to be stored is decided. When the start of conversion pulse is low from the IC U2 the RAM is write enabled, so that the digitized information is written in that memory location. The stored information in the RAM is transferred to computer's main memory by making the read enable high. A bi-directional buffer (74 HCT 245) IC U4 is buffer is connected since there is a huge difference in speed between PC's and old XT which runs at a very low speed compared to 486.

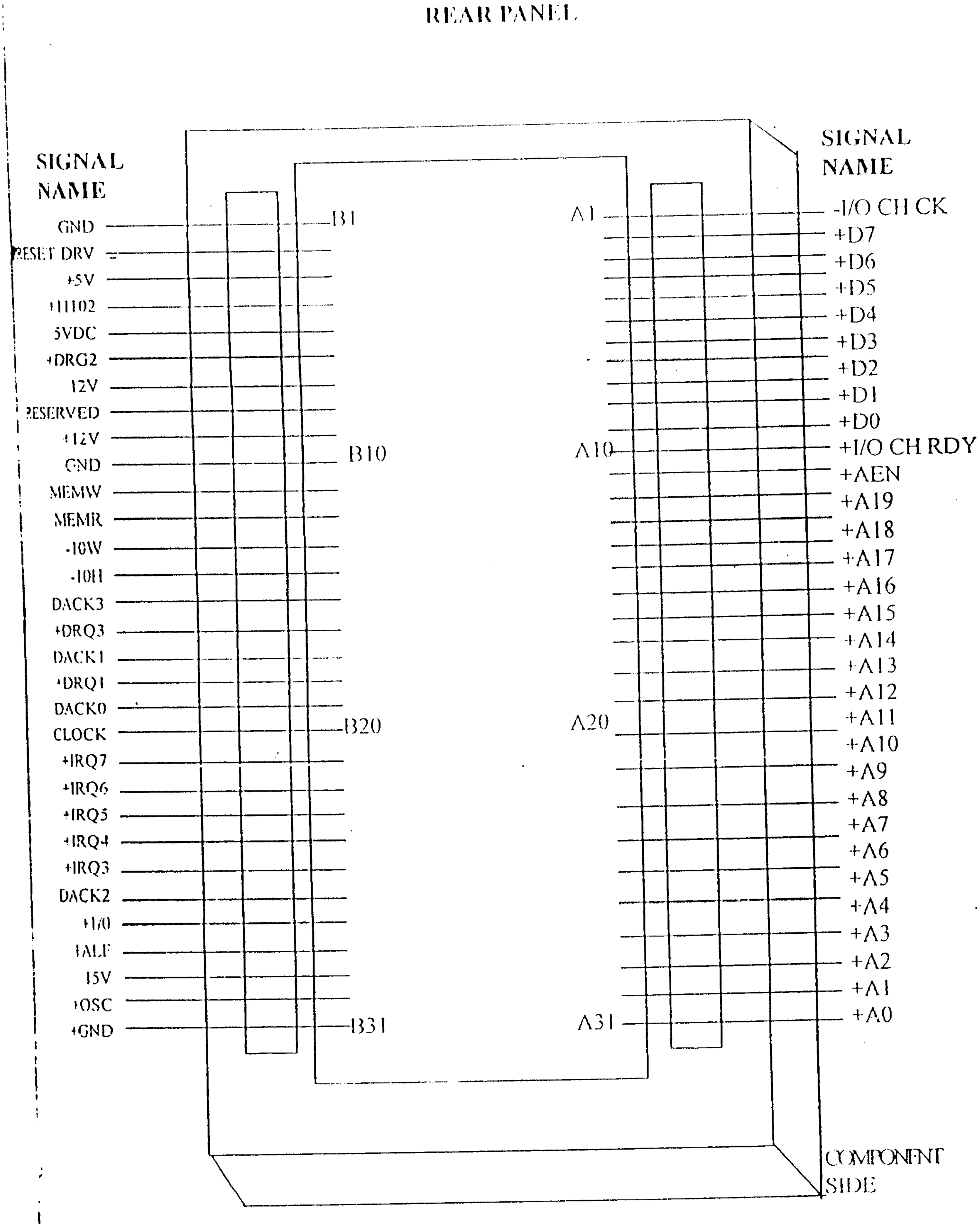


Fig. : 4.1 Peripheral slot of personal computers

DIGITIZER SOFTWARE

Software is the life to this card. It is the master steering the hardware to its destination. Broadly, software could be classified as system software and application software. System software, depends on the hardware and hence it is hardware dependent or to be specific the software is developed for one particular hardware and that would not suite the other. Whereas the application software depends solely on the type of file its going to process and pays no attention to the hardware or the application which created it. In this chapter first we discuss about the system software, TIFF file format, the format we have made use of in storing the images. Since the application software needs although discussion we have added it as a separate appendix.

SOFTWARE BASICS

System software means that it's the primary software needed to run the system. By meaning primary it is clear that this is essential, without which the card is dead. The system software was written in

Turbo C language. The choice of the software was simple – it offered what we wanted.

TIMING INFORMATION

Although we have dealt adequately about the timing information in the chapter 3, still we present you with a brief discussion since the software controls only the timing information.

A complete picture line has length of 64μ seconds, including the synchronization pulse(53.3 μseconds of picture information and 10.7 μ seconds of horizontal retrace period). If we were to sample at 640 pixels (considering a maximum resolution of 640 pixels per line), the distant between successive samples is 0.1 μ seconds. This means that in 0.1 μ seconds the ADC should have digitized and saved the information to the RAM before we move to the next pixel, which is at a distance of 0.1 μ seconds. Now, the ADC what we have used is far too slow since it achieves a minimum conversion time of 1.2 μ seconds. Hence use of this ADC would not give us the desired results, since it takes a minimum of 1.2 μ seconds for a single conversion, before it starts the next conversion the picture would have moved by 12 pixels. Keep in mind we have considered only the minimum conversion time of ADC and we have not considered the writing time on to the RAM. The figure 5.2 reveals the above discussion.

For us going for an ADC with a conversion time of less than 0.1 μ seconds would be a costly affair and is unjustified, since we make an attempt to make continuos picture frames. To overcome the above problem we decided ;to divide the picture line into 10 slots of 64 pixels each. Apart from the hardware, there are further limiting factors, including the software and the data storage in the 8 KB cache. All in all it is required to sample each picture line 64 times before its completely digitized.

From the above discussion, we would make the following points clear before, we make our software. We would be digitizing each pixel at the rate of 5.33 μ seconds per pixel. To keep this on account, we make use of the timer circuit which supplies the required time period. We need to scan a minimum of 64 times to capture all information about a single frame. Keeping all this in mind we forge ahead into the program.

TO CONSTRUCT A FLOWCHART

Before we do the charting we should declare all the parameters to build the program.

- We need to keep track of the number of times we scan the frame. We make use a variable frame_counter.
- We need to know if an end of line is reached. This achieved by checking the horizontal sync pulse, and the count is maintained in the variable line_per_frame (this supplies the address A4-A12 for the RAM)
- We need to deep track of number of samples taken on each line. This maintained by the variable samples_per_line.
- We need to have a offset variable which loads the preset value for the counter.

TIFF

Nowadays we in computers use many graphical pictures for illustration and for the purpose of interaction. As there are many file formats to represent text files there also file formats to these graphics representation. In such a way one of that file formats is **TAGGED IMAGE FILE FORMAT (TIFF).**

TO REPRESENT AN IMAGE

Picture files are of two types: raster or vector.The raster approach describes images as an array of dots, the vector approach uses a series of geometric commands.

For example consider a raster file that might store an image of a square. It would view the paper as a matrix with white dots(where there is no image) and black dots(where there is an image). The dots are called pixels. Then it would describe the picture as a series of 1's and 0's where (perhaps) 1=black dot, 0=white dot. The figure may look like the following figure.

```
00000000000000000000000000000000
```

```
00000000000000000000000000000000
00000000000011111111111000000000
00000000000011000000011000000000
00000000000011000000011000000000
00000000000011000000011000000000
00000000000011111111111000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

In the above figure we arranged the ones and zeroes to make it easier to see the square. On the file, they would be laid out one bit after another. You wouldn't have carriage returns to indicate where a row ends, as you do in the figure. It could look something like the following figure

```
0000000000000000000000000000000000000000000000000
0000000000000000000000000011111111110000000000000
0000000000011000000011000000000000000000000110000
0000110000000000000000001100000001100000000000000
0000000110000000000000000000001111111111100000000
0000000000000000000000000000000000000000000000000
00000000000
```

If we give your that same stream of bits, and told you that they were supposed to be arranged in a matrix 43 columns and seventeen rows, you could reassemble the picture of the square without any ore piece of information for this to be a useful format. As described, each pixel can be only "white" or "black". Most scanners these days, however can support gray scales, which are intermediate levels of brightness between the white and black. If we assign one bit per pixel , then each pixel can have only two colours – black(1) OR WHITE (0) Assign two bits per pixel, and each pixel can have four colours – black scanners support atleast 16 levels of gray scale, so they assign four bits to each pixel. Some scanners support 256 levels of gray scale, so they assign 8 bits to each pixel.

The vector approach to images, on the other hand, describes the square as just as set of lines. Using the vector approach, the file for the square would look like the following figure as shown below, there are four items in this file.

Item 1 = line, coordinates = (0,0) – (0,100)
Item 1 = line, coordinates = (0,0) – (100,0)
Item 1 = line, coordinates = (0,100) _ (100,100)
Item 1 = line, coordinates = (100,100) - (0,100)

The ***TAGGED IMAGE FILE FORMAT***, or ***TIFF*** is probably the most generally used gray scale raster format. As you might expect, TIFF file takes a .TIF extension. TIFF is an "opcode oriented" file format. This is a very extensive file format and explaining it would require atleast the 200 plus pages that Aldus uses in its TIFF documentation. We will represent an appreciated version of TIFF –

One that covers monochrome clipart, scanned images , and most problems that we might see. If you need the entire TIFF specification, you can get it from the Aldus Corporation, the company which produced the all famous desktop publisher Page Maker.

The heart of the TIFF file is a bitmap. We'll use a small simple bit map for the example. It looks like the following figure with ten dots by eight dots.

```
0011111100
0100000010
0101001010
0100000010
0101111010
0100110010
0100000010
0011111100
```

TIFF requires that each row be represented by entire bytes. That means you can't have a byte that contains the end of one row and the beginning of the next. If the pixels are not nice multiple of 8 – as is the case here, with a width of ten pixels – you have to pad each row out with zeroes until it fits a byte. Here, we'll add six zeroes to the end of each row to bring each row to 16 pixels. Fill out each row, divide it into bytes, and convert to hex and the diagram looks like the following figure.

```
0011111100000000        00111111  00000000  3F 00
0100000010000000        01000000  00000000  40 80
0101001010000000        01010010  00000000  52 80
0100000010000000        01000000  00000000  40 80
0101111010000000        01011110  00000000  5E 80
0100110010000000        01001100  00000000  4C 80
0100000010000000        01000000  00000000  40 80
0011111100000000        00111111  00000000  3F 00
```

Sample Bitmap converted to Hex

The bitmap is then string out as string of data called "*strip*". The strip is then

3F004080528040805E804C8040803F00

Most bitmap[s are broken up into multiple strips, however, so we'll divide his one into two strips , that way, we'll have TIFF file that will be more like the ones you'll encounter in real life. Suppose the file was written out in strips of five rows. Because the bitmap was only eight rows height , we end up with the complete first strip - 3F004080528040805E80 and a strip with the remaining three row

4C8040803F0. Strip one is then ten bytes long and strip two is six bytes long.

Now we have the strips , is that all there is to it ? No, not by any means. A program trying to read these strips would also need to know the following :

- The length or the image , so the program knows how to lay out the rows and columns to make the image useful.
- How to interpret the bitstrip's colours. Is there one bit per pixel (two colours), two bits per pixel (four colours), four bits per pixel (sixteen colours), or eight bits per pixel(256 colours) ?
- The direction of the colour is zero white or black ?
- Does the image use some kind of compression technique?

  This is fairly common to image files
- How many strips are in this sample? Where does each one begin and end?
- What is the image resolution in dots per inch?

❖ Copyright information ,such as who wrote this image and when ?

TIFF answers all of those questions with its header and its image directory. The header is eight bytes long: hex values "49","49","2A" and "00", followed by a four byte offset that points to the mage directory. For reasons that will become clear later, the image file directory is usually at the end of the file.

The four-byte offset is a number from 0 to 4.3 billion that tells the program where to find the some item in the file. The first byte is offset 0, the second is offset 1 and so on obviously TIFF can handle big image.

The header's eight bytes and our sample image's ten bytes add up to eighteen bytes. So the image file directory will start on the 19th byte of the file offset 18(remember, the first one is numbered zero). 18 decimal is 12 hex, so we'll put the offset in bytes 4 through 7 :12 00 00 00. Remember that we store data backwards, so 00000012 is stored 12 00 00 00.

Add the two strips, and a Tiff file now looks like the following hex bytes:

HEX OFFSET	VALUE	HEX OFFSET	VALUE
0000	49	000C	52
0001	49	000D	80
0002	2A	000E	40
0003	00	000F	80
0004	18	0010	5E
0005	00	0010	80
0006	00	0012	4C
0007	00	0013	80
0008	3F	0014	40
0009	00	0015	80
000A	40	0016	3F
000B	80	0017	00

Now let us turn our attention to the image directory. It is arranged as shown in the above table.

BYTE OFFSET	**DESCRIPTION**
(Decimal)	
0/1	Number of entries

2/13	First directory entry. Each directory entry is 12 bytes long, arranged like so :

BYTE 0/1 IN THE ENTRY :

A numeric “opcode” or “tag” identifying this entry. One opcode means “I’m about to tell you the width of the picture”, another means “I’m about to tell you when this picture was made”, and so on.There are dozens of these opcodes – Aldus calls them “tags”, the most important are documented in this section.

BYTE 2/3 IN THE ENTRY :

A quote that indicates what type of data is coming up Its a number from 1to5.

1. a single byte.
2. a sring of ASCII bytes, the last one is a null(ASCII 0)
3. a word (2 bytes)
4. a double word(4 bytes).
5. a fraction ,two 4 bytes words, denominator and then numerator.

BYTE 4/7 IN THE ENTRY :

The length of the data. For example, if the data is an ASCII string, this tells how many characters are in the string. The lengths units are the size indicated by the bytes 2/3. If its a double word, the length is expressing number of double words, but if its a bytes, the length is expressed in number of bytes.

BYTE 8/11 IN THE ENTRY :

Either the data itself or a pointer to the data. If the data is 4 bytes or less, the data goes here. Otherwise, this is a four byte offset pointing to the data.Final four bytes Four nulls (ASCII zeros) ends the image file directory.

IMAGE FILE DIRECTORY ENTRIES

Before we build the image file directory we will need the definitions of the most common opcodes. As we discuss the opcodes, we will built the image file directory entries. Opcodes are in hex.

100: IMAGE WIDTH.

This tells how many columns the image takes. Our image is eight rows by ten columns, so image length equals 10. 10 decimal is 0A hexadecimal. The directory entry starts with the opcode 100. Next is the type. Our length 10 con be expressed either as a word or double-word, so we'll use a word. The ID value for "wpord" is 3, so 0003is the next two hex bytes. Don't forget that 0003 would actually be stored as 03, then as 00, due to "back-words" Intel storage. The length is "1",because we need only to describe length. The length field gets four bytes, however, so "1" is encoded as "00000001" which is stored back-words as 01 00 00 00. Finally, we have four bytes width which we store the width value. Because the value is only two-bytes long, it fits here. The width is 10, but again we have four bytes to store as "back-words", so the last four bytes are 0A 00 00 00. The assembled directory entry is then, 00 01 03 00 01 00 00 00 0A 00 00 00.

101:IMAGE LENGTH

This tells how many rows the image takes .Our image's width is eight rows, so image width equals 8. We assemble the directory entry the same way as we did with image width – it's a "word", and there is only one. The opcode is 01 01 "back-words", the type is 03 00(word), length is 01 00 00 00(just one word, "back- words") and the value is 08 00 00 00.

102:BITS PER PIXEL

The values are again 1,2,4 or 8. It's a "word" type with length of 1. Here, we are discussing an image with one bit per pixel. The directory entry is encoded as 02 01 (opcode), 03 00 (type), 01 00 00 (length), 01 00 00 00(value).

103:COMPRESSION

Some image file are compressed. There are only four possible answers:

1 = No Compression,

2 = Huffman Encoding,

3 = LZW Compression,

32773= Pack Bits compression

The type is also "1", in this case, because it isn't compressed. The directory entry is then 03 01 03 00 01 00 00 00 01 00 00 00.

106:PHOTOMETRIC INTERPRETATION.

Does a zero value represent black or white? Here, we've encoded zero as white, Again the value is a word. If it contains zero, that means zero = white otherwise if it contains 1, zero = black. Encode 06 01 03 00 01 00 00 00 00 00 00 00'

10F:AUTHOR

This is an ASCII string, so the "type of data" value is 5, ASCII . We will encode the author as " MARK M", which won't fit into four bytes because it's a six character and also it must have an ASCII zero at its end, bringing the total length to seven. So the first part encode as 0f 01 01 00 07 00 00 00 and the last four bytes will have to wait until we find a place for the "MARK M" string.

116: ROWS PER STRIP

We said that we choose to organize the image into two strips. We did that so we'd have a more true-to-life file. Most files you will deal with are multi strip files. But large image files are organized into strips which Aldus recommends should be no larger than 8Kb. That means a 31Kb file might end up as three 8Kb strips and the leftover 7Kb would also be strip. One of the ways that the reading program understands the strips is by knowing how many rows to expect in each strip. Recall that we have eight rows we divided into strips of five rows, so we ended up with two strips, one was full five rows and the last one ended up with the remainder, three rows. Note that it's all right to have leftovers. This is a double-word(type of data – 4) value, so we encode it as 16 01 04 00 01 00 00 00 05 00 00 00'

117:STRIP BYTE COUNTS

How long is each strip? Values are double-word, so we'll wind up with two double-words –b eight bytes, too much to in the directory entry. The "length" field this time is "2" , because there are two numeric values – one for each strip. One strip is ten bytes long and the other one is six bytes. The beginning of the entry then encodes to 17 01 04 00 02 00 00 00 and the final four bytes will point to wherever we end up putting the two length values. This means that we have to finish the image directory before we know where the data can go as with " Author" string.

111: STRIP OFFSETS

Where is each strip? Offsets 8 and 12 hex in the file is the starting points inside the file for two strips. The type of data is double-word, type "4". there are two strips so there are two offsets, hence "length" is "2". We will store the offsets "g" and "D" elsewhere in the file . Because we don't know the offset yet, we will have to fix them later. The first eight bytes of encoded information are 11 01 04 00 02 00 00 00.These are the basic opcodes. We'll document dome of the others later. Right now, let's finish the TIFF file. So far we have the header and the data in bytes 0 through 11 hex. the image file directory's firs two bytes tell how many entries to expect in the directory nine , in this case. Then the directory entries follow 12 bytes each. So far the file looks like this:

HEX OFFSET	VALUE	COMMENTS
0000	49 49 2A 00 18 00 00 00	The header
0008	3F 00 40 80 52 80 40 80 5E 80	Strip 1
0012	4C 80 40 80 3F 00	Strip 2
0018	09 00	The number of entries in IFD.
001A	00 01 03 00 01 00 00 00 0A 00 00 00	Image width
0026	01 01 03 00 01 00 00 08 00 00 00	Image length
0032	02 01 03 00 01 00 00 00 01 00 00 00	Bits per sample
003E	02 01 03 00 01 00 00 00 01 00 00 00	Compression
004A	06 01 03 00 01 00 00	Photometric

	00 00 00 00 00	Interpretation
0056	0F 01 05 00 06 00 00 00 ?? ?? ?? ??	Author (offset unknown)
0062	16 01 04 00 0100 00 00 05 00 00 00	Row per strip
006E	17 01 04 00 02 00 00 00 ?? ?? ?? ??	Strip offsets (offsets unknown)
0086	00 00 00 00	End of IFD

The final four zeros terminate the image file directory. Now we can insert the author string and the two four-byte strip byte lengths. "MARK M" encodes in ASCII as 4D 41 52 4B 20 4D 00 – remember that ASCII strings must end with a hex 00. The two strips lengths are 0A hex and 06 stored again "back-words" as 0A 00 00 00 AND 06 00 00 00. insert the offsets for the strips to 08 and 12 hex

The data then end the file:

008A	4D 41 52 4B 20 4D 00	"MARK M" encoded
0091	0A 00 00 00 06 00 00 00	Strip lengths
0099	08 00 00 00 12 00 00 00	Strip offsets

HEX OFFSET	VALUE	COMMENTS
0000	49 49 2A 00 18 00 00 00	The header
0008	3F 00 40 80 52 80 40 80 5E 80	Strip 1

0012	4C 80 40 80 3F 00	Strip 2
0018	09 00	The number of entries in IFD.
001A	00 01 03 00 01 00 00 00 0A 00 00 00	Image width
0026	01 01 03 00 01 00 00 08 00 00 00	Image length
0032	02 01 03 00 01 00 00 00 01 00 00 00	Bits per sample
003E	02 01 03 00 01 00 00 00 01 00 00 00	Compression
004A	06 01 03 00 01 00 00 00 00 00 00 00	Photometric Interpretation
0056	0F 01 05 00 06 00 00 00 8A 00 00 00	Author (offset unknown)
0062	16 01 04 00 0100 00 00 05 00 00 00	Row per strip
006E	17 01 04 00 02 00 00 00 91 00 00 00	Strip offsets (offsets unknown)
0086	00 00 00 00	End of IFD
008A	4D 41 52 4B 20 4D 00	"MARK M" Encoded
0091	0A 00 00 00 06 00 00 00	Strip lengths
0099	08 00 00 00 12 00 00 00	Strip offsets

And that is basic TIFF file. You may see other opcodes, but they tend to be informational in nature and can be ignored. Here are those opcodes:

FF AND FE OPCODES:

Sub-File Type And New Sub-File Type :

It is possible to have a single TIFF file that is composed of several " sub-files". You'll probably not see a TIFF file with multiple sub-files, but some programs write out a sub-file header for all TIFF files even if they contain only one sub-file. The variable type is double-word. If the opcode is "Sub-file type", you'll probably see a value of 1. If the opcode is "New sub-file type", you'll probably see a value of 0. Anything else will require study of the full TIFF specification.

11A AND 11B OPCODES:

X Resolution and Y Resolution

Images are scanned or otherwise created in different resolutions. For example , the scan Jet scans in 1/72', 1/150' or 1/300' resolutions. These days, 1/400' resolution is common among flatbed scanners. This opcode advises the reading program of image resolution. The variables are stored as double-words type "5", the "fraction " type, which has the denominator first and the numerator second. For example we had 1/72" resolution image. The hex values for denominator (72) and numerator(1) are 00000048 and 00000001,respectively. We would store 48 00 00 00 01 00 00 00 somewhere in the file, then we'd construct the IFD entry 0A 11 05 00 01 00 00 00 x x x x, where "xxxx" is the offset of the numerator/denominator combination.

128 HEX OPCODE:

Resolution Unit

This choices metric or English units. Recall that the X Resolution and Y Resolution opcodes are fractions, a scanned image that was scanned at 72 dots per inch would store X resolution as "72" and "1" , to indicate that the width of the dots is 1/72" in the "X" direction. Resolution unit allows you to choose whether this is 1/72 inch or 1/72 centimeter. The type of unit is word. Values are 1 (no units), 2(inches), 3(centimeters).

115 HEX OPCODE:

Samples Per Pixel

Some colour applications store then images in what is called "RGB format" – red data, green data and blue data. The monochrome scanned images we are discussing will have only one sample per pixel, whether grey scaled or not. The type of data is word, the value will equal "1" for monochrome images.

The remaining opcode are all informational. They all use ASCII data fields- gields, recall that are terminated with an ASCII 0.

13B OPCODE:

Artist

Contains artist name , copyright and so on.

132 OPCODE:

Stored as "YYYY:MM;DDHH:MM:SS" Hours are expressed as "military" time – 0 to 23.

13C OPCODE:

Host Computer

Or possibly the software package that was used to write the image.

10E OPCODE:

Image Description.

TIFF includes this space to describe a file.

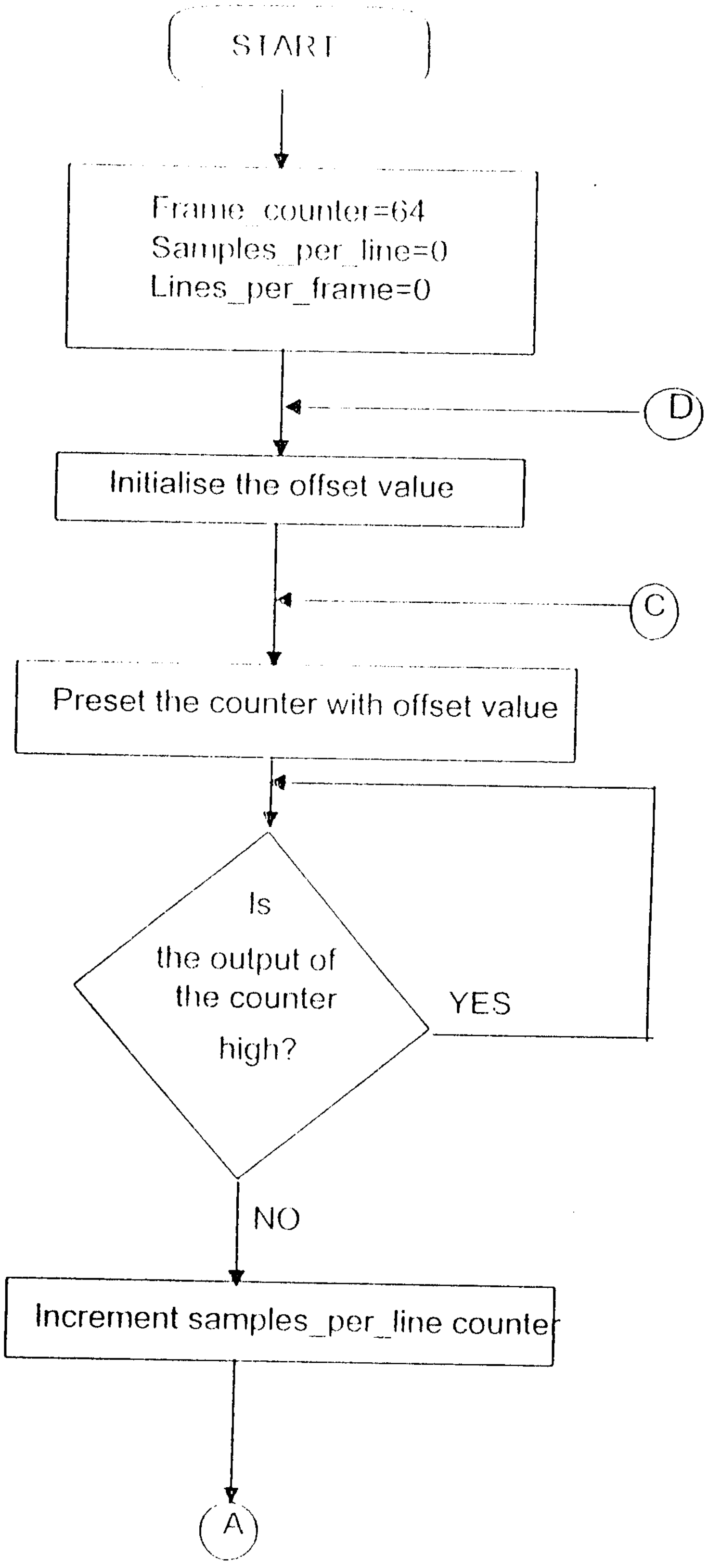
10F AND 110 OPCODES:

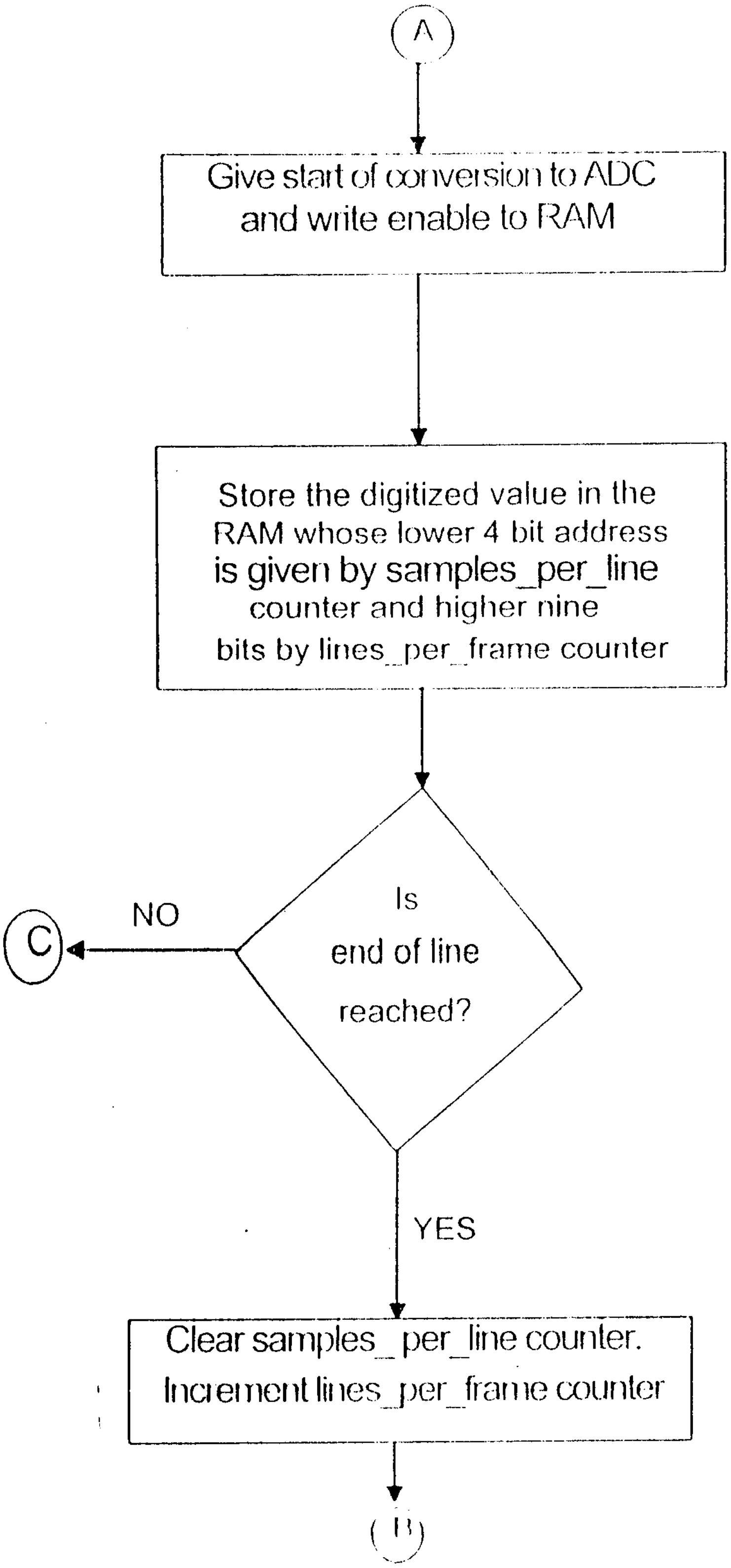
These are for the make and model of the scanner that created the image.

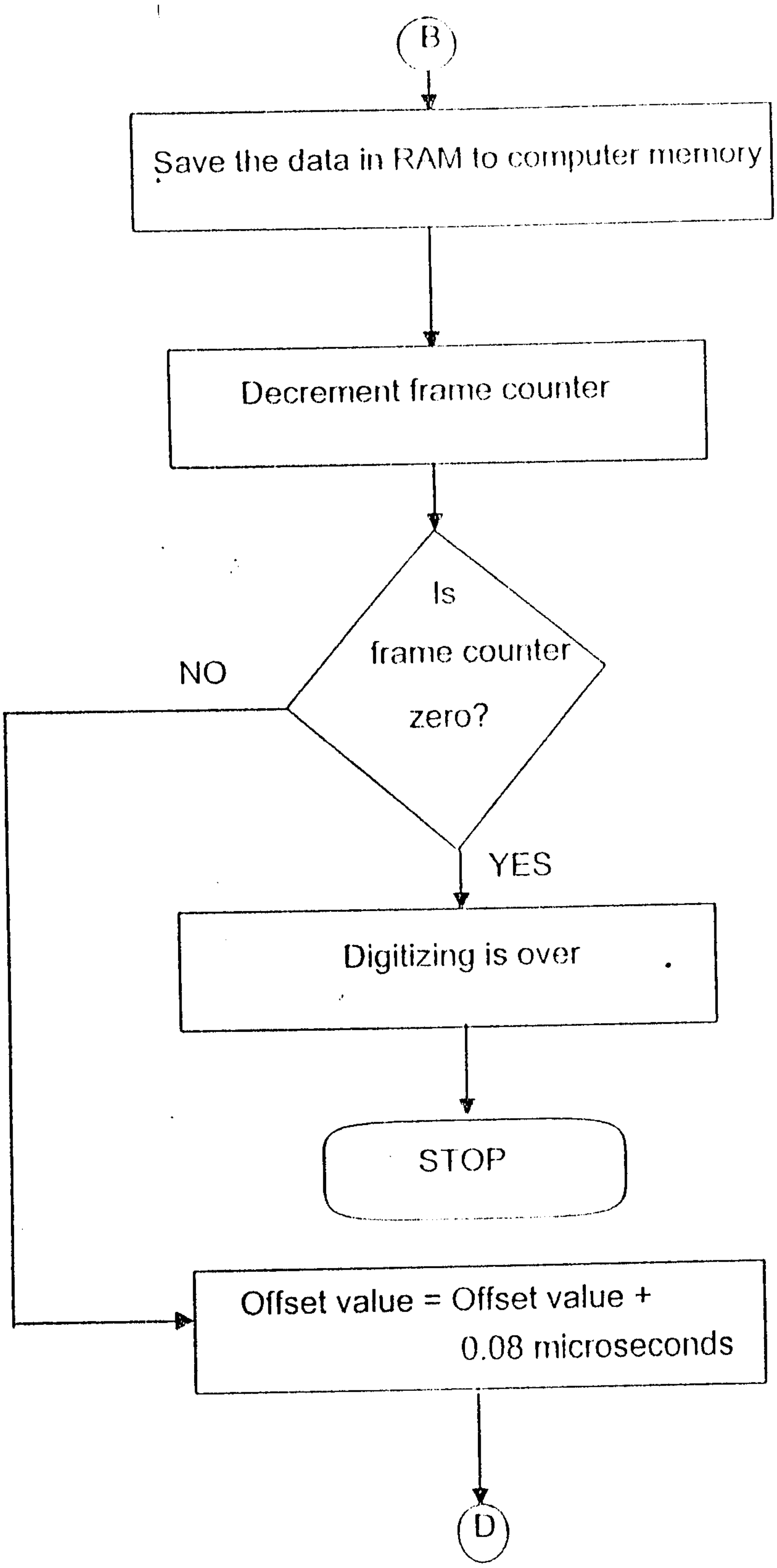
131 OPCODE:

Software:

This is for the name and version of the software that created the image.


START
Frame_counter=64
Samples_per_line=0
Lines_per_frame=0
D
Initialise the offset value
C
Preset the counter with offset value
Is
the output of
the counter
high?
YES
NO
Increment samples_per_line counter
A


A
Give start of conversion to ADC and write enable to RAM
Store the digitized value in the RAM whose lower 4 bit address is given by samples_per_line counter and higher nine bits by lines_per_frame counter
Is end of line reached?
NO
C
YES
Clear samples_per_line counter.
Increment lines_per_frame counter
B


B
Save the data in RAM to computer memory
Decrement frame counter
Is frame counter zero?
NO
YES
Digitizing is over
STOP
Offset value = Offset value + 0.08 microseconds
D

VHDL

INTRODUCTION TO VHDL

VHDL means **V**ery **H**igh speed integrated circuit **H**ardware **D**escription **L**anguage. To compete with the developing electronics world ,vendors built products with increase in its functionality, its performance, decrease in its cost, decrease in its power consumption and even decrease in its size. On this way we are now going to explain about the heart of our project ie., VHDL. The main reason for which VHDL has become the hero of the electronics world is that its possibility to enable the designers to describe large circuits and bring products to market rapidly. Though there many Hardware Description Languages, the advantage of VHDL over other languages are

- ☛ Power and flexibility.
- ☛ Its device dependency.
- ☛ Its portability.
- ☛ Its benchmark capabilities.
- ☛ It's quick time in marketing and its low cost.

DESIGN SYNTHESIS USING VHDL

- Define the design requirements.
- Describe the design in VHDL (formulate and code the design)
- Simulate the source code.
- Synthesize, optimize, and fit (place and route) the design
- Simulate the post-layout (fit) design model
- Program the device.

DEFINING THE DESIGN REQUIREMENTS

Before writing codes for our design we should have a clear idea of the design objectives and requirements. They are

- The function of the design we are doing.
- The clock-to-output times.
- The maximum frequency of operation
- Critical paths.

DESCRIBING THE DESIGN IN VHDL

The first step in designing is to decide upon a design methodology. We are already familiar with the methodologies as top-down, bottom-top, or flat. The first methods involve creating design hierarchies, the latter involves describing the circuit as a monolithic design.

The method in top-down design methodology involves dividing the design into functional units provided each unit has separate inputs and outputs and performing a particular function. A top-level module is created to tie the design components together as in the netlist. Then the components are themselves assigned.

The bottom-up approach involves just the opposite: defining and designing the individual components, then bringing the components together.

The flat approach involves a method in which the details of functional components are defined at the same level as the interconnection of those functional components. These flat designs are used for simple designs and top-down approach is used for complex designs.

SIMULATION OF SOURCE CODE

For large designs, simulating the source code with a VHDL simulator will prove time-efficient. This simulation reduces our time of designing. With source code simulation , flaws can be detected early in the design, allowing you to make corrections with the least possible impact to the schedule. Simple designs need not go for source simulation because it may lead waste of time.

TO SYNTHESIZE, OPTIMIZE AND TO FIT THE DESIGN

Now assuming that we have obtained a source code for a design, let us see about the next process of synthesizing, optimizing, and fitting the design a device.

SYNTHESIZING

To define synthesizing we may say that it is a process by which netlists or equations are created from design descriptions or saying it somewhat clear, it is a process that should be described as taking a design description as input and producing output (logic equations or netlists).

Optimizing

Though we have designed the VHDL program for our design to check whether it is a effective design, we are to go for optimization. Because some forms of expressions may be mapped to logic resources more efficiently than others. For example, whereas a minimal sum of products can be implemented efficiently in a PAL, a whereas a canonical sum of products can be mapped more efficiently to multiplexer or RAM.

Fitting

It is the process of taking the logic produced by the synthesis and optimization process and placing it into a logic device, transforming the logic to obtain the best fit.

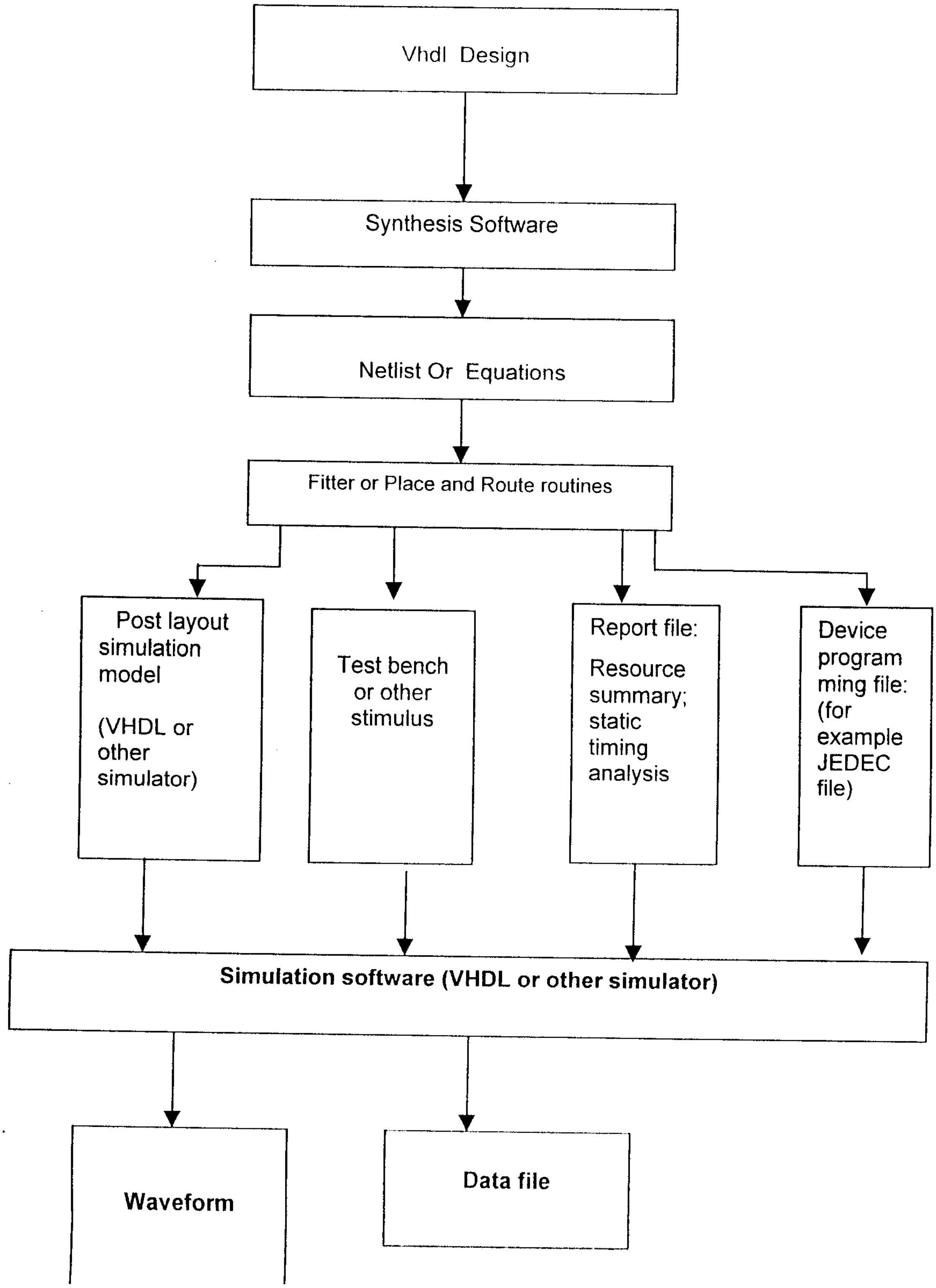
Place & Route

This have a large impact on the performance of FPGA designs. Propagation delays can depend significantly on routing delay. A good placement and route will placecritical portions of a cicuit close together to eliminate route delays.

SIMULATE THE POST LAYOUT DESIGN MODEL

Even if you have performed a presynthesis simulation, you will want to simulate your design after it has been fitted (or placed and routed). A postlayout simulation will enable you to verify not only the functionality of your design but also the timing, such as setup, clock-to-output, and register-to-register times. If you are unable to meet your design objectives, then you will need to either resynthesize and fit your design to a new logic device. You may also want to revisit your VHDL code to ensure that it has been described efficiently and in such a way as to achieve a synthesis and fitting result that meets your design objectives.

FLOWCHART EXPLAINING THE TOOL FLOW DIAGRAM


Vhdl Design
Synthesis Software
Netlist Or Equations
Fitter or Place and Route routines
Post layout simulation model (VHDL or other simulator)
Test bench or other stimulus
Report file: Resource summary; static timing analysis
Device program ming file: (for example JEDEC file)
Simulation software (VHDL or other simulator)
Waveform
Data file

PROGRAMMABLE ARRAY LOGIC

PAL (Programmable array logic) devices, or PAL' s are the simplest of programmable logic devices that are readily available in today's market. PAL's consist of an array of AND gates and an array of OR gates in which the AND array is programmable and the OR array is fixed.

Advantage of Programmable logic:

- Fewer devices are used.
- More cost effective.
- PAL implementation requires fewer total I/O's and fewer switching outputs , resulting in lower standby and switching currents.
- Integration increases design reliability.(Therefore fewer dependencies on the interconnection of devices)
- Flexibility (absent in discrete logic components)

SHORTCOMINGS IN VHDL

- The logic implementation created by the VHDL synthesis tools are inefficient.
- The quality of synthesis varies from tool to tool.

☛ We give up control of defining the gate level implementation of

☛ circuits that are described with high-level, abstract constructs.

PROGRAM STRUCTURE OF VHDL

Now we are going to enter into interesting part of VHDL. Yes, you guessed it correctly. We are going to discuss about programming structure of VHDL. Normally the programming structure of VHDL has the blocks of statements called Entity , Architecture and many other blocks that would be discussed later. Now let us see about very important blocks Entity, Architecture.

The statements in the above said blocks resembles as shown in the following sample.

ENTITY DECLARATION

```
Entity component_name is
     port(
             [signal] identifier { identifier }; [mode] signal_type
             {; [signal] identifier {,identifier}: [mode] signal_type});
end [entity] [entity_name];
```

ARCHITECTURE DECLARATION

```
architecture architecture_name of entity_name is
 type_declaration
  | signal_declaration
  | constant_declaration
  | component_declaration
  | alias_declaration
  | attribute_specification
  | subprogram_body
begin
  { process_statement
  | concurrent_signal_assignment_statement
  |component_instantiation_statement
  | generate_statement}
end [architecture] [architecture_name];
```

APPLICATIONS

IN SATELLITE COMMUNICATION

Almost twenty years before satellite weather maps appeared on TV, Satellite were looking down at earth and sending information about its surface and atmosphere to researchers on the ground. TV camera was mounted in the satellite which imaged the earth and the digitized signal was sent Bit by Bit to the earth stations that could decode it and plot the camera image on the paper

DIGITAL VIDEO EFFECTS

- ✓ Freeze Frame
- ✓ Image Compression
- ✓ Continous Image Expansion
- ✓ Image Stretching Effects
- ✓ Image Rotation Effects

COMMERCIAL APPLICATION

✓ In Medicine

✓ Banking

✓ Communication

✓ Production

✓ Forensic Science

DIGITAL VIDEO EFFECTS

The frame synchronizer unit is designed to accept any wild (non- synchronized) feed from any source, convert the signal to digital bits, synchronize the signal with in-house video and then read out processed signal in analog form, where it enters the program switches like any other video source The result is a signal that is completely synchronous and can be keyed, or resolved in the switcher.

In addition to synchronizing a remote video source the frame synchronizer offer five variable production options:

FREEZE FRAME

The frame synchronizer works by storing a complete frame of video and then reading it out in step with the in-house sync generator. The device is also programmed to held the last complete video frame and continually repeat it. In case a new signal does not curve the remote source.

IMAGE COMPRESSION

Since the frame synchronizer reads out the processed video signals in synchronization with in-house sync generator, if we vary the rate of which the stored signal is read out, we can reduce the complete picture in the raster. For example, if the unit is told to read out the stored signal twice as fast as normal the result is the image with half the normal vertical height and half the normal horizontal width. In other words, an image that is one quarter normal size using joy stick control, we can position this compressed image-which is complete video picture squeezed down to one fourth normal size-anywhere on the screen.

We can compress the image of a field reporter and position it behind the anchor's shoulder. Since the compressed image is always in synchronization. The news caster at the studio can have a two way live talk with a reporter in the field, and the compressed background image can then be faded, wiped or popped off the screen whenever necessary.

CONTINUOUS IMAGE EXPANSION

The reverse of the compression technique is continuous picture expansion, which again, appears as though we are electronically zooming in on a picture. This effect permits the director to enlarge much as a photographer might enlarge a negative in the dark room to eliminate areas along the edges of the frame which are unimportant and emphasize better the relevant picture in the video space.

IMAGE STRETCHING EFFECTS

This unit makes it possible to expand or compress either the horizontal or vertical dimension of an image independently. This means we can literally reshape a picture aspect ratio, although some distortion becomes evident after a certain amount of stretching.

IMAGE ROTATION EFFECT

We can rotate an image within the video space about all three axis using this unit. When we rotate an image on the X-axis, we create a "tumble" effect similar to viewing an object rolling from front to back rotating about the Y-axis a "spinning top" effect. Rotating along the Z-axis creates a depth perspective that gives a 3D look to the image. Some sophisticated units enable to control the depth of the perspective, so that as an object passes from one plane to another, we can control the wrapage or "key stone" effect. In the following paragraphs we, discuss some of the specific application.

COMMERCIAL APPLICATION

IN MEDICINE

With the help of video digitizer we could digitize X-ray CT scan reports and store them in a small floppy not bigger then a 4"x4" inch disk instead of carrying large reports with "DO NOT FOLD" instructions. Moreover once LAN brings it on the PC it could accessible to a large community of Doctors and their views could be sought. Thus saving time a priceless entity. A picture of size 640 x 640 pixel would need 0.4 MB of memory, it may look large, but with many compression techniques available this could be reduced by half.

BANKING

In banking industry the customer's photo, identity signature and other details could be digitized and stored. This would reduce huge files and documents into smaller records. Thus saving the energy and also the area. Moreover transactions would be easier faster and secure.

COMMUNICATION:

Now this digitizer opens new vistas in communication field. Using video cameras, image of the person speaking could be captured, digitized and transmitted, over cables. This enables the speaker not only to hear the voice of the other but also view him over his PC screen through a Modem.

PRODUCTION:

In production also the digitizer finds its applications. The product could be designed by using AUTO-CAD etc., and stored. Now the actual product could be digitized using the card and the actual product could be compared with the original design to see if there are any defects in manufacture. In this method we could get almost 100% accuracy especially for precession productions.

FORENSIC SCIENCES:

In forensic applications it would be easy for a detective to compare the fingerprints with the already available information just at the press of a few buttons against the tedious verification of the prints under the magnifying lenses.

The commercial applications are many .May be this would be faster olution to our Chief Election Commissioner's idea of issuing ID cards for the free and the fair elections. A group people could be pictured by a video camera and very individual's figure could digitized and directly printed on the ID card, instead of sticking snaps etc. This would not only save time, apart from being cheaper but also ensure that no duplication could be made.And what else? Much more than you would imagine. You could even produce a movie of your favorite Hero and Heroine without even paying them or even act with them right in your room on your PC. Is it not exciting?

CONCLUSION

Here in our project analog signal was fed into the Addon-card and then digitized using digital IC's implemented using VHDL synthesis software WARP R4, RAM & ADC. Each of the digitized signal and stored data corresponding to each frame was transferred to the internal PC memory from the RAM. These datas were then, formatted into a TIFF file with the aid of the GWS (Graphic WorkStation) tools. Finally, the formatted TIFF files were viewed.

More over, the Addon-card was designed on the basis of VHDL to support future modification. Thus the whole process was successfully accomplished and viewed on the PC screen.

Ultimately we have laid a foundation towards the implementation of VIDEO DIGITIZER using VHDL software. In future we have an idea of doing the same using an advanced ADC with an upgraded version of VHDL software.

SOURCE_CODE

```
#include<stdio.h>
#pragma inline
#include<dos.h>
#include<math.h>


/* Digitises an image */
/*uses Dos, Crt;*/

const
  dib =300,
  did =300,
  dis =301,
  StripMax =8*1024,
  BlockMax =32*1024,
  MaxBlocks =32;
 /* fn_temp ="WOSSAT",
  fn_red =fn_temp + ".RED",
  fn_green =fn_temp + ".GRN",
  fn_blue =fn_temp + ".BLU",*/
  FTemp[2] , /*{ String == (fn_red, fn_green, fn_blue);*/
  block[31];
/*int*/
  /*PBlock =^Block;*/
/* Block[31] ; { unsigned char;*/

var
  OutTIFF : String;
  RowsPerBlock, BlockSize, Blocks,
  LastBlockRows, LastBlockSize,
  RowsPerStrip, StripSize, Strips,
  GrabWidth, GrabHeight,
  OutWidth, OutHeight,
  OutSkipLines, OutSkipPix,
  SkipTop, SkipBottom, SkipLeft, SkipRight, Signal : int;
  OutSize : long int;
  OptQuiet, OptInterlace, OptEven, OptColour : int;
  ImgBlock : Array[1..MaxBlocks] ) { PBlock;


void Report(Msg: String);
/* Reports message */
{
  if ( OptQuiet ) Exit;
  if ( Msg == "" ) writeln ; else write(Msg);
};

void ReportLn(Msg: String);
/* Reports message (with newline) */
{
  Report(Msg);
  Report("");
};

 Verb(Val: int) : String;
/* Returns 'verbose' of value */
var
  s : String;
{
  Str(Val, s);
  Verb = s;
};

void Digitise;
/* Digitises image */

 WaitFrame(EVal, DVal: unsigned char) : int; assembler;
/* Waits for digitised frame part */
asm

     PUSH DS

     MOV  DX, dis   /* ^ Status lines */

     CLI            /* Disable interrupts */

     IN   AL, DX    /* Read status lines */

     TEST AL, $02   /* Check if vsync */
     JE   @@4

     MOV  BX, $01   /* Odd frame code */
     TEST AL, $04   /* Check frame */
     JE   @@1       /* It's odd */
     MOV  BX, $02   /* It's even */

@@1: IN   AL, DX    /* Wait for end of vsync */
     TEST AL, $02
     JNE  @@1
     TEST AL, $01   /* And even frame's 'border' hsync */
     JNE  @@1

     MOV  AL, EVal  /* Enable digitising */
     OUT  DX, AL

     STI            /* Re-enable interrupts */

@@3: IN   AL, DX    /* Wait for next vsync */
     TEST AL, $02
     JE   @@3
     STI

     MOV  AL, DVal  /* Disable digitising */
     OUT  DX, AL

     MOV  AX, BX    /* Frame digitised */

     JMP  @@5
```

```
@@4: STI              /* Re-enable interrupts */

     MOV  AX, $00     /* Nothing happened */

@@5: POP  DS

};

void GrabNonInterlaced(Frame: int);
/* Digitises non-interlaced image, i.e. only odd or even frames */
var
  col, bnr, brow, boff, row, off : int;
  ptrd : PBlock;
{
  for col = 0 to 63 )
    {
      bnr = 1;
      brow = 0;
      boff = 0;
      ptrd = ImgBlock[bnr];
      port[dis] = $03;
      do { } while !( WaitFrame($80 + (($20+col) && $3f), $00) == Frame ) ;
      port[dis] = $01;
      port[dis] = $00;
      port[dis] = $02;
      port[dis] = $00;
      for row = 1 to OutSkipLines )
        {
          port[dis] = $01;
          port[dis] = $00;
        };
      for row = 1 to OutHeight )
        {
          off = boff + 63 - col;
          asm

            LES  DI, ptrd          /* Get buffer base */
            MOV  AX, off           /* Get start offset */
            ADD  DI, AX            /* To first byte */

            MOV  DX, did           /* Data port */
            IN   AL, DX            /* Get samples */
            MOV  ES:[DI],AL        /* Store ... */
            IN   AL, DX
            MOV  ES:[DI+64],AL
            IN   AL, DX
            MOV  ES:[DI+128],AL
            IN   AL, DX
            MOV  ES:[DI+192],AL
            IN   AL, DX
            MOV  ES:[DI+256],AL
            IN   AL, DX
            MOV  ES:[DI+320],AL
            IN   AL, DX
            MOV  ES:[DI+384],AL
            IN   AL, DX
            MOV  ES:[DI+448],AL
            IN   AL, DX
            MOV  ES:[DI+512],AL
            IN   AL, DX
            MOV  ES:[DI+576],AL

            MOV  DX, dis            /* Status port */
            MOV  AL, $01            /* Reset line sample counter */
            OUT  DX, AL
            MOV  AL, $00
            OUT  DX, AL

          };
          brow = brow + 1;
          if ( brow == RowsPerBlock
            ) {
                   bnr = bnr + 1;
                   brow = 0;
                   boff = 0;
                   ptrd = ImgBlock[bnr];
                 }
            ; else boff = boff + GrabWidth;
        };
    };
};

void GrabInterlaced;
/* Digitises interlaced image, i.e. both odd and even frames */
var
  frame, col, bnr, brow, boff, row, off : int;
  ptrd : PBlock;
{
  for frame = 1 to 2 )
    {
      for col = 0 to 63 )
        {
          bnr = 1;
          if ( frame == 2 ) brow = 1 ; else brow = 0;
          boff = brow * GrabWidth;
          ptrd = ImgBlock[bnr];
          port[dis] = $03;
          do { } while !( WaitFrame($80 + (($20+col) && $3f), $00) == frame ) ;
          port[dis] = $01;
          port[dis] = $00;
          port[dis] = $02;
          port[dis] = $00;
          for row = 1 to OutSkipLines div 2 )
            {
              port[dis] = $01;
              port[dis] = $00;
            };
          for row = 1 to OutHeight div 2 )
            {
              off = boff + 63 - col;
              asm

                LES  DI, ptrd         /* Get buffer base */
```

```
               MOV  AX, off           /* Get start offset */
               ADD  DI, AX            /* To first byte */

               MOV  DX, did           /* Data port */
               IN   AL, DX            /* Get samples */
               MOV  ES:[DI],AL        /* Store ... */
               IN   AL, DX
               MOV  ES:[DI+64],AL
               IN   AL, DX
               MOV  ES:[DI+128],AL
               IN   AL, DX
               MOV  ES:[DI+192],AL
               IN   AL, DX
               MOV  ES:[DI+256],AL
               IN   AL, DX
               MOV  ES:[DI+320],AL
               IN   AL, DX
               MOV  ES:[DI+384],AL
               IN   AL, DX
               MOV  ES:[DI+448],AL
               IN   AL, DX
               MOV  ES:[DI+512],AL
               IN   AL, DX
               MOV  ES:[DI+576],AL

               MOV  DX, dis           /* Status port */
               MOV  AL, $01           /* Reset line sample counter */
               OUT  DX, AL
               MOV  AL, $00
               OUT  DX, AL

              };
              brow = brow + 2;
              if ( brow >= RowsPerBlock
                ) {
                       bnr = bnr + 1;
                       brow = brow - RowsPerBlock;
                       boff = brow * GrabWidth;
                       ptrd = ImgBlock[bnr];
                     }
                ; else boff = boff + GrabWidth * 2;
            };
        };
    };
};


{
  if ( OptInterlace
    ) GrabInterlaced
    ; else if ( OptEven
           ) GrabNonInterlaced(2)
           ; else GrabNonInterlaced(1);
};

void OutputFile(FType: int; OutFile: String);
/* Outputs image data to file (TIFF, temp, or merge of temp R,G,B) */
```

```
var
  Hi, Hlen : int;
  Head : Array[0..2048] ) { unsigned char;
  Tf : File;

void ResetPtr;
/* Resets header pointer */
{
  Hi = 0;
};

void SetPtr(Ptr: int);
/* Sets header pointer */
{
  Hi = Ptr;
};

void Hb(V: unsigned char);
/* Writes byte to header */
{
  Head[Hi] = V;
  Hi = Hi + 1;
};

void Hd(V: int);
/* Writes double byte to header */
{
  Head[Hi] = V && $ff;
  Head[Hi+1] = V div $100;
  Hi = Hi + 2;
};

void Hw(V: long int);
/* Writes quadbyte to header */
{
  Head[Hi] = V && $ff;
  Head[Hi+1] = (V div $100) && $ff;
  Head[Hi+2] = (V div $10000) && $ff;
  Head[Hi+3] = (V div $1000000) && $ff;
  Hi = Hi + 4;
};

void Hs(S: String);
/* Writes string to header */
var
  b : int;
{
  for b = 1 to Length(S) ) Hb(unsigned char(S[b]));
};

void HIFD(Tag, TType: int; Length, V: long int);
/* Writes TIFF IFD entry to header */
{
  Hd(Tag);
  Hd(TType);
  Hw(Length);
  Hw(V);
```

```
};

void Align16(var V: int);
/* Aligns value to 16-bit boudary */
{
  V = (V + 1) && $fffe;
};

void MakeTIFFHeader;
/* Creates TIFF file header */
var
  rowwidth, ents, di,
  ptr_strp, ptr_strb, ptr_xres, ptr_yres, ptr_sig, ptr_samp,
  pstart, strip : int;
  off : long int;
  sig : String;
{
  if ( OptColour
    ) {
           rowwidth = 3 * OutWidth;
           ents = 15;
         }
    ; else {
           rowwidth = OutWidth;
           ents = 14;
         };
  RowsPerStrip = StripMax div rowwidth;
  StripSize = RowsPerStrip * rowwidth;
  Strips = (OutHeight + RowsPerStrip - 1) div RowsPerStrip;
*********************************************************************************
*********************************************************************************
*********************************************************************************
*********************************************************************************
*********************************************************************************
*********************************************************************************
*********************************,3,3,ptr_samp);    /* 8 bits per sample (R,G,B)
*/
           di = di + 2 * 3;
         }
    ; else HIFD(258,3,1,8);             /* 8 bits per sample */
  HIFD(259,3,1,1);                    /* No compression */
  if ( OptColour
    ) HIFD(262,3,1,2)               /* Photometric interpretation 'RGB' */
    ; else HIFD(262,3,1,1);             /* Photometric interpretation 'greyscale'
*/
  HIFD(266,3,1,1);                    /* Big endian fill order */
  ptr_strp = di;
  HIFD(273,4,Strips,ptr_strp);        /* Strip offsets */
  di = di + 4 * Strips;
  if ( OptColour
    ) HIFD(277,3,1,3)               /* 3 samples per pixel (R,G,B) */
    ; else HIFD(277,3,1,1);             /* 1 sample per pixel */
  HIFD(278,4,1,RowsPerStrip);         /* Rows per strip */
  ptr_strb = di;
  HIFD(279,3,Strips,ptr_strb);        /* Strip bytecounts */
  di = di + 2 * Strips;
  ptr_xres = di;
  HIFD(282,5,1,ptr_xres);              /* XResolution */
  di = di + 8;
  ptr_yres = di;
  HIFD(283,5,1,ptr_yres);              /* YResolution */
  di = di + 8;
  if ( OptColour
    ) HIFD(284,3,1,1);              /* Planar configuration RGB triples */
  HIFD(296,3,1,2);                     /* Resolution unit = inch */
  ptr_sig = di;
  HIFD(305,2,Length(sig)+1,ptr_sig); /* Signature */
  di = di + Length(sig)+1;
  Hw(0);                               /* End of IFD */
  pstart = di;
  Align16(pstart);
  if ( OptColour
    ) {
           SetPtr(ptr_samp);
           Hd(8);
           Hd(8);
           Hd(8);
         };
  SetPtr(ptr_strp);
  off = pstart;
  for strip = 1 to Strips )
    {
      Hw(off);
      off = off + StripSize;
    };
  SetPtr(ptr_strb);
  for strip = 1 to Strips-1 ) Hd(StripSize);
  Hd(OutSize-(Strips-1)*StripSize);
  SetPtr(ptr_xres);
  Hw(100);
  Hw(1);
  SetPtr(ptr_yres);
  Hw(100);
  Hw(1);
  SetPtr(ptr_sig);
  Hs(sig);
  Hb(0);
  Align16(di);
  if ( di != pstart
    ) writeln("Internal error : Misallocation in TIFF header !");
  HLen = pstart;
};

void WriteImage;
/* Writes out image data */
var
  bnr, bytes, lines, line, pd, pl, i : int;
  ptrd : PBlock;
{
  if ( GrabWidth == OutWidth
    ) {
           for bnr = 1 to Blocks )
             {
               if ( bnr != Blocks
                  ) bytes = BlockSize
                  ; else bytes = LastBlockSize;
                BlockWrite(Tf, ImgBlock[bnr]^, bytes);
              };
          }
    ; else {
            for bnr = 1 to Blocks )
              {
                ptrd = ImgBlock[bnr];
                if ( bnr != Blocks
                  ) lines = RowsPerBlock
                  ; else lines = LastBlockRows;
                pl = 0;
                pd = OutSkipPix;
                for line = 1 to lines )
                  {
                    for i = pl to pl+OutWidth-1 ) ptrd^[i] = ptrd^[i+pd];
                    pl = pl + OutWidth;
                    pd = pd + GrabWidth - OutWidth;
                  };
                BlockWrite(Tf, ptrd^, lines * OutWidth);
              };
          };
};

void MergeToColour;
/* Merges R,G and B temp files into colour TIFF file */
var
  tfr, tfg, tfb : File;
  blkr, blkg, blkb, blko : PBlock;
  szo, szt, numread, ptro, i : int;
{
  blkr = ImgBlock[1];
  blkg = ImgBlock[2];
  blkb = ImgBlock[3];
  blko = ImgBlock[4];
  szt = SizeOf(blko^) div 3;
  szo = szt * 3;
  Assign(tfr, FTemp[0]);
  Reset(tfr, 1);
  Assign(tfg, FTemp[1]);
  Reset(tfg, 1);
  Assign(tfb, FTemp[2]);
  Reset(tfb, 1);
  do {
  BlockRead(tfr, blkr^, szt, numread);
  BlockRead(tfg, blkg^, szt, numread);
  BlockRead(tfb, blkb^, szt, numread);
  ptro = 0;
  for i = 0 to numread-1 )
    {
      blko^[ptro] = blkr^[i];
      blko^[ptro+1] = blkg^[i];
      blko^[ptro+2] = blkb^[i];
      ptro = ptro + 3;
  . };
 BlockWrite(Tf, blko^, numread * 3);
```

```
  } while !( (numread < szt) || Eof(tfr) ) ;
  Close(tfr);
  Close(tfg);
  Close(tfb);
  Erase(tfr);
  Erase(tfg);
  Erase(tfb);
 };

 {
  Assign(Tf, OutFile);
  Rewrite(Tf, 1);
  switch ( FType ) {
   1 : {
          MakeTIFFHeader;
          BlockWrite(Tf, Head, Hlen);
          WriteImage;
        };
   2 : WriteImage;
   3 : {
          MakeTIFFHeader;
          BlockWrite(Tf, Head, Hlen);
          MergeToColour;
        };
  };
  Close(Tf);
 };

 Init : int;
/* Initialise */
var
  Go : int;

void SyntaxMsg;
/* Displays syntax message */
{ writeln;
  writeln("Usage : PCDigi [{-|/}<switch>...] <filename>");
  writeln;
  writeln("<Switches>");
  writeln("  ?: Display this message");
  writeln("  n: Non-interlaced (default, odd or even frame only, 640x320)");
  writeln("  i: Interlaced (both odd and even frames, 640x640)");
  writeln("  o: Odd frame (default, ignored for -i)");
  writeln("  e: Even frame (ignored for -i)");
  writeln("  v: Verbose, give messages (default)");
  writeln("  q: Quiet, no messages");
  writeln("  g: Digitise greyscale image.(default, 1 pass)");
  writeln("  c: Digitise 24-bit RGB colour image (3 passes), -s ignored");
  writeln("  t<val>: Skip top <val> lines in output image (default 0)");
  writeln("  b<val>: Skip bottom <val> lines in output image (default 0)");
  writeln("  l<val>: Skip left <val> pixels in output image (default 0)");
  writeln("  r<val>: Skip right <val> pixels in output image (default 0)");
  writeln("  s<val>: Digitise signal <val> (0=R,1=G,2=B,3=B/W, default 3)");
  Go = 0;
};

void ErrMsg(ErrMsg: String);
```

```
/* Displays error message & syntax */
{
  writeln(ErrMsg, " !");
  writeln;
  SyntaxMsg;
};

 ReadParams : int;
/* Reads commandline parameters */

void ArgVal(S: String; var V: int; MaxLen: int);
/* Reads argument value */
var
  a : String;
  c : int;
{
  a = Copy(S, 3, MaxLen);
  Val(a, V, c);
};

var
  ps : String;
  pc, params, c : int;
{
  OutTIFF = "";
  OptQuiet = 0;
  OptInterlace = 0;
  OptEven = 0;
  OptColour = 0;
  SkipTop = 0;
  SkipBottom = 0;
  SkipLeft = 0;
  SkipRight = 0;
  Signal = 3;
  params = ParamCount;
  pc = 1;
  while ( (pc <= params) && Go )
    {
      ps = ParamStr(pc);
      if ( (ps[1] == "-") || (ps[1] == "/")
        ) {
               switch ( ps[2] ) {
                 "?" : SyntaxMsg;
                 "q" : OptQuiet = 1;
                 "v" : OptQuiet = 0;
                 "i" : OptInterlace = 1;
                 "n" : OptInterlace = 0;
                 "e" : OptEven = 1;
                 "o" : OptEven = 0;
                 "c" : OptColour = 1;
                 "g" : OptColour = 0;
                 "t" : ArgVal(ps, SkipTop, 4);
                 "b" : ArgVal(ps, SkipBottom, 4);
                 "l" : ArgVal(ps, SkipLeft, 4);
                 "r" : ArgVal(ps, SkipRight, 4);
                 "s" : ArgVal(ps, Signal, 1);
               ; else ErrMsg("Unknown switch """ + ps[2] + """");
               };
             }
        ; else if ( pc == params
               ) {
                      for c = 1 to Length(ps) ) ps[c] = UpCase(ps[c]);
                      OutTIFF = ps;
                    }
               ; else ErrMsg("Trailing junk");
      pc = pc + 1;
    };
  if ( params == 0 ) SyntaxMsg;
  if ( (params > 0) && Go && (OutTIFF == "") ) ErrMsg("Missing filename");
  ReadParams = Go;
};

var
  bnr, cnr : int;
  fdir : DirStr;
  fname : NameStr;
  fext : ExtStr;
{
  Init = 0;
  Go = 1;
  if ( ! ReadParams ) Exit;
  if ( OptInterlace ) GrabHeight = 640 ; else GrabHeight = 320;
  OutHeight = GrabHeight - SkipTop - SkipBottom;
  if ( OutHeight <= 0
    ) {
           ErrMsg("No output height left, lower -t/-b values");
           Exit;
         };
  if ( OptInterlace && (((SkipTop mod 2) + (SkipBottom mod 2) + (OutHeight mod
2)) > 0)
    ) {
           ErrMsg("Resulting height, -t and -b values must be even for -i");
           Exit;
         };
  GrabWidth = 640;
  OutWidth = GrabWidth - SkipLeft - SkipRight;
  if ( OutWidth <= 0
    ) {
           ErrMsg("No output width left, lower -l/-r values");
           Exit;
         };
  OutSkipLines = SkipTop;
  OutSkipPix = SkipLeft;
  if ( OptColour
    ) OutSize = OutWidth * OutHeight * 3
    ; else OutSize = OutWidth * OutHeight;
  RowsPerBlock = BlockMax div GrabWidth;
  BlockSize = RowsPerBlock * GrabWidth;
  Blocks = (OutHeight + RowsPerBlock - 1) div RowsPerBlock;
  LastBlockRows = OutHeight - (Blocks - 1) * RowsPerBlock;
  LastBlockSize = LastBlockRows * GrabWidth;
  if ( Blocks < 4
    ) cnr = 4
    ; else cnr = Blocks;
  for bnr    1 to cnr ) ImgBlock[bnr]    New(PBlock);
  FSplit(OutTIFF, fdir, fname, fext);
  OutTIFF = fdir + fname +
".TIF";[illegible]
[illegible]ctions.
```

FORENSIC SCIENCES:

In forensic applications it would be easy for a detective to compare the fingerprints with the alrea[illegible]
[illegible]lenses.

The commercial applications are many .May be this would be faster olution to our Chief Election Commissioner's idea of gnal " + Verb(Signal) + " (");

```
  switch ( Signal ) {
    0 : Report("Red)");
    1 : Report("Green)");
    2 : Report("Blue)");
    3 : Report("B/w)");
  };
  Report(" ...");
};

void ShowWriting(What, Name: String);
/* Shows 'writing' info */
{
  Report("Writing " + What + " """ + Name + """, ");
  Report(Verb(OutWidth) + "x" + Verb(OutHeight) + " ...");
};


type
  q == array[0..1024] ) { unsigned char;

void zap(var z:q); assembler;
asm
            CLI
            LES  DI, z          /* Get buffer base */

            MOV  DX, dis        /* Status port */
            MOV  CX, $100
@@1:        IN   AL, DX         /* Get samples */
            MOV  ES:[DI],AL     /* Store ... */
            INC  DI
            IN   AL, DX
            MOV  ES:[DI],AL
            INC  DI
            IN   AL, DX
            MOV  ES:[DI],AL
            INC  DI
            IN   AL, DX
            MOV  ES:[DI],AL
            INC  DI
            DEC  CX
            JNZ  @@1
            STI
```

```
};

var
  s : String;
  p : int;
{
  if ( Init
    ) {
            if ( OptColour
              ) {
                     for p = 0 to 2 )
                       {
                         s = FTemp[p];
                         ShowDigitising(p);
                         SwitchSignal(p);
                         Digitise;
                         Report("");
                         ShowWriting("temp file", s);
                         OutputFile(2, s);
                         Report("");
                       };
                     ShowWriting("TIFF file", OutTIFF);
                     OutputFile(3, OutTIFF);
                     Report("");
                   }
              ; else {
                     ShowDigitising(Signal);
                     SwitchSignal(Signal);
                     Digitise;
                     Report("");
                     ShowWriting("TIFF file", OutTIFF);
                     OutputFile(1, OutTIFF);
                     Report("");
                   };
          };
 }.
 ||
```

COUNTER

```
PACKAGE cout_types is
  TYPE bit8 IS range 0 to 255;
END cout_types;
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
 USE WORK.std_logic_1164.ALL;
 USE WORK.cout_types.ALL;
 USE work.numeric_std.ALL;
  ENTITY count IS
  PORT (clk,ld,clr,e : IN std_logic;
           din: in bit8;
           dout: INOUT  bit8);
          -- count : OUT unsigned (13 DOWNTO 0));
  END count;

ARCHITECTURE synth OF count IS
  SIGNAL count_val : bit8;
BEGIN
   PROCESS(ld,clr,e,din,dout)
BEGIN
  --if(e='1')then
 IF (ld = '0') THEN
     count_val <= din;
  ELSIF (clr = '1') THEN
     count_val <= 0;
  ELSIF (dout >= 16) THEN
     count_val<= 0;

   else
      if(e='1')then
      count_val <= dout + 1;
      else
      count_val<=0;
      end if;


END IF;
END PROCESS;
```

```
PROCESS
 BEGIN
  WAIT UNTIL clk'event and clk = '1' ;
   if(e='1')then
   dout<= count_val;
 end if;
END PROCESS;
END synth;
```

BUFFER

```
PACKAGE cut_types is
  TYPE bit8 IS range 0 to 255;
END cut_types;

library ieee;
use ieee.std_logic_1164.all;
USE WORK.cut_types.ALL;

Entity buf is
port ( a_2: out bit8;
       b_2: in bit8;
       g,e1: in std_logic);
end buf;

Architecture bufer of buf is

begin
  process (g)
begin

  if(e1='0') then
  if(g='0') then
    a_2<=b_2;
end if;
end if;
end process;
end bufer;
```

COUNTER

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use work.numeric_std.ALL;

Entity cnbs_4 is
port( clk,rst: in std_logic;

        cout: out unsigned( 8 downto 0));
end cnbs_4;
Architecture cbs_4 of cnbs_4 is
--signal a: std_logic_vector(13 downto 0);
-- signal cout :  unsigned (13 DOWNTO 0);

begin
process (rst,clk)
 begin
   if(rst = '1') then
    cout <= "000000000";
   elsif(clk'event and clk = '1') then
    -- a<="0001";
    cout <= cout + 1;
   end if;
 end process;
end cbs_4;
```

COMPARATOR & DECODER

```
Library ieee ;
   Use ieee.std_logic_1164.ALL;
   use work.numeric_std.ALL;
   use work.std_arith.ALL;
 --  use work.mnemonics.all;
Entity comp is
port(s1,s2 : in std_logic_vector (7 downto 0);
         f: in std_logic;
        o: out std_logic);
end comp;
Architecture com of comp is
begin
process(s1,s2,f)
 begin
    if(f='0') then
       if(s1=s2) then
         o<='0';
        end if;
     end if;
  end process;
end com;

Library ieee ;
  Use ieee.std_logic_1164.ALL;
  use work.numeric_std.ALL;
  use work.std_arith.ALL;

       Entity decoder is
       port (  a,b,c :in std_logic;
        y_1,y_2,y_5,y_6: out std_logic);
        end decoder;

        Architecture decoder of decoder is
        begin
           y_1<='0' when (a='0' and b='0' and c='1') else '0';
           y_2<= '0' when(a='0' and b='1' and c='0' ) else '0';
           y_5<= '0' when (a='1' and b='0' and c='1') else '0';
           y_6<='0' when  (a='1' and b='1' and c='0')  else '0';
            end decoder;
Library ieee;
Use ieee.std_logic_1164.ALL;
use work.numeric_std.all;
use work.std_arith.all;
entity top is
port( f,a,b,c: in std_logic;
    s1,s2: in std_logic_vector(7 downto 0);
    o,y_1,y_2,y_5,y_6: out std_logic);

end top;
Architecture top_level of top is
component comp
port(s1,s2 : in std_logic_vector (7 downto 0);
        f: in std_logic;
       o: out std_logic);
end component;

component decoder
port( a,b,c: in std_logic;
       y_1,y_2,y_5,y_6: out std_logic);
end component ;

for u2:decoder use entity work.decoder(decoder);
for u1:comp use entity work.comp(com);
```

```
begin
u1:comp port map(s1,s2,f,o);
u2: decoder port map(a,b,c,y_1,y_2,y_5,y_6);
end;
```

LATCH

```
 Library ieee ;
 Use ieee.std_logic_1164.ALL;
use work.numeric_std.ALL;
  use work.std_arith.ALL;
Entity latch is
port(clk,e2 : in std_logic;
     d:in std_logic_vector(7 downto 0);
     q: out std_logic_vector(7 downto 0));
end latch;
   Architecture lat of latch is
          begin
         process(clk,d)
         begin
    if e2='0' then
      if clk='1' then
         q<=d;
       end if;
    end if;
end process;
 end lat;
```

MULTIPLEXER & COUNTER

```
 Library ieee ;
 Use ieee.std_logic_1164.ALL;
use work.numeric_std.ALL;
 use work.std_arith.ALL;
entity mux1 is
port (e3,sel: in std_logic;
     a1,b1,a2,b2,a3,b3,a4,b4:in std_logic;
     y1,y2,y3,y4: out std_logic);
end mux1;
Architecture muxx of mux1 is
begin
process (e3,sel,a1,b1,a2,b2,a3,b3,a4,b4)
begin
if(e3='0')then
if(sel='0')then
 if(a1='0') then
y1<='1';
else y1<='0';
end if;
 if(a2='0') then
y2<='1';
else y2<='0';
end if;
 if(a3='0') then
y3<='1';
else y3<='0';
end if;
 if(a4='0') then
y4<='1';
```

```
else y4<='0';
end if;
else
if(b1='0')then
y1<='1';
else
y1<='0';
end if;
if(b2='0')then
y2<='1';
else
y2<='0';
end if;
if(b3='0')then
y3<='1';
else
y3<='0';
end if;
if(b4='0')then
y4<='1';
else
y4<='0';
end if;
end if;
end if;
end process;
end muxx;
Library ieee;
Use ieee.std_logic_1164.ALL;
Use work.numeric_std.ALL;

Entity cnbs_4 is
port( clk,rst: in std_logic;
      cout: out unsigned( 3 downto 0));
end cnbs_4;
Architecture cbs_4 of cnbs_4 is
--signal a: std_logic_vector(13 downto 0);
-- signal cout :  unsigned (13 DOWNTO 0);

begin
process (rst,clk)
 begin
   if(rst = '1') then
    cout <= "0000";
   elsif(clk'event and clk = '1') then
    -- a<="0001";
     cout <= cout + 1;
   end if;
 end process;
end cbs_4;



Library ieee;
Use ieee.std_logic_1164.ALL;
use work.numeric_std.all;
use work.std_arith.all;
entity top is
port (e3,sel,a1,b1,a2,b2,a3,b3,a4,b4,clk,rst: in std_logic;

      cout: out unsigned( 3 downto 0);
       y1,y2,y3,y4: out std_logic);
end top;
Architecture top_level of top is
component mux1
port (e3,sel,a1,b1,a2,b2,a3,b3,a4,b4: in std_logic;
      y1,y2,y3,y4: out std_logic);
```

```
end component;
component cnbs_4
port( clk,rst: in std_logic;
        -- cut: out unsigned (13 downto 0));
    cout: out unsigned( 3 downto 0));
end component;
for u2:cnbs_4 use entity work.cnbs_4(cbs_4);
for u1: mux1 use entity work.mux1(muxx);
begin
u1:mux1 port map(e3,sel,a1,b1,a2,b2,a3,b3,a4,b4,y1,y2,y3,y4);
u2: cnbs_4 port map(clk,rst,cout);
end;
```

INTERGATION OF ALL DIGITAL IC'S

```
Library ieee ;
   Use ieee.std_logic_1164.ALL;
   use work.numeric_std.ALL;
   use work.std_arith.ALL;
--   use work.mnemonics.all;
Entity comp is
port(s1,s2 : in std_logic_vector (7 downto 0);
        f: in std_logic;
       o: inout std_logic);
end comp;
Architecture com of comp is
begin
process(s1,s2,f)
 begin
    if(f='0') then
       if(s1=s2) then
         o<='0';
       end if;
    end if;
  end process;
end com;

Library ieee ;
   Use ieee.std_logic_1164.ALL;
   use work.numeric_std.ALL;
   use work.std_arith.ALL;

       Entity decoder is
       port (  a,b,c :in std_logic;
             o:inout std_logic;
        y_1,y_2,y_5,y_6: out std_logic);
       end decoder;

       Architecture decode of decoder is
       begin
          y_1<='0' when (a='0' and b='0' and c='1'and o='0') else '0';
          y_2<= '0' when(a='0' and b='1' and c='0'and o='0' ) else '0';
          y_5<= '0' when (a='1' and b='0' and c='1'and o='0') else '0';
          y_6<='0' when  (a='1' and b='1' and c='0'and o='0')  else '0';
          end decode;
 PACKAGE cut_types is
  TYPE bit16 IS range 0 to 65536;
 END cut_types;

 library ieee;
 use ieee.std_logic_1164.all;
 USE WORK.cut_types.ALL;

 Entity buf is
 port ( a_2: out bit16;
      b_2: in bit16;
      g,e1: in std_logic);
 end buf;

 Architecture bufer of buf is

 begin
  process (g)
 begin

  if(e1='0') then
  if(g='0') then
   a_2<=b_2;
 end if;
 end if;
 end process;
 end bufer;

 Library ieee ;
 Use ieee.std_logic_1164.ALL;
use work.numeric_std.ALL;
 use work.std_arith.ALL;
Entity latch is
port(clk : in std_logic;
     d:in std_logic_vector(7 downto 0);
     q: out std_logic_vector(7 downto 0));
end latch;
  Architecture lat of latch is
         begin
        process(clk,d)
        begin
 if clk='1' then
        q<=d;
     end if;
  end process;
end lat;

Library ieee ;
Use ieee.std_logic_1164.ALL;
use work.numeric_std.ALL;
 use work.std_arith.ALL;
entity mux1 is
port (e3,sel: in std_logic;
     a1,b1,a2,b2,a3,b3,a4,b4:in std_logic;
     y1,y2,y3,y4: out std_logic);
end mux1;
Architecture muxx of mux1 is
begin
process (e3,sel,a1,b1,a2,b2,a3,b3,a4,b4)
begin
if(e3='0')then
if(sel='0')then
 if(a1='0') then
y1<='1';
else y1<='0';
end if;
 if(a2='0') then
y2<='1';
else y2<='0';
end if;
 if(a3='0') then
y3<='1';
else y3<='0';
end if;
 if(a4='0') then
y4<='1';
else y4<='0';
end if;
else
if(b1='0')then
y1<='1';
else
y1<='0';
end if;
if(b2='0')then
y2<='1';
else
y2<='0';
end if;
if(b3='0')then
y3<='1';
```

```
  else
  y3<='0';
  end if;
  if(b4='0')then
  y4<='1';
  else
  y4<='0';
  end if;
  end if;
  end if;
  end process;
  end muxx;

  PACKAGE ut_types is
    TYPE bit8 IS range 0 to 255;
  END ut_types;
  LIBRARY ieee;
  USE ieee.std_logic_1164.ALL;
   USE WORK.std_logic_1164.ALL;
   USE WORK.ut_types.ALL;
   USE work.numeric_std.ALL;
   ENTITY count IS
   PORT (e,clk,ld,clr,rst : IN std_logic;
             din: in bit8;
             dout: INOUT  bit8);
            -- count : OUT unsigned (13 DOWNTO 0));
  END count;

 ARCHITECTURE synth OF count IS
   SIGNAL count_val : bit8;
 BEGIN
    PROCESS(e,ld,clr,din,dout)
 BEGIN
  IF (ld = '1') THEN
      count_val <= din;
  ELSIF (clr = '1') THEN
      count_val <= 0;
  ELSIF (dout >= 255) THEN
      count_val<= 0;
  ELSE
      if(e='1')then
      count_val <= dout + 1;
      end if;
  END IF;
END PROCESS;
```

```
  PROCESS
   BEGIN
    WAIT UNTIL clk'event and clk = '1';
     if(e='1')then
     dout<= count_val;
     end if;
    END PROCESS;
  END synth;

  Library ieee;
  Use ieee.std_logic_1164.ALL;
  Use work.numeric_std.ALL;

  Entity cnbs_4 is
  port( clk,rst: in std_logic;

        cout: out unsigned( 11 downto 0));
  end cnbs_4;
  Architecture cbs_4 of cnbs_4 is
  --signal a: std_logic_vector(13 downto 0);
  -- signal cout :  unsigned (13 DOWNTO 0);

  begin
  process (rst,clk)
   begin
     if(rst = '1') then
      cout <= "000000000000";
     elsif(clk'event and clk = '1') then
      -- a<="0001";
      cout <= cout + 1;
     end if;
   end process;
  end cbs_4;

 PACKAGE cot_types is
   TYPE bit8 is range 0 to 255;
 end cot_types;
 PACKAGE ct_types is
   TYPE bit16 IS range 0 to 65536;
 END ct_types;



Library ieee;
Use ieee.std_logic_1164.ALL;
USE WORK.cot_types.ALL;
```

```
 use work.ct_types.all;
 use work.numeric_std.all;
 use work.std_arith.all;
 entity top is
  port( a,b,c,f,g,e1,e,e3,sel,clk,ld,clr,rst: in std_logic;
          s1,s2,d        : in std_logic_vector(7 downto 0);
            a1,b1,a2,b2,a3,b3,a4,b4:in std_logic;
            y1,y2,y3,y4:out std_logic;
            a_2:out bit16;
            b_2:in bit16;
            din : in bit8;
            dout: inout  bit8;
            --cut: out unsigned(13 downto 0);
     cout: out unsigned( 11 downto 0);
        o:inout std_logic;
  y_1,y_2,y_5,y_6: out std_logic;
         q        :out std_logic_vector(7 downto 0));
 end top;
 Architecture top_level of top is
 component comp
 port(s1,s2 : in std_logic_vector (7 downto 0);
         f: in std_logic;
       o: inout std_logic);
 end component;

 component decoder
 port( a,b,c: in std_logic;
                o:inout std_logic;
     y_1,y_2,y_5,y_6: out std_logic);
 end component ;

 component buf
 port ( a_2: out bit16;
      b_2: in bit16;
      g,e1: in std_logic);
 end component;
 component latch
 port( clk:in std_logic;
 d:in std_logic_vector(7 downto 0);
 q : out std_logic_vector(7 downto 0));
 end component ;

 component mux1
 port (e3,sel: in std_logic;
      a1,b1,a2,b2,a3,b3,a4,b4:in std_logic;
      y1,y2,y3,y4: out std_logic);
```

```
end component;

component count
port (clk,ld,clr,rst,e : in std_logic;
          din: in bit8;
          dout: inout  bit8);
end component;
component cnbs_4
port( clk,rst: in std_logic;
      -- cut: out unsigned (13 downto 0));
cout: out unsigned( 11 downto 0));
end component;

for u7: cnbs_4  use entity work.cnbs_4(cbs_4);
for u6: count use entity work.count(synth);
for u5: mux1 use entity work.mux1(muxx);
for u4:latch use entity work.latch(lat);
for u3:buf use entity work.buf(bufer);
for u2:decoder use entity work.decoder(decode);
for u1:comp use entity work.comp(com);
begin
u1:comp port map(s1,s2,e,o);
u2: decoder port map(a,b,c,o,y_1,y_2,y_5,y_6);
u3:buf port map(a_2,b_2,g,e1);
u4: latch port map(clk,d,q);
u5:mux1 port map(e3,sel,a1,b1,a2,b2,a3,b3,a4,b4,y1,y2,y3,y4);
u6: count port map(clk,ld,clr,rst,e,din,dout);
u7: cnbs_4 port map(clk,rst,cout);

end*****
```

22V10

The 22V10 architecture is shown in block diagram form in figure for the purpose of assigning terms to the various blocks or features of the 22V10 architecture. We will use these terms in our discussion of CPLD's. The term logic array inputs or simply inputs refers to all of the signals that are inputs to the logic array. These inputs include dedicated device inputs, feedbacks from macrocells, and inputs from I/Os that are configured as device inputs. The term product-term array describes the programmable AND gates (product terms). The product-term distribution scheme is the mechanism for distributing product terms to the macrocells. Macrocells typically contain a register (flip-flop) and combinational path with polarity control as well as one or more feedback Paths. I/O cells is a term used to describe the structure of the I/O buffers and flexibility of the output enable controls.

Some logic expressions require more product terms per OR gate than others, the architects of the 22V10 (the V is for variable) varied the number of product terms per OR gate from 8 to 16 (Figure 2-13). The top and bottom macrocells are allocated 8 product terms, the middle macrocells are allocated 16 product terms, and the others are allocated 10, 12, or 14, depending on location. The 22 in 22V10 is for the 22 inputs to the logic array; the 10 is for the 10 outputs.

Why did the architects of the 22V10 not allocate 16 product terms to every macrocell? The most likely reason is that doing so would increase the cost to manufacture the device (the increased die size would result in fewer dice per wafer and, therefore, higher unit costs). Because many applications do not require more product terms, the additional product terms per macrocell would usually go unused, in which case the additional cost would not provide additional functionality.

National Semiconductor

MM54HC688/MM74HC688
8-Bit Magnitude Comparator (Equality Detector)

General Description

This equality detector utilizes microCMOS™ Technology, 3.5 micron silicon gate P-well CMOS, to compare bit for bit two 8-bit words and indicates whether or not they are equal. The $\overline{P=Q}$ output indicates equality when it is low. A single active low enable is provided to facilitate cascading of several packages and enable comparison of words greater than 8 bits.

This device is useful in memory block decoding applications, where memory block enable signals must be generated from computer address information.

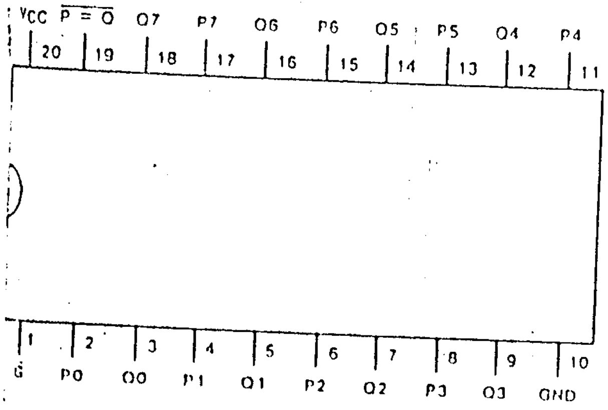
The comparator's output can drive 10 low power Schottky equivalent loads. This comparator is functionally and pin compatible to the 54LS688/74LS688. All inputs are protected from damage due to static discharge by diodes to V_{CC} and ground.

Features

- Typical propagation delay: 20 ns
- Wide power supply range: 2–6V
- Low quiescent current: 80 μA (74 series)
- Large output current: 4 mA (74 series)

Connection and Logic Diagrams

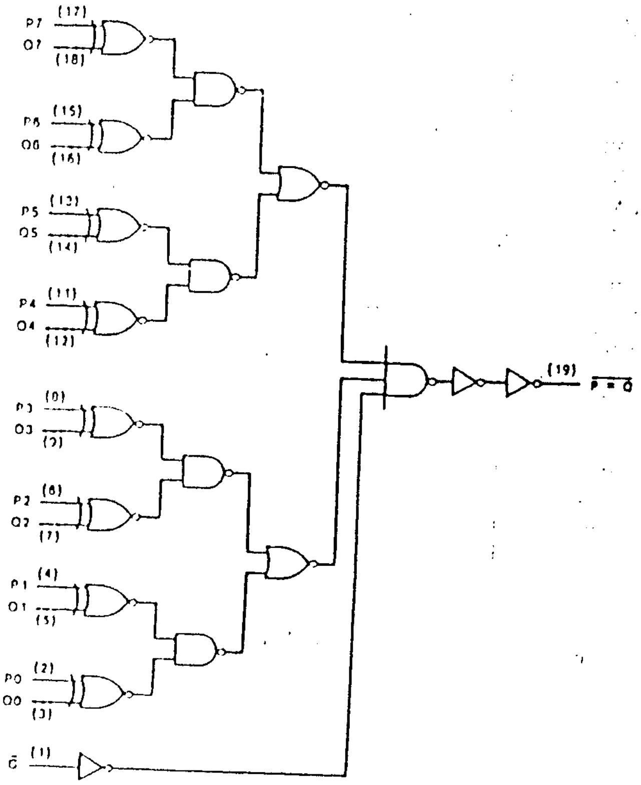
Dual-In-Line Package



TL/F/5018-1

MM54HC688/MM74HC688

54HC688 (J) 74HC688 (J,N)



TL/F/5018-2

Truth Table

Inputs		
Data	Enable	
P,Q	$\overline{G}$	$\overline{P=Q}$
P = Q	L	L
P > Q	L	H
P < Q	L	H
X	H	H

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC}+1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC}+0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage(V_{CC})	4.5	5.5	V
DC Input or Output Voltage (V_{IN},V_{OUT})	0	V_{CC}	V
Operating Temperature Range(T_A)			
MM74HCT688	−40	+85	°C
MM54HCT688	−55	+125	°C
Input Rise or Fall Times (t_r, t_f)		500	ns

DC Electrical Characteristics

(V_{CC}=5V ±10% unless otherwise specified)

Symbol	Parameter	Conditions	T_A=25°C		74HCT T_A=−40 to 85°C	54HCT T_A=−55 to 125°C	Units
			Typ	Guaranteed Limits			
V_{IH}	Minimum High Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum Low Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum High Level Output Voltage	V_{IN}=0.8V or 2.0V $\|I_{OUT}\|$=20 μA	V_{CC}	V_{CC}−.1	V_{CC}−.1	V_{CC}−.1	V
		$\|I_{OUT}\|$=4.0 mA, V_{CC}=4.5V	4.2	3.98	3.84	3.7	V
		$\|I_{OUT}\|$=4.8 mA, V_{CC}=5.5V	5.7	4.98	4.84	4.7	V
V_{OL}	Maximum Low Level Voltage	V_{IN}=V0.8V or 2.0V $\|I_{OUT}\|$=20 μA	0	0.1	0.1	0.1	V
		$\|I_{OUT}\|$=4.0 mA, V_{CC}=4.5V	0.2	0.26	0.33	0.4	V
		$\|I_{OUT}\|$=4.8 mA, V_{CC}=5.5V	0.2	0.26	0.33	0.4	V
I_{IN}	Maximum Input Current	V_{IN}=V_{CC} or GND		±0.1	±1.0	±1.0	μA
I_{CC}	Maximum Quiescent Supply Current	V_{IN}=V_{CC} or GND I_{OUT}=0 μA		8.0	80	160	μA
		V_{IN}=2.4V or 0.4V (Note 4)	100				μA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: Measured per pin. All other inputs held at V_{CC} or ground.

AC Electrical Characteristics

$V_{CC}=5V$, $T_A=25°C$, $C_L=15$ pF, $t_r=t_f=6$ ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PHL}, t_{PLH}	Maximum Propagation Delay - P or Q to output		19	30	ns
t_{PHL}	Maximum Propagation Delay - Enable to output		13	20	ns
t_{PHL}	Maximum Propagation Delay - Enable to output		10	18	ns

AC Electrical Characteristics

$V_{CC}=5V$ ± 10%, $C_L=50$ pF, $t_r=t_f=6$ ns (unless otherwise specified)

Symbol	Parameter	Conditions	$T_A=25°C$		74HCT $T_A=-40$ to 85°C	64HCT $T_A=-55$ to 125°C	Units
			Typ		Guaranteed Limits		
t_{PHL}, t_{PLH}	Maximum Propagation Delay - P or Q to output		22	34	42	51	ns
t_{PHL}	Maximum Propagation Delay - Enable to output		16	23	29	35	ns
t_{PLH}	Maximum Propagation Delay - Enable to output		13	21	26	32	ns
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time		8	15	19	22	ns
C_{PD}	Power Dissipation Capacitance (Note 5)			60			pF
C_{IN}	Maximum Input Capacitance		5	10	10	10	pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D=C_{PD}V_{CC}^2f+I_{CC}V_{CC}$, and the no load dynamic current consumption, $I_S=C_{PD}V_{CC}+I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74HCT AC Switching Waveforms and Test Circuits.

National Semiconductor

PRELIMINARY

M54HCT138/MM74HCT138 3-to-8 Line·Decoder

neral Description

coder utilizes microCMOS™ Technology, 3.0 micron n gate N-well CMOS, and are well suited to memory ss decoding or data routing applications. Both circuits e high noise immunity and low power consumption y associated with CMOS circuitry, yet have speeds arable to low power Schottky TTL logic.

MM54HCT138/MM74HCT138 have 3 binary select in- A, B, and C). If the device is enabled these inputs mine which one of the eight normally high outputs will . Two active low and one active high enables (G1, nd G2B) are provided to ease the cascading de- i.

coders' output can drive 10 low power Schottky TTL ent loads and are functionally and pin equivalent to LS138/74LS138. All inputs are protected from dam- e to static discharge by diodes to V_{CC} and ground.

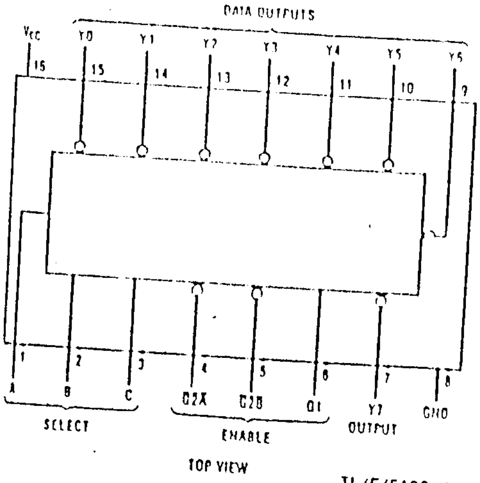
MM54HCT/MM74HCT devices are intended to interface between TTL and NMOS components and standard CMOS devices. These parts are also plug in replacements for LS-TTL devices and can be used to reduce power consumption in existing designs.

Features

- TTL Input Compatible
- Typical propagation delay: 20 ns
- Low quiescent current: 80 μA maximum (74HCT series)
- Low input current: 1 μA maximum
- Fanout of 10 LS-TTL loads

nection Diagram

Dual-In-Line Package



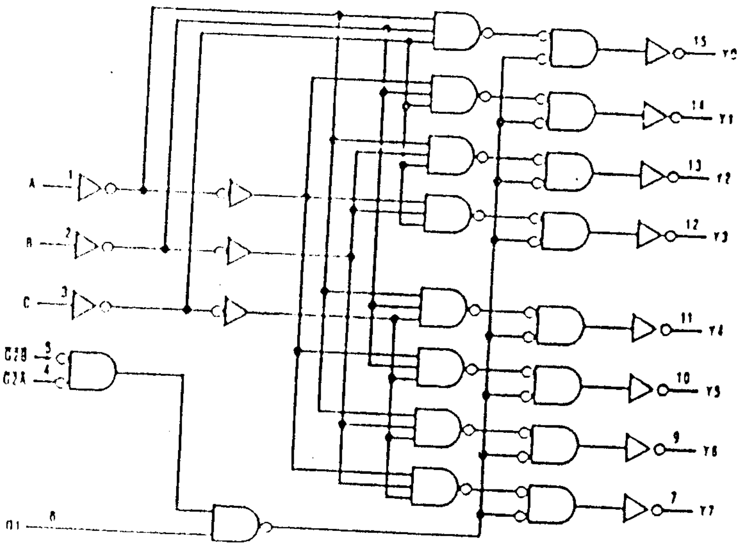
TOP VIEW

TL/F/5120-1

MM54HCT138/MM74HCT138

54HCT138 (J) 74HCT138 (J,N)

Logic Diagram



TL/F/5120-2

Table

Inputs					Outputs							
Enable		Select										
G1	$\overline{G2}$*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A + G2B

H = high level, L = low level, X = don't care

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC}+1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC}+0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage(V_{CC})	4.5	5.5	V
DC Input or Output Voltage (V_{IN},V_{OUT})	0	V_{CC}	V
Operating Temperature Range(T_A)			
MM74HCT	−40	+85	°C
MM54HCT	−55	+125	°C
Input Rise or Fall Times (t_r, t_f)		500	ns

DC Electrical Characteristics

V_{CC}=5V ±10% (unless otherwise specified)

Symbol	Parameter	Conditions	T_A=25°C		74HCT T_A=−40 to 85°C	54HCT T_A=−55 to 125°C	Units
			Typ	Guaranteed Limits			
V_{IH}	Minimum High Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum Low Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum High Level Output Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$=20 µA	V_{CC}	V_{CC}−0.1	V_{CC}−0.1	V_{CC}−0.1	V
		$\lvert I_{OUT}\rvert$=4.0 mA, V_{CC}=4.5V	4.2	3.98	3.84	3.7	V
		$\lvert I_{OUT}\rvert$=4.8 mA, V_{CC}=5.5V	5.7	4.98	4.84	4.7	V
V_{OL}	Maximum Low Level Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$=20 µA	0	0.1	0.1	0.1	V
		$\lvert I_{OUT}\rvert$=4.0 mA, V_{CC}=4.5V	0.2	0.26	0.33	0.4	V
		$\lvert I_{OUT}\rvert$=4.8 mA, V_{CC}=5.5V	0.2	0.26	0.33	0.4	V
I_{IN}	Maximum Input Current	V_{IN}=V_{CC} or GND		±0.1	±1.0	±1.0	µA
I_{CC}	Maximum Quiescent Supply Current	V_{IN}=V_{CC} or GND I_{OUT}=0 µA		8.0	80	160	µA
		V_{IN}=2.4V or 0.4V (Note 4)	100				µA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: This is measured per input pin. All other inputs are held at V_{CC} or ground.

AC Electrical Characteristics $T_A = 25°C$, $V_{CC} = 5.0V$, $t_r = t_f = 6$ ns $C_L = 15$ pF (unless otherwise specified)

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PHL}	Maximum Propagation Delay, A, B, or C to Output		20	35	ns
t_{PLH}	Maximum Propagation Delay, A, B, or C to Output		13	25	ns
t_{PHL}	Maximum Propagation Delay, G1 to Y Output		14	25	ns
t_{PLH}	Maximum Propagation Delay, G1 to Y Output		13	25	ns
t_{PHL}	Maximum Propagation Delay, $\overline{G2A}$ or $\overline{G2B}$ to Y Output		17	30	ns
t_{PLH}	Maximum Propagation Delay, $\overline{G2A}$ or $\overline{G2B}$ to Y Output		13	25	ns

AC Electrical Characteristics

$V_{CC} = 5V \pm 10\%$ $C_L = 50$ pF, $t_r = t_f = 6$ ns (unless otherwise specified)

Symbol	Parameter	Conditions	$T_A = 25°C$		74HCT $T_A = -40$ to 85°C	54HCT $T = -55$ to 125°C	Units
			Typ		Guaranteed Limits		
t_{PHL}	Maximum Propagation Delay A, B, or C to Output		24	40	50	60	ns
t_{PLH}	Maximum Propagation Delay A, B, or C to Output		18	30	38	45	ns
t_{PHL}	Maximum Propagation Delay G1 to Y Output		17	30	38	45	ns
t_{PLH}	Maximum Propagation Delay G1 to Y Output		20	30	38	45	ns
t_{PHL}	Maximum Propagation Delay $\overline{G2A}$ or $\overline{G2B}$ to Y Output		23	35	43	52	ns
t_{PLH}	Maximum Propagation Delay $\overline{G2A}$ or $\overline{G2B}$ to Y Output		18	30	38	45	ns
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time			15	19	22	ns
C_{IN}	Input Capacitance			5	10	10	pF
C_{PD}	Power Dissipation Capacitance	(Note 5)					pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74HCT AC Switching Waveforms and Test Circuits.

National Semiconductor

IM54HC574/MM74HC574
RI-STATE® Octal D-Type Flip-Flop

eneral Description

se high speed octal D-type flip-flops utilize micro-
OS™ Technology, 3.5 micron silicon gate P-well CMOS.
y possess the high noise immunity and low power con-
ption of standard CMOS integrated circuits, as well as
ability to drive 15 LS-TTL loads. Due to the large output
e capability and the TRI-STATE feature, these devices
deally suited for interfacing with bus lines in a bus orga-
d system.

se devices are positive edge triggered flip-flops. Data at
D inputs, meeting the set-up and hold time require-
ts, are transferred to the Q outputs on positive going
sitions of the CLOCK (CK) input. When a high logic level
plied to the OUTPUT CONTROL (OC) input, all outputs
o a high impedance state, regardless of what signals are
ent at the other inputs and the state of the storage
ents.

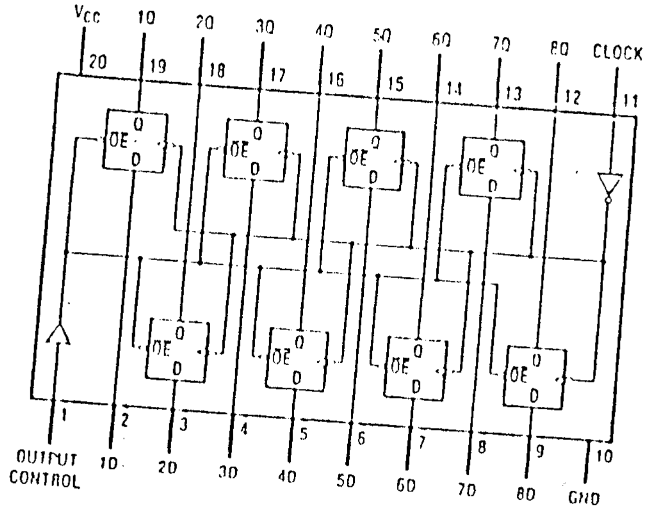
The 54HC/74HC logic family is speed, function, and pinout compatible with the standard 54LS/74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

Features

- Typical propagation delay: 15 ns
- Wide operating voltage range: 2V–6V
- Low input current: 1 µA maximum
- Low quiescent current: 80 µA maximum
- Compatible with bus-oriented systems
- Output drive capability: 15 LS-TTL loads

nnection Diagrams

Dual-In-Line Package



TOP VIEW

TL/F/5213-1

MM54HC574/MM74HC574

54HC574 (J) 74HC574 (J,N)

h Table

Output Control	Clock	Data	Output
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

H = High Level, L = Low Level
X = Don't Care
↑ = Transition from low-to-high
Z = High impedance state
Q_0 = The level of the output before steady state
nput conditions were established

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC}+1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC}+0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions 81

	Min	Max	Units
Supply Voltage(V_{CC})	2	6	V
DC Input or Output Voltage (V_{IN},V_{OUT})	0	V_{CC}	V
Operating Temperature Range(T_A)			
MM74HC	−40	+85	°C
MM54HC	−55	+125	°C
Input Rise or Fall Times (t_r, t_f) V_{CC}=2.0V		1000	ns
V_{CC}=4.5V		500	ns
V_{CC}=6.0V		400	ns

DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	V_{CC}	T_A=25°C		74HC T_A=−40 to 85°C	54HC T_A=−55 to 125°C	Units
				Typ	Guaranteed Limits			
V_{IH}	Minimum High Level Input Voltage		2.0V		1.5	1.5	1.5	V
			4.5V		3.15	3.15	3.15	V
			6.0V		4.2	4.2	4.2	V
V_{IL}	Maximum Low Level Input Voltage		2.0V		0.3	0.3	0.3	V
			4.5V		0.9	0.9	0.9	V
			6.0V		1.2	1.2	1.2	V
V_{OH}	Minimum High Level Output Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert\le20\ \mu A$	2.0V	2.0	1.9	1.9	1.9	V
			4.5V	4.5	4.4	4.4	4.4	V
			6.0V	6.0	5.9	5.9	5.9	V
		V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert\le4.0$ mA	4.5V	4.2	3.98	3.84	3.7	V
		$\lvert I_{OUT}\rvert\le5.2$ mA	6.0V	5.7	5.48	5.34	5.2	V
V_{OL}	Maximum Low Level Output Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert\le20\ \mu A$	2.0V	0	0.1	0.1	0.1	V
			4.5V	0	0.1	0.1	0.1	V
			6.0V	0	0.1	0.1	0.1	V
		V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert\le4.0$ mA	4.5V	0.2	0.26	0.33	0.4	V
		$\lvert I_{OUT}\rvert\le5.2$ mA	6.0V	0.2	0.26	0.33	0.4	V
I_{IN}	Maximum Input Current	V_{IN}=V_{CC} or GND	6.0V		±0.1	±1.0	±1.0	µA
I_{CC}	Maximum Quiescent Supply Current	V_{IN}=V_{CC} or GND I_{OUT}=0 µA	6.0V		8.0	80	160	µA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH}, and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC}=5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

AC Electrical Characteristics V_{CC} = 5V, T_A = 25°C, t_r = t_f = 6 ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
f_{MAX}	Maximum Operating Frequency		50	35	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock to Q	C_L = 45 pF	12	20	ns
t_{PZH}, t_{PZL}	Maximum Output Enable Time	R_L = 1 kΩ C_L = 45 pF	13	25	ns
t_{PHZ}, t_{PLZ}	Maximum Output Disable Time	R_L = 1 kΩ C_L = 5 pF	11	20	ns
t_S	Minimum Set-Up Time			20	ns
t_H	Minimum Hold Time			0	ns
t_W	Minimum Pulse Width			16	ns

AC Electrical Characteristics V_{CC} = 2.0 - 6.0V, C_L = 50 pF, t_r = t_f = 6 ns (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	T_A = 25°C		74HC T_A = −40 to 85°C	54HC T_A = −55 to 125°C	Units
				Typ	Guaranteed Limits			
f_{MAX}	Maximum Operating Frequency	C_L = 50 pF	2.0V		6	5	4	MHz
			4.5V		30	24	20	MHz
			6.0V		35	28	23	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock to $\overline{Q}$	C_L = 50 pF	2.0V	40	115	143	173	ns
		C_L = 150 pF	2.0V	51	155	194	233	ns
		C_L = 50 pF	4.5V	13	23	29	35	ns
		C_L = 150 pF	4.5V	19	31	47	47	ns
		C_L = 50 pF	6.0V	12	20	25	30	ns
		C_L = 150 pF	6.0V	18	27	34	41	ns
t_{PZH}, t_{PZL}	Maximum Output Enable Time	R_L = 1 kΩ C_L = 50 pF	2.0V	45	140	175	210	ns
		C_L = 150 pF	2.0V	59	180	225	270	ns
		C_L = 50 pF	4.5V	14	28	35	42	ns
		C_L = 150 pF	4.5V	20	36	45	54	ns
		C_L = 50 pF	6.0V	12	24	30	36	ns
		C_L = 150 pF	6.0V	18	31	39	47	ns
t_{PHZ}, t_{PLZ}	Maximum Output Disable Time	R_L = 1 kΩ C_L = 50 pF	2.0V	35	125	156	188	ns
			4.5V	12	25	31	38	ns
			6.0V	10	21	27	32	ns
t_S	Minimum Set-Up Time		2.0V		100	125	150	ns
			4.5V		20	25	30	ns
			6.0V		17	21	25	ns
t_H	Minimum Hold Time		2.0V		0	0	0	ns
			4.5V		0	0	0	ns
			6.0V		0	0	0	ns
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time	C_L = 50 pF	2.0V	25	60	75	90	ns
			4.5V	7	12	15	18	ns
			6.0V	6	10	13	15	ns
t_W	Minimum Pulse Width		2.0V	30	80	100	120	ns
			4.5V	9	16	20	24	ns
			6.0V	8	14	18	20	ns
t_r, t_f	Maximum Output Rise andFall Time		2.0V		1000	1000	1000	ns
			4.5V		500	500	500	ns
			6.0V		400	400	400	ns
C_{PD}	Power Dissipation Capacitance (Note 5)	OC = VCC		30				pF
		OC = GND		50				pF
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF
C_{OUT}	Maximum Output Capacitance			15	20	20	20	pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD}\,V_{CC}^2\,f + I_{CC}\,V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD}\,V_{CC}\,f + I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74HC AC Switching Waveforms and Test Circuits.

National Semiconductor

PRELIMINARY

MM54HCT245/MM74HCT245 Octal TRI-STATE® Transceiver

General Description

This TRI-STATE bi-directional buffer utilizes microCMOS™ technology, 3.0 micron silicon gate N-well CMOS, and is intended for two-way asynchronous communication between data buses. They have high drive current outputs which enable high speed operation even when driving large bus capacitances. These circuits possess the low power consumption of CMOS circuitry, yet have speeds comparable to low power Schottky TTL circuits.

All three devices are TTL input compatible and can drive up to 15 LS-TTL loads, and all inputs are protected from damage due to static discharge by diodes to Vcc and ground.

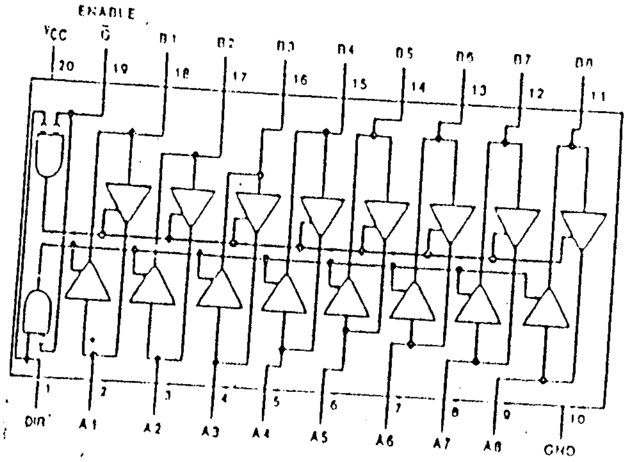
MM54HCT245/MM74HCT245 has one active low enable input (G), and a direction control (DIR). When the DIR input is high, data flows from the A inputs to the B outputs. When DIR is low, data flows from B to A.

MM54HCT/74HCT devices are intended to interface between TTL and NMOS components and standard CMOS devices. These parts are also plug in replacements for LS-TTL devices and can be used to reduce power consumption in existing designs.

Features

- TTL Input Compatible
- Octal TRI-STATE outputs for µp bus applications: 6 mA, typ.
- High speed: 12 ns typical propagation delay
- Low Power: 80 µA (74 Series)

Connection Diagram



TL/F/5165-1

MM54HCT245/MM74HCT245

54HCT245 (J) 74HCT245 (J,N)

Truth Table

Control Inputs		Operation
G̅	DIR	245
L	L	B data to A bus
L	H	A data to B bus
H	X	isolation

H = high level L = low level, X = irrelevant

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC}+1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC}+0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±35 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±70 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage(V_{CC})	4.5	5.5	V
DC Input or Output Voltage (V_{IN},V_{OUT})	0	V_{CC}	V
Operating Temperature Range(T_A)			
MM74HCT	−40	+85	°C
MM54HCT	−55	+125	°C
Input Rise or Fall Times (t_r, t_f)		500	ns

DC Electrical Characteristics

(V_{CC}=5V ±10%, unless otherwise specified.)

Symbol	Parameter	Conditions	T_A=25°C		74HCT T_A= −40 to 85°C	54HCT T_A= −55 to 125°C	Units
			Typ	Guaranteed Limits			
V_{IH}	Minimum High Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum Low Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum High Level Output Voltage	$V_{IN}=V_{IH}$ or V_{IL} $\lvert I_{OUT}\rvert$=20 μA $\lvert I_{OUT}\rvert$=6.0 mA, V_{CC}=4.5V $\lvert I_{OUT}\rvert$=7.2 mA, V_{CC}=5.5V	 V_{CC} 4.2 5.7	 V_{CC}−0.1 3.98 4.98	 V_{CC}−0.1 3.84 4.84	 V_{CC}−0.1 3.7 4.7	 V V V
V_{OL}	Maximum Low Level Voltage	$V_{IN}=V_{IH}$ or V_{IL} $\lvert I_{OUT}\rvert$=20 μA $\lvert I_{OUT}\rvert$=6.0 mA, V_{CC}=4.5V $\lvert I_{OUT}\rvert$=7.2 mA, V_{CC}=5.5V	 0 0.2 0.2	 0.1 0.26 0.26	 0.1 0.33 0.33	 0.1 0.4 0.4	 V V V
I_{IN}	Maximum Input Current	$V_{IN}=V_{CC}$ or GND		±0.1	±1.0	±1.0	μA
I_{OZ}	Maximum Tri-State Output Leakage Current	$V_{OUT}=V_{CC}$ or GND $\overline{G}=V_{IH}$		±0.5	±5.0	±10	μA
I_{CC}	Maximum Quiescent Supply Current	$V_{IN}=V_{CC}$ or GND I_{OUT}=0 μA		8	80	160	μA
		V_{IN}=2.4V or 0.4V (Note 4)	100				μA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: Measured per input. All other inputs at V_{CC} or ground.

AC Electrical Characteristics MM54HCT245/MM74HCT245

V_{CC} = 5.0V, $t_r = t_f$ = 6 ns, T_A = 25°C, (unless otherwise specified)

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PHL}, t_{PLH}	Maximum Output Propagation Delay	C_L = 45 pF	15	20	ns
t_{PZL}, t_{PZH}	Maximum Output Enable Time	C_L = 45 pF R_L = 1 kΩ	18	28	ns
t_{PLZ}, t_{PHZ}	Maximum Output Disable Time	C_L = 5 pF R_L = 1 kΩ	16	25	ns

AC Electrical Characteristics MM54HCT245/74HCT245

V_{CC} = 5.0V ±10%, $t_r = t_f$ = 6 ns (unless otherwise specified)

Symbol	Parameter	Conditions		T_A = 25°C		74HCT T_A = −40 to 85°C	54HCT T_A = −55 to 125°C	Units
				Typ	Guaranteed Limits			
t_{PHL}, t_{PLH}	Maximum Output Propagation Delay	C_L = 50 pF		14	23	29	34	ns
		C_L = 150 pF		17	30	38	45	ns
t_{PZH}, t_{PZL}	Maximum Output Enable Time	R_L = 1 kΩ	C_L = 50 pF	17	30	38	45	ns
			C_L = 150 pF	20	35	43	52	ns
t_{PHZ}, t_{PLZ}	Maximum Output Disable Time	R_L = 1 kΩ		18	30	38	45	ns
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time	C_L = 50 pF		8	12	15	18	ns
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF
C_{OUT}	Maximum Output Capacitance			15	20	20	20	pF
C_{PD}	Power Dissipation Capacitance	(Note 5) $\overline{G}$ = V_{CC} $\overline{G}$ = GND						pF

Note 5: C_{PD} determines the no load power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74HCT AC Switching Waveforms and Test Circuits.

74160, 74161, 74163, LS160A, LS161A, LS162A, LS163A Counters

'160, '162 BCD Decade Counter
'161, '163 4-Bit Binary Counter
Product Specification

ic Products

ATURES

- ynchronous counting and ading
- wo Count Enable Inputs for n-t cascading
- ositive edge-triggered clock
- synchronous reset ('160, '161)
- ynchronous reset ('162, '163)
- ysteresis on Clock Input (LS nly)

SCRIPTION

chronous presettable decade 160, 74LS160A, 74LS162A) and 4-bit 161, 74LS161A, 74163, 74LS163A) nters feature an internal carry look-ad and can be used for high-speed nting. Synchronous operation is pro-d by having all flip-flops clocked ltaneously on the positive-going of the clock. The Clock input is ered.

outputs of the counters may be set to HIGH or LOW level. A LOW l at the Parallel Enable (PE) input bles the counting action and causes data at the $D_0 - D_3$ inputs to be ed into the counter on the positive-g edge of the clock (providing that set-up and hold requirements for PE met). Preset takes place regardless he levels at Count Enable (CEP,) inputs.

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (TOTAL)
74160 - 74163	32MHz	61mA
74LS160A - 74LS163A	32MHz	19mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGE $V_{CC} = 5V \pm 5\%$; $T_A = 0°C$ to $+70°C$
Plastic DIP	N74160N, N74LS160AN, N74161N, N74LS161AN N74LS162AN, N74163N, N74LS163AN
Plastic SO	N74LS161AD, N74S163AD

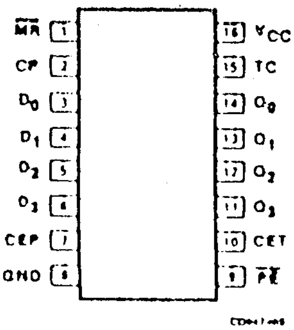
NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

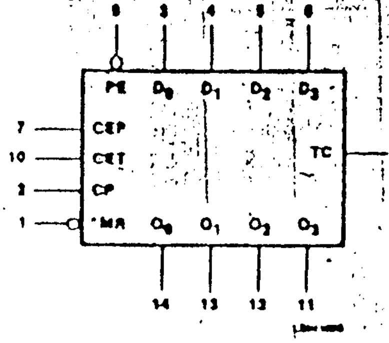
PINS	DESCRIPTION	74	74LS
CP, CET	Inputs	2ul	2LSul
D, CEP	Inputs	1ul	1LSul
PE	Input	1ul	2LSul
All	Outputs	10ul	10LSul
MR	Input ('160, '161)	1ul	1LSul
MR	Input ('162, '163)	1ul	2LSul

NOTE:
Where a 74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}, and a 74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

CONFIGURATION



LOGIC SYMBOL

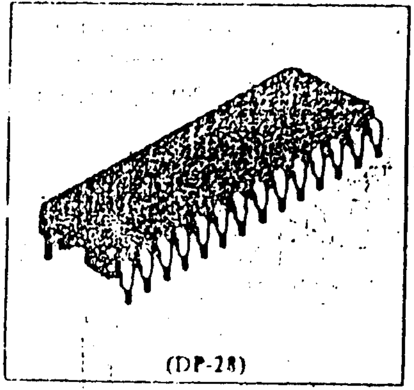


V_{CC} = Pin 16

HM6264P-10, HM6264P-12, HM6264P-15

8192-word × 8-bit High Speed Static CMOS RAM

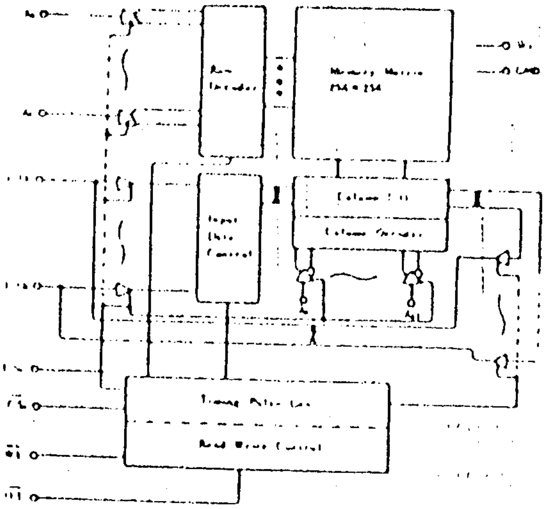
FEATURES



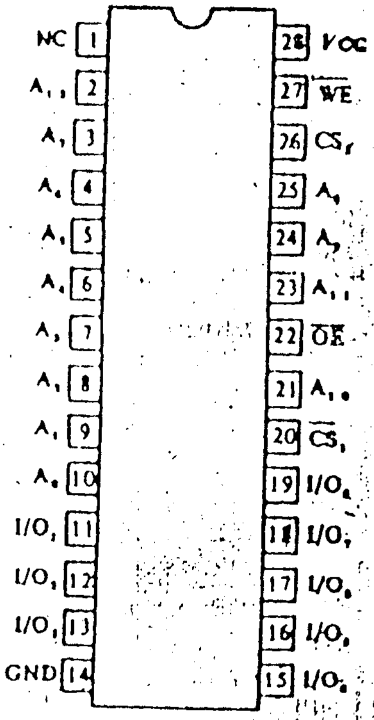
(DP-28)

- Fast access Time 100ns/120ns/150ns (max.)
- Low Power Standby Standby: 0.1mW (typ.)
  Low Power Operation Operating 200mW (typ.)
- Single +5V Supply
- Completely Static Memory...... No clock or Timing Strobe Required
- Equal Access and Cycle Time
- Common Data Input and Output, Three State Output
- Directly TTL Compatible: All Input and Output
- Standard 28pin Package Configuration
- Pin Out Compatible with 64K EPROM HN482764

BLOCK DIAGRAM



PIN ARRANGEMENT



(Top View)

ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Rating	Unit
Terminal Voltage *	V_T	−0.5 ** to +7.0	V
Power Dissipation	P_T	1.0	W
Operating Temperature	T_{opr}	0 to +70	°C
Storage Temperature	T_{stg}	−55 to +125	°C
Storage Temperature (Under Bias)	T_{bias}	−10 to +85	°C

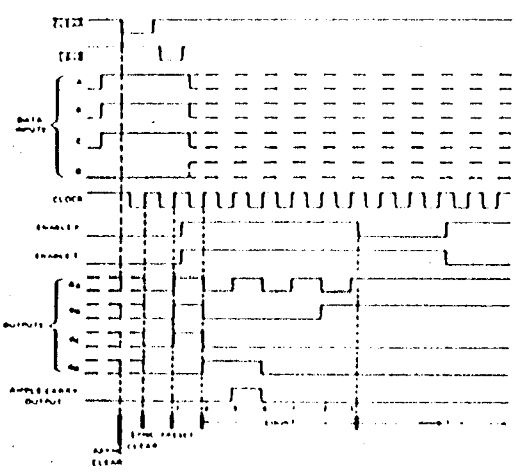
* With respect to GND. ** Pulse width 50ns: −3.0V

TRUTH TABLE

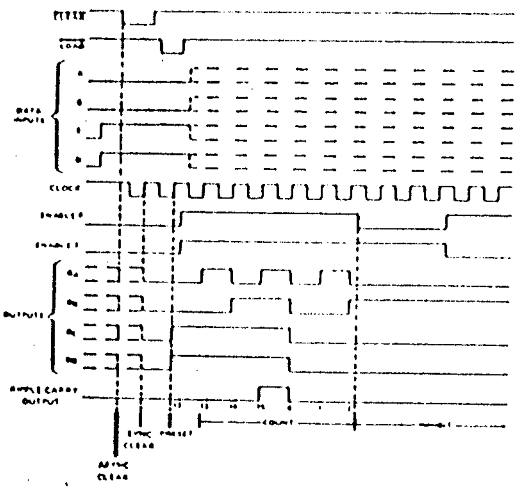
WE	CS1	CS2	OE	Mode	I/O Pin	V_{CC} Current	Note
X	H	X	X	Not Selected (Power Down)	High Z	I_{SB}, I_{SB1}	
X	X	L	X		High Z	I_{SB}, I_{SB2}	
H	L	H	H	Output Disabled	High Z	I_{CC}, I_{CC1}	
H	L	H	L	Read	Dout	I_{CC}, I_{CC1}	
L	L	H	H	Write	Din	I_{CC}, I_{CC1}	Write Cycle (1)
L	L	H	L		Din	I_{CC}, I_{CC1}	Write Cycle (2)

X: Don't care.

160,162



161,163



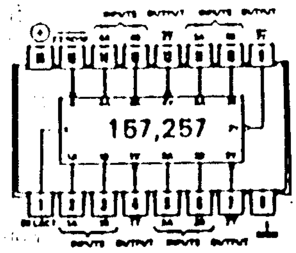
These counters are fully programmable; that is, the outputs may be preset to either level. As presetting is synchronous, setting up a low level at the $\overline{\text{load}}$ input disables the counter and causes the outputs to agree with the setup data after the next clock pulse regardless of the levels of the enable inputs. Low-to-high transitions at the $\overline{\text{load}}$ input of the HC(T)-types should be avoided when the clock is low if the enable inputs are high at or before the transition. This restriction is not applicable to the'LS 160A thru 'LS 163A. The clear function of the '160, '161 is asynchronous and a low level at the clear input sets all four of the flip-flop outputs low regardless of the levels of clock, $\overline{\text{load}}$, or enable inputs The clear function for the '162, '163 is synchronous and a low level at the $\overline{\text{clear}}$ input sets all four of the flip-flop outputs low after the next clock pulse, regardless of the levels of the enabe inputs. This synchronous clear allows the count lengths to be modified as decoding the maximum count desired can be accomplished with one external NAND gate. The gate output is connected to the clear input to synchronously clear the counter to 0000 (LLLL). Low-to-high transitions at the $\overline{\text{clear}}$ input of the '162 and '163 should be avoided when the clock is low if the enable and $\overline{\text{load}}$ inputs are high at or before the transition. The carry look-ahead circuitry provides for cascading counters for n-bit synchronous applications without additional gating. Instrumental in accomplishing this function are two count-enable inputs and a ripple carry output. Both count-enable inputs (P and T) must be high to count, and input T is fed forward to enable the ripple carry output. The ripple carry output thus enabled will produce a high-level output pulse with a duration approximately equal to the high-level portion of the Q_A output. This high-level overflow ripple carry pulse can be used to enable successive cascaded stages. High-to-low-level transitions at the enable P or T inputs of the HC(T)-types should occur only when the clock input is high. Transitions at the enable P or T inputs of the 'LS 160A thru 'LS 163A are allowed regardless of the level of the clock input.

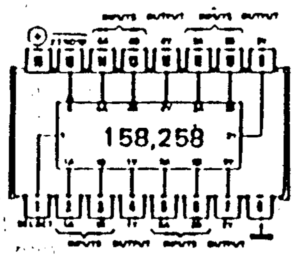
'LS 160A thru 'LS163A feature a fully independent clock circuit. Changes at control inputs (enable P or T, or clear) that will modify the operating mode have no effect until clocking occurs. The function of the counter (whether enabled, disabled, loading, or counting) will be dictated solely by the conditions meeting the stable setup and hold times.

Description

These monolithic data selectors/multiplexers contain inverters and drivers to supply full on-chip data selection to the four output gates. A separate strobe input is provided. A 4-bit word is selected from one of two sources and is routed to the four outputs.

157 Quad 2- to 1-line data selectors/multiplexers
non-inverted data outputs
158 Inverted data outputs
257 Non inverted 3 state outputs
Fan-out = 3 x standard fan-out
258 Inverted 3 state outputs
Fan-out = 3 x standard fan-out





FUNCTION TABLE

INPUTS				Y OUTP.	$\overline{Y}$ OUTP.
STROBE	SELECT	A	B	HC(T)157/ 257 LS157/ 257	HC(T)158/ 258 LS158/ 258
H	X	X	X	L*	H*
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

* = high impedance for the LS 257A and LS 258A

	supply current (mA)	typ. average propagation delay time (ns)
HC(T)157	*	10(12)
LS 157	9,8	9
HC(T)158	*	11(13)
LS 158	4,8	7
HC(T)257	*	12(13)
LS 257A	12	12
HC(T)258	*	7(11)
LS 258A	12	12

LS 157, LS 158: The S and $\overline{G}$ Inputs have an Input-load of 2
LS 257A, LS 258A: The S Input has an Input-load of 2

159: see 154

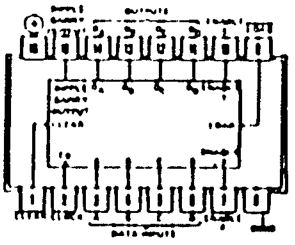
Description

These synchronous, presettable counters feature an internal cary look-ahead for application in high-speed counting designs. The '160,, '162, are decade counters and the '161, '163, are 4-bit binary counters. Synchronous operation is provided by having all flip-flops clocked simultaneaously so that the outputs change coincident with each ohter when so instructed by the count-enable inputs and internal gating. This mode of operation eliminates the output counting spikes that are normally associated with asynchronous (ripple clock) counters. A buffered clock input triggers the four flip-flops on the rising (positive-going) edge of the clock input waveform.

160
161 Synchronous counters with direct clear
162
163 Fully synchronous counters

40160 . . . 40163

V_{DD}	5	10	15	V
Maximum clock frequency (RCA)	3	8,5	12	MHz



	supply curr. (mA)	f_{clock} max. (MHz)	Fan in			
			$\overline{\text{clear}}$	$\overline{\text{load}}$	clock	en T
HC(T)160	*	30				
LS 160A	18	25	1	2	2	2
HC(T)161	*	30				
LS 161A	18	25	1	2	2	2
HC(T)162	*	30				
LS 162A	18	25	2	2	2	2
HC(T)163	*	30				
LS 163A	18	25	2	2	2	2

National Semiconductor

MM54HC14/MM74HC14 Hex Inverting Schmitt Trigger

General Description

The MM54HC14/MM74HC14 utilizes microCMOS™ Technology, 3.5 micron silicon gate P-well CMOS, to achieve the low power dissipation and high noise immunity of standard CMOS, as well as the capability to drive 10 LS-TTL loads.

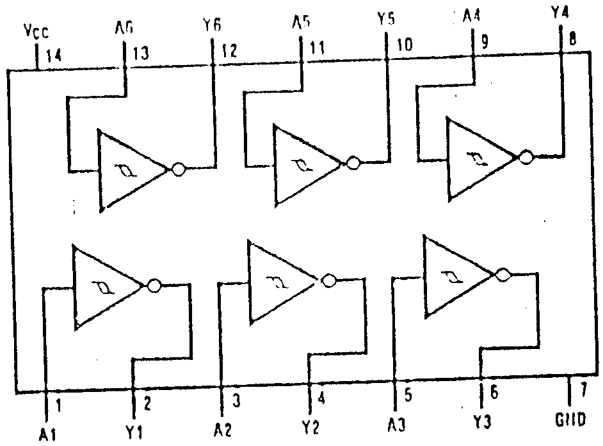
The 54HC/74HC logic family is functionally and pinout compatible with the standard 54LS/74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

Features

- Typical propagation delay: 13 ns
- Wide power supply range: 2–6V
- Low quiescent current: 20 μA maximum (74HC seri
- Low input current: 1 μA maximum
- Fanout of 10 LS-TTL loads
- Typical hysteresis voltage: 0.9V at V_{CC} = 4.5V

Connection Diagram

Dual-In-Line Package



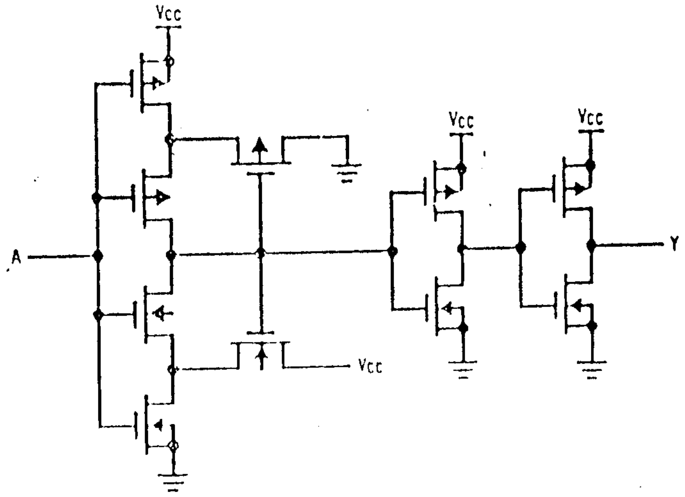
TOP VIEW

TL/F/5105-1

MM54HC14/MM74HC14

54HC14(J) 74HC14(J,N)

Schematic Diagram



TL/F/5105-2

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max
Supply Voltage(V_{CC})	2	6
DC Input or Output Voltage (V_{IN},V_{OUT})	0	V_{CC}
Operating Temperature Range(T_A)		
MM74HC	−40	+85
MM54HC	−55	+125

DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	V_{CC}	T_A = 25°C		74HC T_A = −40 to 85°C	54HC T_A = −55 to 125°C
				Typ	Guaranteed Limits		
V_T+	Maximum Positive Going Threshold Voltage		2.0V	1.2	1.5	1.5	1.5
			4.5V	2.7	3.15	3.15	3.15
			6.0V	3.2	4.2	4.2	4.2
V_T−	Minimum Negative Going Threshold Voltage		2.0V	0.7	0.3	0.3	0.3
			4.5V	1.8	0.9	0.9	0.9
			6.0V	2.2	1.2	1.2	1.2
V_H	Hysterisis Voltage	Min	2.0V	0.5	0.2	0.2	0.2
			4.5V	0.9	0.4	0.4	0.4
			6.0V	1.0	0.6	0.6	0.6
		Max	2.0V	0.5	1.2	1.2	1.2
			4.5V	0.9	2.25	2.25	2.25
			6.0V	1.0	3.0	3.0	3.0
V_{OH}	Minimum High Level Output Voltage	V_{IN} = V_{IL} $\|I_{OUT}\| \le 20\ \mu A$	2.0V	2.0	1.9	1.9	1.9
			4.5V	4.5	4.4	4.4	4.4
			6.0V	6.0	5.9	5.9	5.9
		V_{IN} = V_{IL} $\|I_{OUT}\| \le 4.0$ mA	4.5V	4.2	3.98	3.84	3.7
		$\|I_{OUT}\| \le 5.2$ mA	6.0V	5.7	5.48	5.34	5.2
V_{OL}	Maximum Low Level Output Voltage	V_{IN} = V_{IH} $\|I_{OUT}\| \le 20\ \mu A$	2.0V	0	0.1	0.1	0.1
			4.5V	0	0.1	0.1	0.1
			6.0V	0	0.1	0.1	0.1
		V_{IN} = V_{IH} $\|I_{OUT}\| \le 4.0$ mA	4.5V	0.2	0.26	0.33	0.4
		$\|I_{OUT}\| \le 5.2$ mA	6.0V	0.2	0.26	0.33	0.4
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND	6.0V		±0.1	±1.0	±1.0
I_{CC}	Maximum Quiescent Supply Current	V_{IN} = V_{CC} or GND I_{OUT} = 0 μA	6.0V		2.0	20	40

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH}, and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC} = 5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

AC Electrical Characteristics $V_{CC}=5V$, $T_A=25°C$, $C_L=15$ pF, $t_r=t_f=6$ ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PHL}, t_{PLH}	Maximum Propagation Delay		12	22	ns

AC Electrical Characteristics $V_{CC}=2.0V$ to $6.0V$, $C_L=50$ pF, $t_r=t_f=6$ ns (unless otherwise specified)

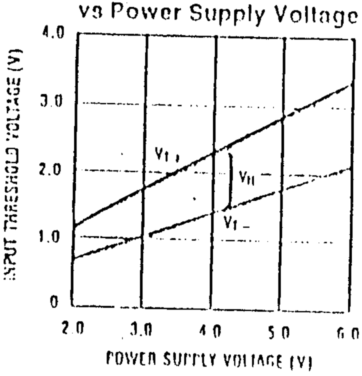
Symbol	Parameter	Conditions	V_{CC}	$T_A=25°C$		74HC $T_A=-40$ to 85°C	54HC $T_A=-55$ to 125°C	Units
				Typ	Guaranteed Limits			
t_{PHL}, t_{PLH}	Maximum Propagation Delay		2.0V 4.5V 6.0V	60 13 11	125 25 21	156 31 26	188 38 32	ns ns ns
t_{TLH}, t_{THL}	Maximum Output Rise and Fall Time		2.0V 4.5V 6.0V	30 8 7	75 15 13	95 19 16	110 22 19	ns ns ns
C_{PD}	Power Dissipation Capacitance (Note 5)	(per gate)		27				pF
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D=C_{PD}\ V_{CC}{}^2\ f+I_{CC}\ V_{CC}$, and the no load dynamic current consumption, $I_S=C_{PD}\ V_{CC}\ f+I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74 HC AC Switching Waveforms and Test Circuits.

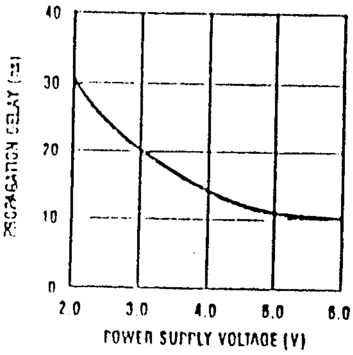
Typical Performance Characteristics

Input Threshold, V_{T+}, V_{T-}, vs Power Supply Voltage



TL/F/5105-3

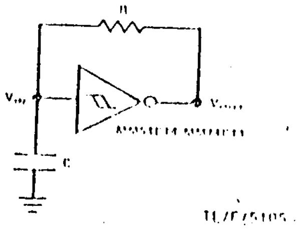
Propagation Delay vs Power Supply



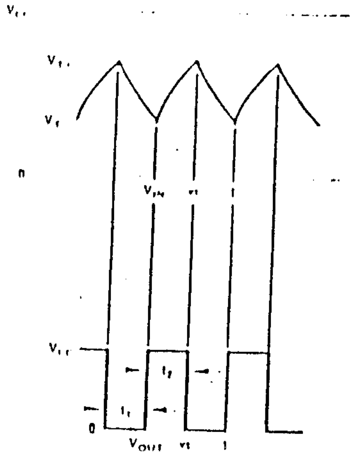
TL/F/5105-4

Typical Applications

Low Power Oscillator



TL/F/5105-5



TL/F/5105-6

$$t_1=RC\ln\frac{V_{T+}}{V_{T-}}$$

$$t_2=RC\ln\frac{V_{CC}-V_{T-}}{V_{CC}-V_{T+}}$$

$$f\approx\frac{1}{RC\ln\frac{V_{T+}(V_{CC}-V_{T-})}{V_{T-}(V_{CC}-V_{T+})}}$$

Note: The equations assume $t_1+t_2\gg t_{pd0}+t_{pd1}$

National Semiconductor

93 microCMOS

ADC0820 8-Bit High Speed μP Compatible A/D Converter with Track/Hold Function

General Description

By using a half-flash conversion technique, the 8-bit ADC0820 CMOS A/D offers a 1.5 μs conversion time and dissipates only 75 mW of power. The half-flash technique consists of 32 comparators, a most significant 4-bit ADC and a least significant 4-bit ADC.

The input to the ADC0820 is tracked and held by the input sampling circuitry eliminating the need for an external sample-and-hold for signals moving at less than 100 mV/μs.

For ease of interface to microprocessors, the ADC0820 has been designed to appear as a memory location or I/O port without the need for external interfacing logic.

Key Specifications

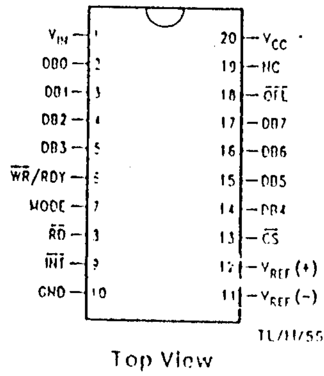
■ Resolution	8 Bits
■ Conversion Time	2.5 μs Max (RD Mode) 1.5 μs Max (WR-RD Mode)
■ Input signals with slew rate of 100 mV/μs converted without external sample-and-hold to 8 bits	
■ Low Power	75 mW Max
■ Total Unadjusted Error	± ½ LSB and ± 1 LSB

Features

- Built-in track-and-hold function
- No missing codes
- No external clocking
- Single supply—5 V_{CC}
- Easy interface to all microprocessors, or operates stand-alone
- Latched TRI-STATE® output
- Logic inputs and outputs meet both MOS and T²L voltage level specifications
- Operates ratiometrically or with any reference value equal to or less than V_{CC}
- 0V to 5V analog input voltage range with single 5V supply
- No zero or full-scale adjust required
- Overflow output available for cascading
- 0.3" standard width 20-pin DIP
- 20-pin molded chip carrier package
- 20-pin small outline package

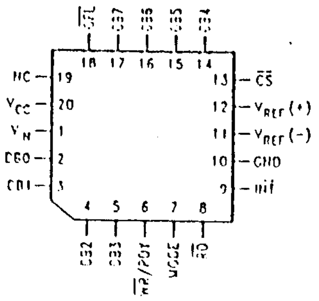
Connection and Functional Diagrams

Dual-In-Line and Small Outline Packages



Top View

Molded Chip Carrier Package



TL/H/5501-33

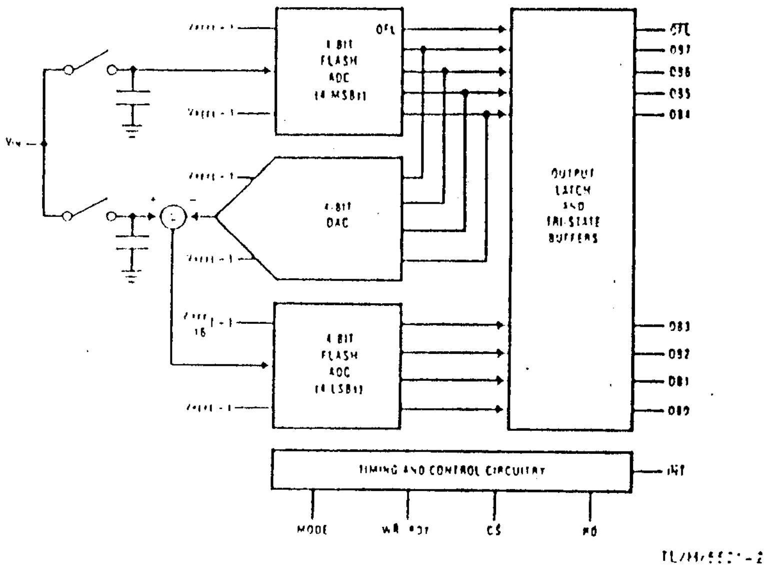


FIGURE 1

See Ordering Information

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	10V
Logic Control Inputs	−0.2V to V_{CC} + 0.2V
Voltage at Other Inputs and Output	−0.2V to V_{CC} + 0.2V
Storage Temperature Range	−65°C to + 150°C
Package Dissipation at T_A = 25°C	875 mW
Input Current at Any Pin (Note 5)	1 mA
Package Input Current (Note 5)	4 mA
ESD Susceptability (Note 9)	1200V
Lead Temp. (Soldering, 10 sec.)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Surface Mount Package	
Vapor Phase (60 sec.)	215°C
Infrared (15 sec.)	220°C

Operating Ratings (Notes 1 & 2)

Temperature Range	$T_{MIN} \leq T_A \leq T_{MAX}$
ADC0820BD, ADC0820CJ	$-55°C \leq T_A \leq +125°C$
ADC0820BCD, ADC0820CCJ	$-40°C \leq T_A \leq +85°C$
ADC0820BCN, ADC0820CCN	$0°C \leq T_A \leq 70°C$
ADC0820BCV, ADC0820CCV	$0°C \leq T_A \leq 70°C$
ADC0820BCWM, ADC0820CCWM	$0°C \leq T_A \leq 70°C$
V_{CC} Range	4.5V to 8V

Converter Characteristics

The following specifications apply for RD mode (pin 7 = 0), V_{CC} = 5V, $V_{REF}(+)$ = 5V, and $V_{REF}(-)$ = GND unless otherwise specified. **Boldface limits apply from T_{MIN} to T_{MAX};** all other limits $T_A = T_J = 25°C$.

Parameter	Conditions	ADC0820BD, ADC0820CJ ADC0820BCD, ADC0820CCJ			ADC0820BCN, ADC0820CCN ADC0820BCV, ADC0820CCV ADC0820BCWM, ADC0820CCWM			Limit Units
		Typ (Note 6)	Tested Limit (Note 7)	Design Limit (Note 8)	Typ (Note 6)	Tested Limit (Note 7)	Design Limit (Note 8)	
Resolution			8			8	8	Bits
Total Unadjusted Error (Note 3)	ADC0820BD, BCD ADC0820BCN ADC0820CD, CCD ADC0820CCN		±½ ±1 			 ±½ ±1	 ±½ ±1	LSB LSB LSB LSB
Minimum Reference Resistance		2.3	1.00		2.3	1.2		kΩ
Maximum Reference Resistance		2.3	6		2.3	5.3	6	kΩ
Maximum $V_{REF}(+)$ Input Voltage			V_{CC}			V_{CC}	V_{CC}	V
Minimum $V_{REF}(-)$ Input Voltage			GND			GND	GND	V
Minimum $V_{REF}(+)$ Input Voltage			$V_{REF}(-)$			$V_{REF}(-)$	$V_{REF}(-)$	V
Maximum $V_{REF}(-)$ Input Voltage			$V_{REF}(+)$			$V_{REF}(+)$	$V_{REF}(+)$	V
Maximum V_{IN} Input Voltage			V_{CC} + 0.1			V_{CC} + 0.1	V_{CC} + 0.1	V
Minimum V_{IN} Input Voltage			GND − 0.1			GND − 0.1	GND − 0.1	V
Maximum Analog Input Leakage Current	$\overline{CS}$ = V_{CC} V_{IN} = V_{CC} V_{IN} = GND		3 −3			0.3 −0.3	3 −3	μA μA
Power Supply Sensitivity	V_{CC} = 5V ± 5%	±1/16	±¼		±1/16	±¼	±¼	LSB

DC Electrical Characteristics

The following specifications apply for V_{CC} = 5V, unless otherwise specified. **Boldface limits apply from T_{MIN} to T_{MAX};** all other limits $T_A = T_J$ = 25°C.

Parameter	Conditions		ADC0820BD, ADC0820CJ ADC0820BCD, ADC0820CCJ			ADC0820BCN, ADC0820CCN ADC0820BCV, ADC0820CCV ADC0820BCWM, ADC0820CCWM			Limit Units
			Typ (Note 6)	Tested Limit (Note 7)	Design Limit (Note 8)	Typ (Note 6)	Tested Limit (Note 7)	Design Limit (Note 8)	
$V_{IN(1)}$, Logical "1" Input Voltage	V_{CC} = 5.25V	$\overline{CS}$, $\overline{WR}$, $\overline{RD}$		2.0			2.0	**2.0**	V
		Mode		3.5			3.5	**3.5**	V
$V_{IN(0)}$, Logical "0" Input Voltage	V_{CC} = 4.75V	$\overline{CS}$, $\overline{WR}$, $\overline{RD}$		0.8			0.8	**0.8**	V
		Mode		1.5			1.5	**1.5**	V
$I_{IN(1)}$, Logical "1" Input Current	$V_{IN(1)}$ = 5V; $\overline{CS}$, $\overline{RD}$ $V_{IN(1)}$ = 5V; $\overline{WR}$ $V_{IN(1)}$ = 5V; Mode		0.005 0.1 50	1 3 200		0.005 0.1 50	 0.3 170	**1** **3** **200**	μA μA μA
$I_{IN(0)}$, Logical "0" Input Current	$V_{IN(0)}$ = 0V; $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, Mode		−0.005	−1		−0.005		**−1**	μA
$V_{OUT(1)}$, Logical "1" Output Voltage	V_{CC} = 4.75V, I_{OUT} = −360 μA; DB0–DB7, $\overline{OFL}$, $\overline{INT}$ V_{CC} = 4.75V, I_{OUT} = −10 μA; DB0–DB7, $\overline{OFL}$, $\overline{INT}$			2.4 4.5			2.8 4.6	**2.4** **4.5**	V V
$V_{OUT(0)}$, Logical "0" Output Voltage	V_{CC} = 4.75V, I_{OUT} = 1.6 mA; DB0–DB7, $\overline{OFL}$, $\overline{INT}$, $\overline{RDY}$			0.4			0.34	**0.4**	V
I_{OUT}, TRI-STATE Output Current	V_{OUT} = 5V; DB0–DB7, RDY V_{OUT} = 0V; DB0–DB7, RDY		0.1 −0.1	3 −3		0.1 −0.1	0.3 −0.3	**3** **−3**	μA μA
I_{SOURCE}, Output Source Current	V_{OUT} = 0V; DB0–DB7, $\overline{OFL}$ $\overline{INT}$		−12 −9	−6 −4.0		−12 −9	−7.2 −5.3	**−6** **−4.0**	mA mA
I_{SINK}, Output Sink Current	V_{OUT} = 5V; DB0–DB7, $\overline{OFL}$, $\overline{INT}$, $\overline{RDY}$		14	7		14	8.4	**7**	mA
I_{CC}, Supply Current	$\overline{CS}$ = $\overline{WR}$ = $\overline{RD}$ = 0		7.5	15		7.5	13	**15**	mA

AC Electrical Characteristics

The following specifications apply for V_{CC} = 5V, $t_r = t_f$ = 20 ns, $V_{REF}(+)$ = 5V, $V_{REF}(-)$ = 0V and T_A = 25°C unless otherwise specified.

Parameter		Conditions	Typ (Note 6)	Tested Limit (Note 7)	Design Limit (Note 8)	Units
t_{CRD}, Conversion Time for RD Mode		Pin 7 = 0, *(Figure 2)*	1.6		2.5	μs
t_{ACC0}, Access Time (Delay from Falling Edge of $\overline{RD}$ to Output Valid)		Pin 7 = 0, *(Figure 2)*	t_{CRD} + 20		t_{CRD} + 50	ns
t_{CWR-RD}, Conversion Time for WR-RD Mode		Pin 7 = V_{CC}; t_{WR} = 600 ns, t_{RD} = 600 ns; *(Figures 3a and 3b)*			1.52	μs
t_{WR}, Write Time	Min	Pin 7 = V_{CC}; *(Figures 3a and 3b)*		600		ns
	Max	(Note 4) See Graph	50			μs
t_{RD}, Read Time	Min	Pin 7 = V_{CC}; *(Figures 3a and 3b)* (Note 4) See Graph		600		ns
t_{ACC1}, Access Time (Delay from Falling Edge of $\overline{RD}$ to Output Valid)		Pin 7 = V_{CC}, t_{RD} < t_I; *(Figure 3a)* C_L = 15 pF	190		280	ns
		C_L = 100 pF	210		320	ns
t_{ACC2}, Access Time (Delay from Falling Edge of $\overline{RD}$ to Output Valid)		Pin 7 = V_{CC}; t_{RD} > t_I; *(Figure 3b)* C_L = 15 pF	70		120	ns
		C_L = 100 pF	90		150	ns

AC Electrical Characteristics (Continued) The following specifications apply for $V_{CC} = 5V$, $t_r = t_f = 20$ ns, $V_{REF}(+) = 5V$, $V_{REF}(-) = 0V$ and $T_A = 25°C$ unless otherwise specified.

Parameter	Conditions	Typ (Note 6)	Tested Limit (Note 7)	Design Limit (Note 8)	Units
t_I, Internal Comparison Time	Pin 7 = V_{CC}; *(Figures 3b and 4)* $C_L = 50$ pF	800		1300	ns
t_{1H}, t_{0H}, TRI-STATE Control (Delay from Rising Edge of $\overline{RD}$ to Hi-Z State)	$R_L = 1k$, $C_L = 10$ pF	100		200	ns
t_{INTL}, Delay from Rising Edge of $\overline{WR}$ to Falling Edge of $\overline{INT}$	Pin 7 = V_{CC}, $C_L = 50$ pF $t_{RD} > t_I$; *(Figure 3b)* $t_{RD} < t_I$; *(Figure 3a)*	 $t_{RD} + 200$		 t_I $t_{RD} + 290$	 ns ns
t_{INTH}, Delay from Rising Edge of $\overline{RD}$ to Rising Edge of $\overline{INT}$	*(Figures 2, 3a and 3b)* $C_L = 50$ pF	125		225	ns
t_{INTHWR}, Delay from Rising Edge of $\overline{WR}$ to Rising Edge of $\overline{INT}$	*(Figure 4)*, $C_L = 50$ pF	175		270	ns
t_{RDY}, Delay from $\overline{CS}$ to RDY	*(Figure 2)*, $C_L = 50$ pF, Pin 7 = 0	50		100	ns
t_{ID}, Delay from $\overline{INT}$ to Output Valid	*(Figure 4)*	20		50	ns
t_{RI}, Delay from $\overline{RD}$ to $\overline{INT}$	Pin 7 = V_{CC}, $t_{RD} < t_I$ *(Figure 3a)*	200		290	ns
t_P, Delay from End of Conversion to Next Conversion	*(Figures 2, 3a, 3b and 4)* (Note 4) See Graph			500	ns
Slew Rate, Tracking		0.1			V/µs
C_{VIN}, Analog Input Capacitance		45			pF
C_{OUT}, Logic Output Capacitance		5			pF
C_{IN}, Logic Input Capacitance		5			pF

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

Note 2: All voltages are measured with respect to the GND pin, unless otherwise specified.

Note 3: Total unadjusted error includes offset, full-scale, and linearity errors.

Note 4: Accuracy may degrade if t_{WR} or t_{RD} is shorter than the minimum value specified. See Accuracy vs t_{WR} and Accuracy vs t_{RD} graphs.

Note 5: When the input voltage (V_{IN}) at any pin exceeds the power supply rails ($V_{IN} < V^-$ or $V_{IN} > V^+$) the absolute value of current at that pin should be limited to 1 mA or less. The 4 mA package input current limits the number of pins that can exceed the power supply boundaries with a 1 mA current limit to four.

Note 6: Typicals are at 25°C and represent most likely parametric norm.

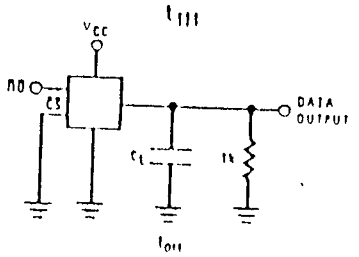
Note 7: Tested limits are guaranteed to National's AOQL (Average Outgoing Quality Level).

Note 8: Design limits are guaranteed but not 100% tested. These limits are not used to calculate outgoing quality levels.

Note 9: Human body model, 100 pF discharaged through a 1.5 kΩ resistor.

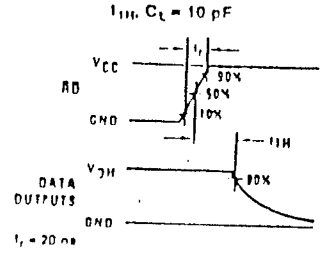
TRI-STATE Test Circuits and Waveforms

t_{1H}



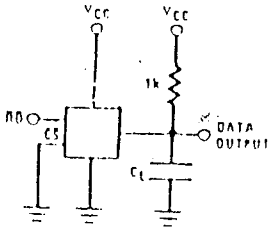
t_{0H}

TL/H/5501-3

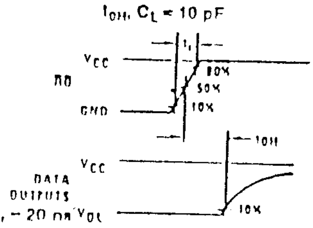


TL/H/5501-4

t_{0H}



TL/H/5501-5



TL/H/5501-6

National Semiconductor

MM54HC390/MM74HC390 Dual 4-Bit Decade Counter
MM54HC393/MM74HC393 Dual 4-Bit Binary Counter

eneral Description

hese counter circuits contain Independent ripple carry unters and utilize microCMOS™ Technology, 3.5 micron licon gate P-well CMOS. The MM54HC390/MM74HC390 corporate dual decade counters, each composed of a di- de-by-two and a divide-by-five counter. The divide-by-two d divide-by-five counters can be cascaded to form dual cade, dual bi-quinary, or various combinations up to a gle divide-by-100 counter. The MM54HC393/M74HC393 ntain two 4-bit ripple carry binary counters, which can be scaded to create a single divide-by-256 counter.

ch of the two 4-bit counters is incremented on the high to w transition (negative edge) of the clock input, and each s an independent clear input. When clear is set high all r bits of each counter are set to a low level. This enables unt truncation and allows the implementation of divide-by- counter configurations.

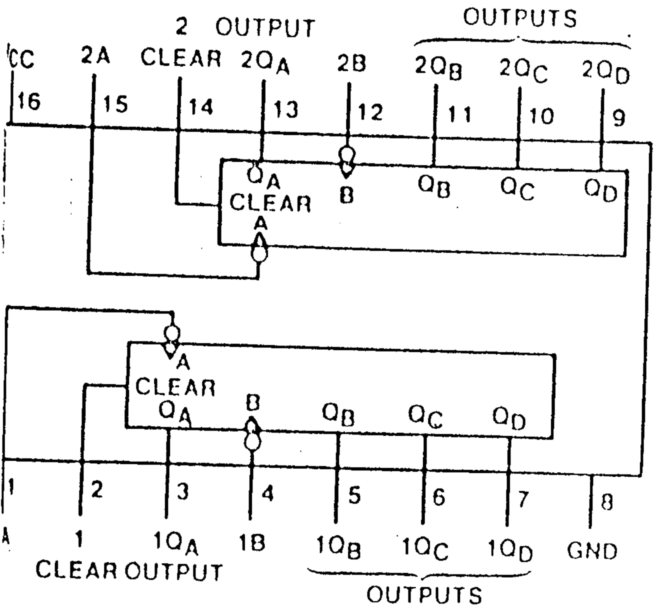
ch of the counters outputs can drive 10 low power hottky TTL equivalent loads. These counters are functionaly as well as pin equivalent to the 54LS390/74LS390 and the 54LS393/74LS393, respectively. All inputs are protected from damage due to static discharge by diodes to V_C and ground.

Features

- Typical operating frequency: 50 MHz
- Typical propagation delay: 13 ns (Ck to Q_A)
- Wide operating supply voltage range: 2–6V
- Low input current: <1 μA
- Low quiescent supply current: 80 μA maximum (74HC series)
- Fanout of 10 LS-TTL loads

onnection Diagrams

Dual-In-Line Package

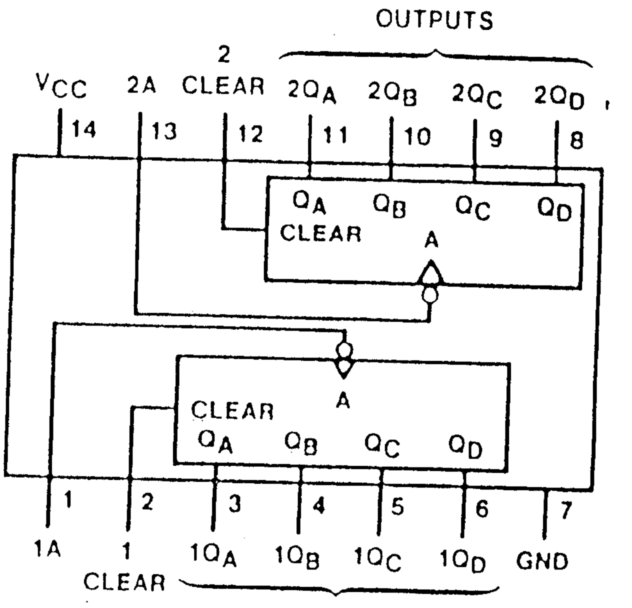


TL/F/5337-1

MM54HC390/MM74HC390

54HC390 (J) 74HC390 (J,N)

Dual-In-Line Package



TL/F/5337-2

MM54HC393/MM74HC393

54HC393 (J) 74HC393 (J,N)

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage(V_{CC})	2	6	V
DC Input or Output Voltage (V_{IN},V_{OUT})	0	V_{CC}	V
Operating Temperature Range(T_A)			
MM74HC	−40	+85	°C
MM54HC	−55	+125	°C
Input Rise or Fall Times			
(t_r, t_f) V_{CC}=2.0V		1000	ns
V_{CC}=4.5V		500	ns
V_{CC}=6.0V		400	ns

DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	V_{CC}	T_A=25°C		74HC T_A=−40 to 85°C	54HC T_A=−55 to 125°C	Units
				Typ	Guaranteed Limits			
V_{IH}	Minimum High Level Input Voltage		2.0V		1.5	1.5	1.5	V
			4.5V		3.15	3.15	3.15	V
			6.0V		4.2	4.2	4.2	V
V_{IL}	Maximum Low Level Input Voltage		2.0V		0.3	0.3	0.3	V
			4.5V		0.9	0.9	0.9	V
			6.0V		1.2	1.2	1.2	V
V_{OH}	Minimum High Level Output Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 20\ \mu A$	2.0V	2.0	1.9	1.9	1.9	V
			4.5V	4.5	4.4	4.4	4.4	V
			6.0V	6.0	5.9	5.9	5.9	V
		V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 4.0$ mA	4.5V	4.2	3.98	3.84	3.7	V
		$\lvert I_{OUT}\rvert \le 5.2$ mA	6.0V	5.7	5.48	5.34	5.2	V
V_{OL}	Maximum Low Level Output Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 20\ \mu A$	2.0V	0	0.1	0.1	0.1	V
			4.5V	0	0.1	0.1	0.1	V
			6.0V	0	0.1	0.1	0.1	V
		V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 4.0$ mA	4.5V	0.2	0.26	0.33	0.4	V
		$\lvert I_{OUT}\rvert \le 5.2$ mA	6.0V	0.2	0.26	0.33	0.4	V
I_{IN}	Maximum Input Current	V_{IN}=V_{CC} or GND	6.0V		±0.1	±1.0	±1.0	μA
I_{CC}	Maximum Quiescent Supply Current	V_{IN}=V_{CC} or GND I_{OUT}=0 μA	6.0V		8.0	80	160	μA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH}, and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC}=5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

AC Electrical Characteristics MM54HC390/MM74HC390

$V_{CC}=5V$, $T_A=25°C$, $C_L=15$ pF, $t_r=t_f=6$ ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
f_{MAX}	Maximum Operating Frequency, Clock A or B		50	30	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_A Output		12	20	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_C (Q_A connected to Clock B)		32	50	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock B to Q_B or Q_D		15	21	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock B to Q_C		20	32	ns
t_{PHL}	Maximum Propagation Delay, Clear to any Output		15	28	ns
t_{REM}	Minimum Removal Time, Clear to Clock		−2	5	ns
t_W	Minimum Pulse Width, Clear or Clock		10	16	ns

AC Electrical Characteristics $C_L=50$ pF, $t_r=t_f=6$ ns (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	$T_A=25°C$		74HC $T_A=-40$ to 85°C	54HC $T_A=-55$ to 125°C	Units
				Typ	Guaranteed Limits			
f_{MAX}	Maximum Operating Frequency		2.0V		5	4	3	MHz
			4.5V		27	21	18	MHz
			6.0V		31	24	20	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_A		2.0V	45	120	150	180	ns
			4.5V	15	24	30	35	ns
			6.0V	13	21	26	31	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_C (Q_A connected to Clock B)		2.0V	100	290	360	430	ns
			4.5V	35	58	72	87	ns
			6.0V	30	50	62	75	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock B to Q_B or Q_D		2.0V	50	130	160	195	ns
			4.5V	16	26	33	39	ns
			6.0V	13	22	28	33	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock B to Q_C		2.0V	60	185	230	280	ns
			4.5V	20	37	46	55	ns
			6.0V	17	32	40	48	ns
t_{PHL}	Maximum Propagation Delay, Clear to any Q		2.0V	55	165	210	250	ns
			4.5V	17	33	41	49	ns
			6.0V	15	28	35	42	ns
t_{REM}	Minimum Removal Time Clear to Clock		2.0V		25	25	25	ns
			4.5V		5	5	5	ns
			6.0V		5	5	5	ns
t_W	Minimum Pulse Width Clear or Clock		2.0V	30	80	100	120	ns
			4.5V	10	16	20	24	ns
			6.0V	9	14	18	20	ns
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time		2.0V	30	75	95	110	ns
			4.5V	8	15	19	22	ns
			6.0V	7	13	16	19	ns
t_r, t_f	Maximum Input Rise and Fall Time		2.0V		1000	1000	1000	ns
			4.5V		500	500	500	ns
			6.0V		400	400	400	ns
C_{PD}	Power Dissipation Capacitance (Note 5)	(per counter)		55				pF
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D=C_{PD}\,V_{CC}{}^2\,f+I_{CC}\,V_{CC}$, and the no load dynamic current consumption, $I_S=C_{PD}\,V_{CC}\,f+I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74HC AC Switching Waveforms and Test Circuits.

AC Electrical Characteristics MM54HC393/MM74HC393

V_{CC} = 5V, T_A = 25°C, C_L = 15 pF, t_r = t_f = 6 ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
f_{MAX}	Maximum Operating Frequency		50	30	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_A		13	20	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_B		19	35	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_C		23	42	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Clock A to Q_D		27	50	ns
t_{PHL}	Maximum Propagation Delay, Clear to any Q		15	28	ns
t_{REM}	Minimum Removal Time		−2	5	ns
t_W	Minimum Pulse Width Clear or Clock		10	16	ns

AC Electrical Characteristics C_L = 50 pF, t_r = t_f = 6 ns (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	T_A = 25°C		74HC T_A = −40 to 85°C	54HC T_A = −55 to 125°C	Units
				Typ	Guaranteed Limits			
f_{MAX}	Maximum Operating Frequency		2.0V		5	4	3	
			4.5V		27	21	18	MH
			6.0V		31	24	20	MH
t_{PHL}, t_{PLH}	Maximum Propagation Delay Clock A to Q_A		2.0V	45	120	150	180	ns
			4.5V	15	24	30	35	ns
			6.0V	13	21	26	31	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay Clock A to Q_B		2.0V	68	190	240	285	ns
			4.5V	23	38	47	57	ns
			6.0V	20	32	40	48	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay Clock A to Q_C		2.0V	90	240	300	360	ns
			4.5V	30	48	60	72	ns
			6.0V	26	41	51	61	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay Clock to Q_D		2.0V	100	290	360	430	ns
			4.5V	35	58	72	87	ns
			6.0V	30	50	62	75	ns
t_{PHL}	Maximum Propagation Delay Clear to any Q		2.0V	54	165	210	250	ns
			4.5V	18	33	41	49	ns
			6.0V	15	28	35	42	ns
t_{REM}	Minimum Clear Removal Time		2.0V		25	25	25	ns
			4.5V		5	5	5	ns
			6.0V		5	5	5	ns
t_W	Minimum Pulse Width Clear or Clock		2.0V	30	80	100	120	ns
			4.5V	10	16	20	24	ns
			6.0V	9	14	18	20	ns
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time		2.0V	30	75	95	110	ns
			4.5V	8	15	19	22	ns
			6.0V	7	13	16	19	ns
t_r, t_f	Maximum Input Rise and Fall Time				1000	1000	1000	ns
					500	500	500	ns
					400	400	400	ns
C_{PD}	Power Dissipation Capacitance (Note 5)	(per counter)		42				pF
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74HC AC Switching Waveforms and Test Circuits.

National Semiconductor

MM54HC157/MM74HC157 Quad 2-Input Multiplexer MM54HC158/MM74HC158 Quad 2-Input Multiplexer (Inverted Output)

General Description

These high speed QUAD 2-to-1 LINE DATA SELECTOR/MULTIPLEXERS utilize microCMOS™ Technology, 3.5 micron silicon gate P-well CMOS. They possess the high noise immunity and low power consumption of standard CMOS integrated circuits, as well as the ability to drive 10 LS-TTL loads.

These devices each consist of four 2-input digital multiplexers with common select and STROBE inputs. On the MM54HC157/MM74HC157, when the STROBE input is at logical "0" the four outputs assume the values as selected from the inputs. When the STROBE input is at a logical "1" the outputs assume logical "0". The MM54HC158/MM74HC158 operates in the same manner, except that its outputs are inverted. Select decoding is done internally resulting in a single select input only. If enabled, the select input determines whether the A or B inputs get routed to their corresponding Y outputs.

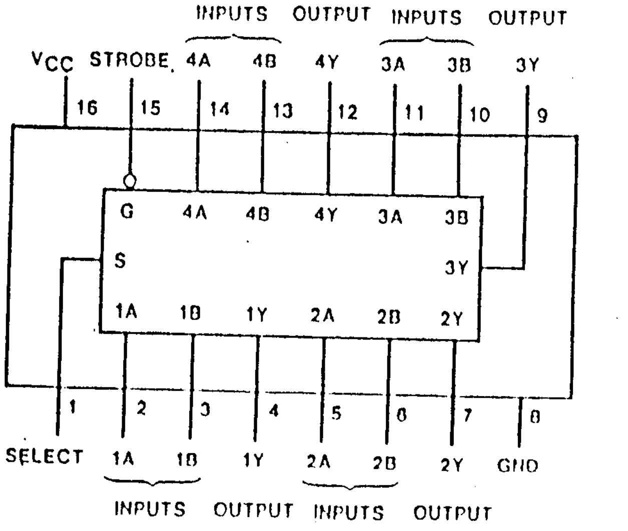
The 54HC/74HC logic family is functionally as well as pinout compatible with the standard 54LS/74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

Features

- Typical propagation delay: 14 ns data to any output
- Wide power supply range: 2–6V
- Low power supply quiescent current: 80 μA maximum (74HC series)
- Fan-out of 10 LS-TTL loads
- Low input current: 1 μA maximum

Connection Diagrams

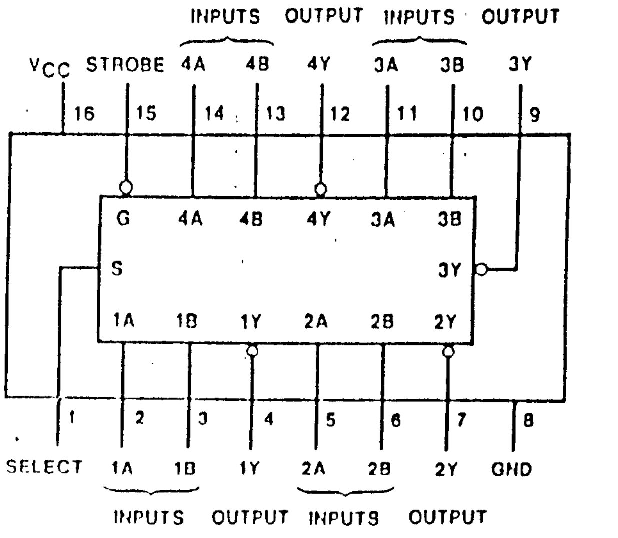
Dual-In-Line Package



MM54HC157/MM74HC157

54HC157 (J) 74HC157 (J,N)

Dual-In-Line Package



MM54HC158/MM74HC158

54HC158 (J) 74HC158 (J,N)

Function Table

Inputs				Output Y	
Strobe	Select	A	B	HC157	HC158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage(V_{CC})	2	6	V
DC Input or Output Voltage (V_{IN},V_{OUT})	0	V_{CC}	V
Operating Temperature Range(T_A)			
MM74HC	−40	+85	°C
MM54HC	−55	+125	°C
Input Rise or Fall Times (t_r, t_f) V_{CC}=2.0V		1000	ns
V_{CC}=4.5V		500	ns
V_{CC}=6.0V		400	ns

DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	V_{CC}	T_A=25°C		74HC T_A=−40 to 85°C	54HC T_A=−55 to 125°C	Units
				Typ	Guaranteed Limits			
V_{IH}	Minimum High Level Input Voltage		2.0V		1.5	1.5	1.5	V
			4.5V		3.15	3.15	3.15	V
			6.0V		4.2	4.2	4.2	V
V_{IL}	Maximum Low Level Input Voltage		2.0V		0.3	0.3	0.3	V
			4.5V		0.9	0.9	0.9	V
			6.0V		1.2	1.2	1.2	V
V_{OH}	Minimum High Level Output Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$≤20 µA	2.0V	2.0	1.9	1.9	1.9	V
			4.5V	4.5	4.4	4.4	4.4	V
			6.0V	6.0	5.9	5.9	5.9	V
		V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$≤4.0 mA	4.5V	4.2	3.98	3.84	3.7	V
		$\lvert I_{OUT}\rvert$≤5.2 mA	6.0V	5.7	5.48	5.34	5.2	V
V_{OL}	Maximum Low Level Output Voltage	V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$≤20 µA	2.0V	0	0.1	0.1	0.1	V
			4.5V	0	0.1	0.1	0.1	V
			6.0V	0	0.1	0.1	0.1	V
		V_{IN}=V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$≤4.0 mA	4.5V	0.2	0.26	0.33	0.4	V
		$\lvert I_{OUT}\rvert$≤5.2 mA	6.0V	0.2	0.26	0.33	0.4	V
I_{IN}	Maximum Input Current	V_{IN}=V_{CC} or GND	6.0V		±0.1	±1.0	±1.0	µA
I_{CC}	Maximum Quiescent Supply Current	V_{IN}=V_{CC} or GND I_{OUT}=0 µA	6.0V		8.0	80	160	µA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH}, and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC}=5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

AC Electrical Characteristics

$V_{CC}=5V$, $T_A=25°C$, $C_L=15$ pF, $t_r=t_f=6$ ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Data to Output		14	20	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Select to Output		14	20	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Strobe to Output		12	18	ns

AC Electrical Characteristics $C_L=50$ pF, $t_r=t_f=6$ ns (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	$T_A=25°C$		74HC $T_A=-40$ to 85°C	54HC $T_A=-55$ to 125°C	Units
				Typ	Guaranteed Limits			
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Data to Output		2.0V	63	125	158	186	ns
			4.5V	13	25	32	37	ns
			6.0V	11	21	27	32	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Select to Output		2.0V	63	125	158	186	ns
			4.5V	13	25	32	37	ns
			6.0V	11	21	27	32	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay, Strobe to Output		2.0V	58	115	145	171	ns
			4.5V	12	23	29	34	ns
			6.0V	10	20	25	29	ns
t_{TLH}, t_{THL}	Maximum Output Rise and Fall Time		2.0V	30	75	95	110	ns
			4.5V	8	15	19	22	ns
			6.0V	7	13	16	19	ns
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF
C_{PD}	Power Dissipation Capacitance (Note 5)							pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D=C_{PD}\ V_{CC}^2\ f+I_{CC}\ V_{CC}$, and the no load dynamic current consumption, $I_S=C_{PD}\ V_{CC}\ f+I_{CC}$.

Note 6: Refer to Section 1 for Typical MM54/74HC AC Switching Waveforms and Test Circuits

National Semiconductor

M54HC4066/MM74HC4066 uad Analog Switch

neral Description

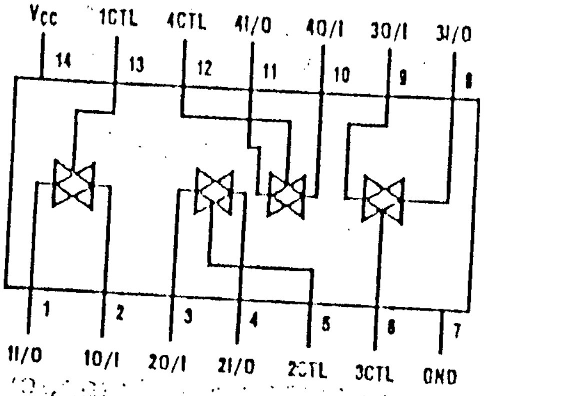
se devices are digitally controlled analog switches utiliz- advanced silicon-gate CMOS technology. These ches have low "on" resistance and low "off" leakages. y are bidirectional switches, thus any analog input may used as an output and visa-versa. Also the '4066 ches contain linearization circuitry which lowers the resistance and increases switch linearity. The '4066 ces allow control of up to 12V (peak) analog signals digital control signals of the same range. Each switch its own control input which disables each switch when All analog inputs and outputs and digital inputs are pro- d from electrostatic damage by diodes to V_{CC} and nd.

Features

- Typical switch enable time: 15 ns
- Wide analog input voltage range: 0–12V
- Low "on" resistance: 30 typ. ('4066)
- Low quiescent current: 80 µA maximum (74HC)
- Matched switch characteristics
- Individual switch controls

nnection Diagram

Dual-In-Line Package



TL/F/5355-1

Top View

Order Number MM54HC4066* or MM74HC4066*

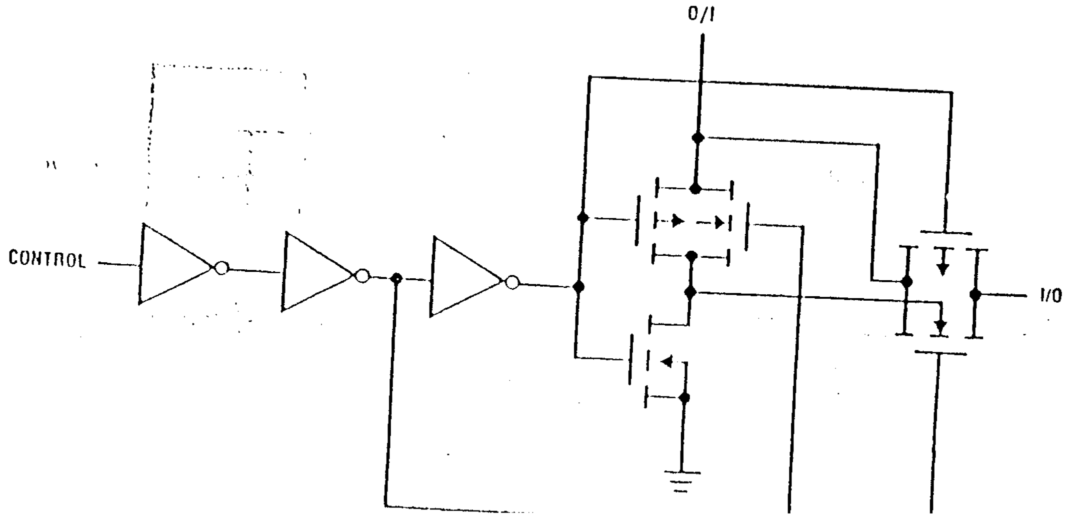
*Please look into Section 8, Appendix D for availability of various package types.

Truth Table

Input	Switch
CTL	I/O–O/I
L	"OFF"
H	"ON"

See the CMOS Logic Databook for Complete Specifications

nematic Diagram



M1881 Video Sync Separator

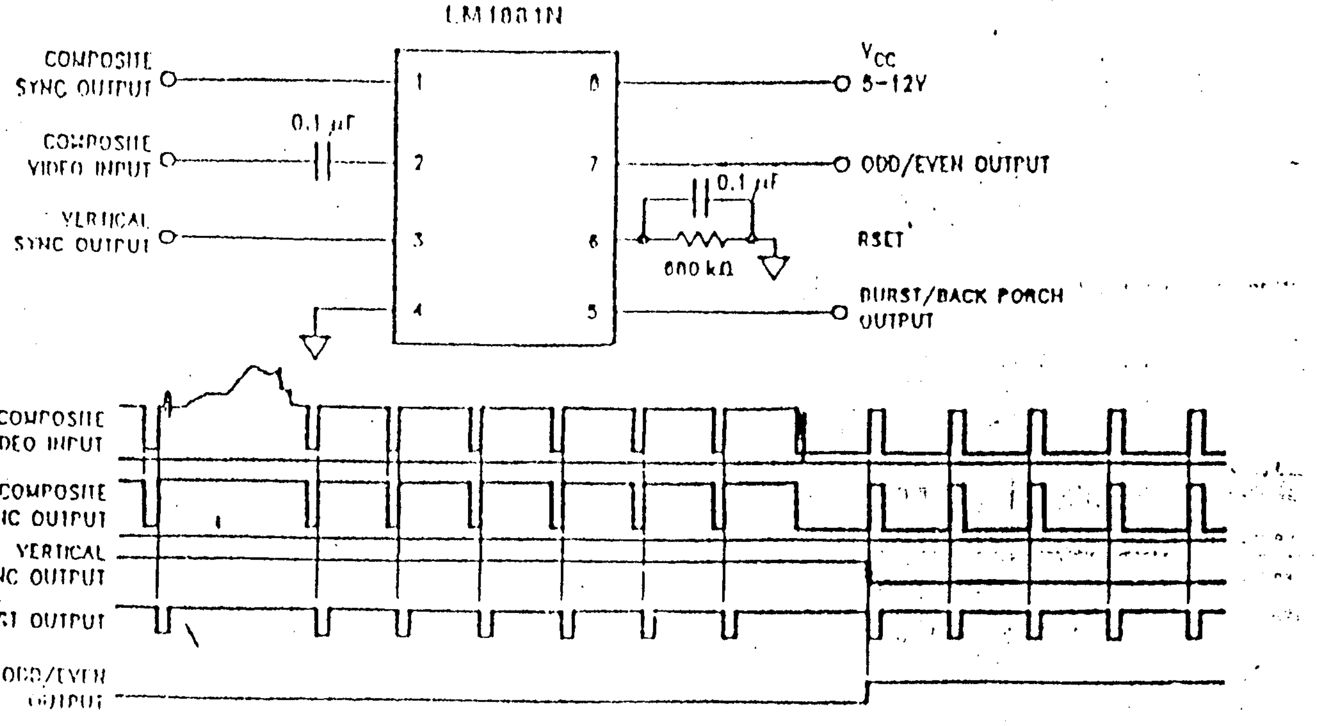
neral Description

LM1881 Video sync separator extracts timing informa including composite and vertical sync, burst/back porch ng, and odd/even field information from standard nega going sync NTSC, PAL*, and SECAM video signals with litude from 0.5V to 2V p-p. The integrated circuit is also ble of providing sync separation for non-standard, fast-izontal rate video signals by changing an external hor-al scan rate setting resistor. The vertical output is pro-d on the rising edge of the first serration in the vertical period. A default vertical output is produced after a delay if the rising edge mentioned above does not oc-within the internally set delay period, such as might be ase for a non-standard video signal.

Features

- AC coupled composite input signal
- >10 kΩ input resistance
- <10 mA power supply drain current
- Composite sync and vertical outputs
- Odd/even field output
- Burst gate/back porch output
- Resistor programmable horizontal scan rate (up to 94 kHz)
- Edge triggered vertical output
- Default triggered vertical output for non-standard vide signal (video games-home computers)

nnection Diagram



Order Number LM1881M or LM1881N
See NS Package Number M08A or N08E

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	13.2V
Input Voltage	3 Vp-p
Output Sink Currents; Pins 1, 3, 5	5 mA
Output Sink Current; Pin 7	2 mA
Package Dissipation (Note 1)	1100 mW
Operating Temperature Range	0°C – 70°C
Storage Temperature Range	−65°C to +150°C
ESD Susceptibility (Note 2)	2 kV
Soldering Information	
Dual-In-Line Package (10 sec.)	260°C
Small Outline Package	
Vapor Phase (60 sec.)	215°C
Infrared (15 sec.)	220°C

See AN-450 "Surface Mounting Methods and their Effect on Product Reliability" for other methods of soldering surface mount devices.

Electrical Characteristics

V_{CC} = 5V; Rset = 680 kΩ; T_A = 25°C; Unless otherwise specified

Parameter	Conditions		Typ	Tested Limit (Note 3)	Design Limit (Note 4)	Units (Limits)
Supply Current	Outputs at Logic 1	V_{CC} = 5V	5.2	10		mAmax
		V_{CC} = 12V	5.5	12		mAmax
DC Input Voltage	Pin 2		1.5	1.3		Vmin
				1.8		Vmax
Input Threshold Voltage	Note 5		70	55		mVmin
				85		mVmax
Input Discharge Current	Pin 2; V_{IN} = 2V		11	6		µAmin
				16		µAmax
Input Clamp Charge Current	Pin 2; V_{IN} = 1V		0.8	0.2		mAmin
R_{SET} Pin Reference Voltage	Pin 6; Note 6		1.22	1.10		Vmin
				1.35		Vmax
Composite Sync. & Vertical Outputs	I_{OUT} = 40 µA; Logic 1	V_{CC} = 5V	4.5	4.0		Vmin
		V_{CC} = 12V		11.0		Vmin
	I_{OUT} = 1.6 mA; Logic 1	V_{CC} = 5V	3.6	2.4		Vmin
		V_{CC} = 12V		10.0		Vmin
Burst Gate & Odd/Even Outputs	I_{OUT} = 40 µA; Logic 1	V_{CC} = 5V	4.5	4.0		Vmin
		V_{CC} = 12V		11.0		Vmin
Composite Sync. Output	I_{OUT} = −1.6 mA; Logic 0; Pin 1		0.2	0.8		Vmax
Vertical Sync. Output	I_{OUT} = −1.6 mA; Logic 0; Pin 3		0.2	0.8		Vmax
Burst Gate Output	I_{OUT} = −1.6 mA; Logic 0; Pin 5		0.2	0.8		Vmax
Odd/Even Output	I_{OUT} = −1.6 mA; Logic 0; Pin 7		0.2	0.8		Vmax
Vertical Sync Width			230	190		µsmin
				300		µsmax
Burst Gate Width	2.7 kΩ from Pin 5 to V_{CC}		4	2.5		µsmin
				4.7		µsmax
Vertical Default Time	Note 7		65	32		µsmin
				90		µsmax

Note 1: For operation in ambient temperatures above 25°C, the device must be derated based on a 150°C maximum junction temperature and a package thermal resistance of 110°C/W, junction to ambient.

Note 2: ESD susceptibility test uses the "human body model, 100 pF discharged through a 1.5 kΩ resistor".

Note 3: Typicals are at T_J = 25°C and represent the most likely parametric norm.

Note 4: Tested Limits are guaranteed to National's AOQL (Average Outgoing Quality Level).

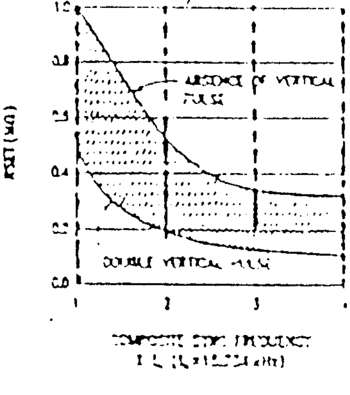
Note 5: Relative difference between the input clamp voltage and the minimum input voltage which produces a horizontal output pulse.

Note 6: Careful attention should be made to prevent parasitic capacitance coupling from any output pin (Pins 1, 3, 5, and 7) to the R_{SET} pin (Pin 6).

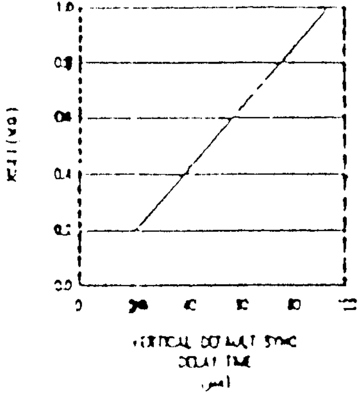
Note 7: Delay time between the start of vertical sync (at input) and the vertical output pulse.

Typical Performance Characteristics

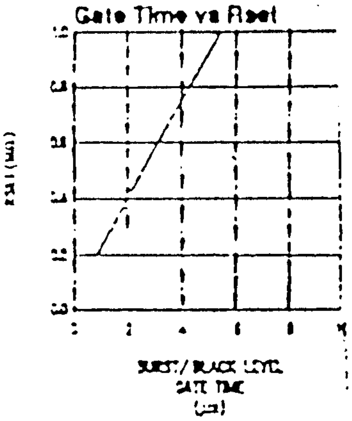
Rset Value Selection vs Composite Sync Frequency



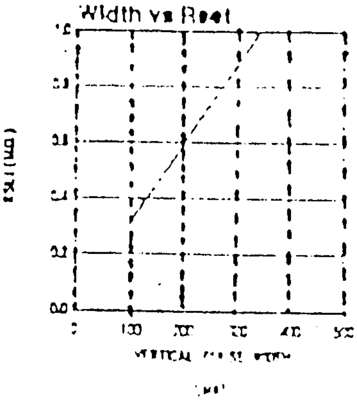
Vertical Default Sync Delay Time vs Rset



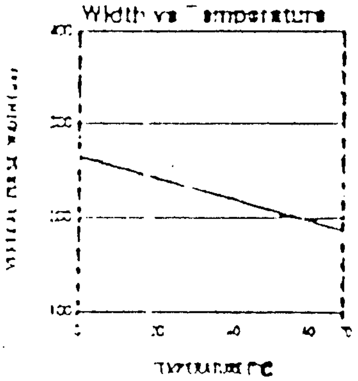
Burst/Black Level Gate Time vs Rset



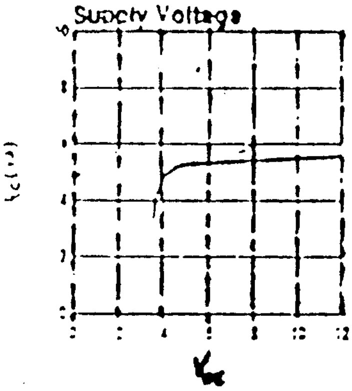
Vertical Pulse Width vs Rset



Vertical Pulse Width vs Temperature



Supply Current vs Supply Voltage



```
                                 C22V10

          clk =| 1|                                             |24|* not used
    din_IBV_7 =| 2|                                             |23|= dout_IBV_2
    din_IBV_6 =| 3|                                             |22|= dout_IBV_4
    din_IBV_5 =| 4|                                             |21|= dout_IBV_7
    din_IBV_4 =| 5|                                             |20|= dout_IBV_5
    din_IBV_3 =| 6|                                             |19|* not used
    din_IBV_2 =| 7|                                             |18|* not used
    din_IBV_1 =| 8|                                             |17|= dout_IBV_6
    din_IBV_0 =| 9|                                             |16|= dout_IBV_0
            e =|10|                                             |15|= dout_IBV_1
          clr =|11|                                             |14|= dout_IBV_3
     not used *|12|                                             |13|= ld
               __________________________________________________
```

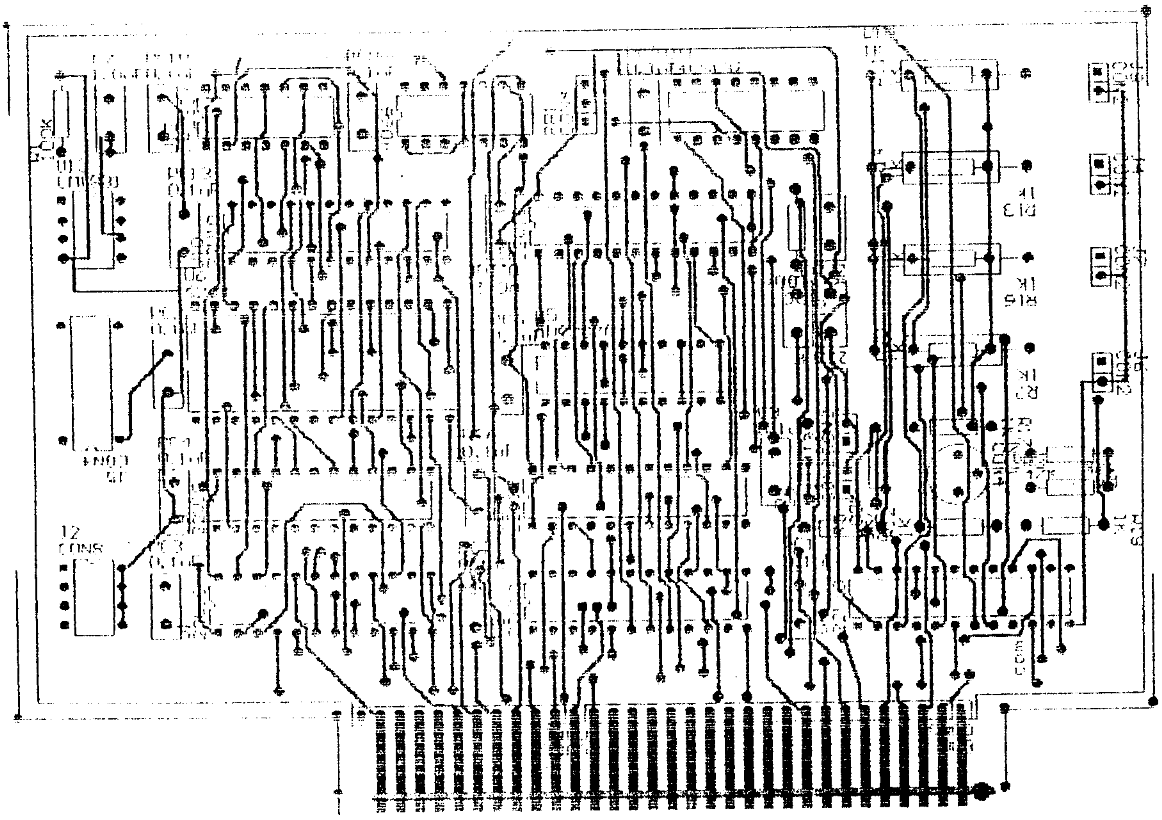
163 counter (u8)

```
                                 C22V10
               __________________________________________________
          clk =| 1|                                             |24|* not used
          rst =| 2|                                             |23|= cout_6
     not used *| 3|                                             |22|= cout_5
     not used *| 4|                                             |21|= cout_3
     not used *| 5|                                             |20|= cout_1
     not used *| 6|                                             |19|* not used
     not used *| 7|                                             |18|= cout_0
     not used *| 8|                                             |17|= cout_2
     not used *| 9|                                             |16|= cout_4
     not used *|10|                                             |15|= cout_8
     not used *|11|                                             |14|= cout_7
     not used *|12|                                             |13|* not used
               __________________________________________________
```

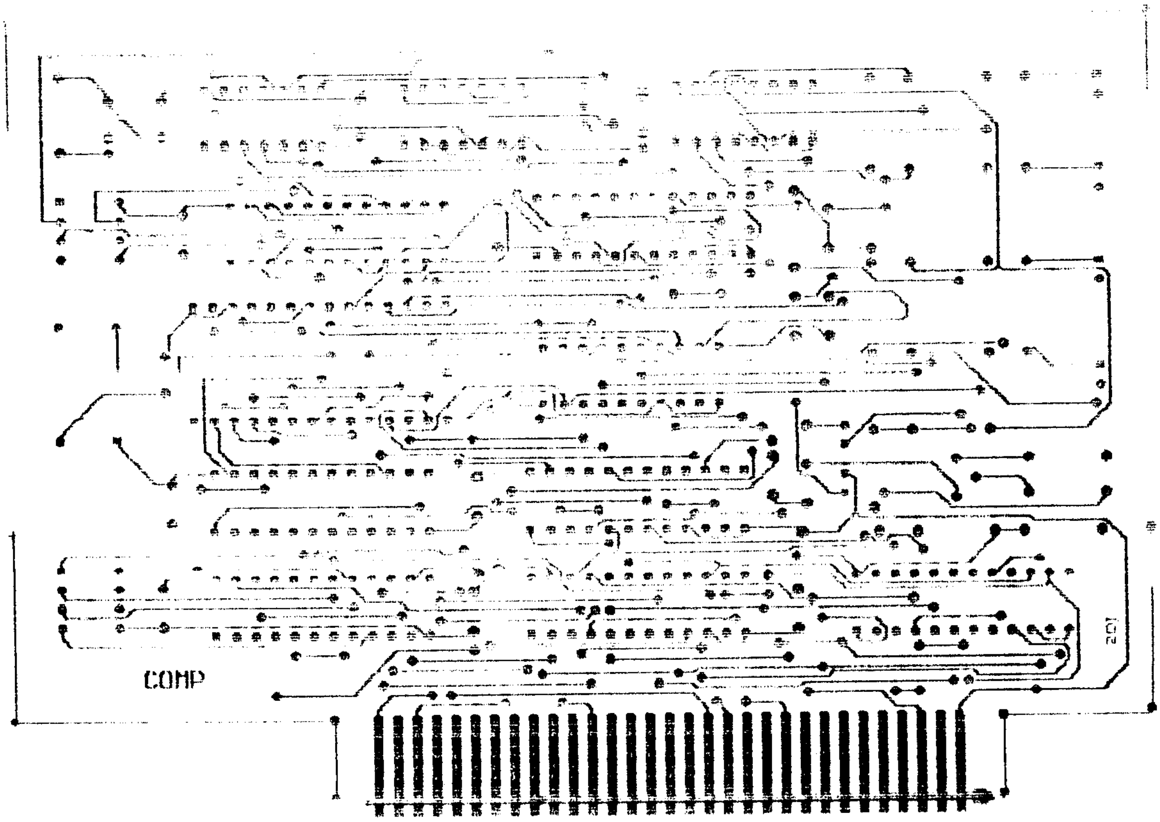
393 counter (u9)

```
                                 C22V10
               __________________________________________________
    b_2_IBV_7 =| 1|                                             |24|* not used
    b_2_IBV_6 =| 2|                                             |23|= a_2_IBV_6
    b_2_IBV_5 =| 3|                                             |22|= a_2_IBV_4
    b_2_IBV_4 =| 4|                                             |21|= a_2_IBV_2
    b_2_IBV_3 =| 5|                                             |20|= a_2_IBV_0
    b_2_IBV_2 =| 6|                                             |19|* not used
    b_2_IBV_1 =| 7|                                             |18|* not used
    b_2_IBV_0 =| 8|                                             |17|= a_2_IBV_1
           e1 =| 9|                                             |16|= a_2_IBV_3
            g =|10|                                             |15|= a_2_IBV_5
     not used *|11|                                             |14|= a_2_IBV_7
     not used *|12|                                             |13|* not used
               __________________________________________________
```

buffer (u3 & u4)



U13
LM420
12
CONS
0.1uF
U2
R13
1K
R16
1K
R7
1K


COMP
102

BIBLIOGRAPHY

1. Modern practice, principles, technology and servicing,

   R.R. Gulati

   Wiley Eastern Limited, New Delhi, Edition - 1991.

2. VHDL for Programmable Logic

   By Kevin Skahill

   ADDISON - WESLEY

3. VHDL

   By Douglas L. Perry.

   Tata Mc-Graw Hill Publishing Company

4. Basic Television and video systems.

   Bernard Grop

   Mc Graw Hill Company, Fifth Edition.

5. Microprocessors and Interfacing Techniques

   By Douglas V. Hall

   Tata Mc Graw Hill Company Limitied, New Delhi,

   Edition - 1991

6. ‘ C’ for dummies

Dan Gookin

Comdex Computer Publishing (A division of PUSTAK MAHAL)

New Delhi - 110 002.

7. CMOS logic data book,

National Semiconductors Corporation, Edition 1988.

8. Colour television and video technology - Principles and application,

A.K. Maini,

CBS publishers & Distributers, New Delhi - 110 032. Edition - 1987.

MICRO COMPUTER WITH ADD ON UTILITIES

PROJECT REPORT

SUBMITTED BY

K. BALAJI

S. GAYATHRI

V. JAYAKUMAR

N. SIVASUBRAMANIAM

UNDER THE GUIDANCE OF

MISS. K. KAVITHA RAJALAKSHMI
DEPARTMENT OF ELECTRONICS &
COMMUNICATION ENGG.
KCT.

MR. B. SUGUMAR
SENIOR HARDWARE
CONSULTANT
C.M.C. LTD.

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF **BACHELOR OF ENGINEERING**
IN ELECTRONICS & COMMUNICATION ENGINEERING
OF THE BHARATHIAR UNIVERSITY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641 006

1991 - 1992

CERTIFICATE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641 006

PROJECT WORK

Register Number 8827D0173, 177, 181, 193.

This is to certify that the Project entitled

MICRO-COMPUTER WITH ADD ON UTILITIES

has been submitted by

Mr/Ms. K. BALAJI, S. GAYATHIRI, Y. JAYAKUMAR, N. SIVA SUBRAMANIAM.

In partial fulfilment of the requirements for the award of the degree of **Bachelor of Engineering in Electronics And Communication Engineering** of the Bharathiar University - 641 046

During the Academic Year 1991 - 92

30.3.92

Guide

Head of the Department

Submitted for the Viva-Voce examination held on..................................

Internal Examiner

External Examiner

CONTENTS

DEDICATED TO OUR

BELOVED PARENTS

AND

TEACHERS

SYNOPSIS

This project titled µC-WAU [MICROCOMPUTER WITH ADD-ON UTILITIES] deals with the various applications of the newfound versatile electronic chip, microprocessor, which surpasses human capability in performing repetitrous operations with an ease.

This project deals with the design, fabrication and performance of the complete EPROM programming cum copier including the add on utilities that are incorporated with this. We have demonstrated few add-on utilities.

This project has I/P, O/P, memory and logical units similar to the computer and it can be rightly called as microcomputer with add-on utilities.

ACKNOWLEDGEMENT

We are grateful to our respected Principal MAJOR. T.S.RAMAMURTHY who encouraged and motivated us all the way to proceed through and complete this project.

We cordially thank our beloved Head of the Department PROF. K.PALANISWAMI, for being our source of inspiration and for his valuable ideas which helped us meliorate this project

We express our gratitude from the bottom of our heart to our internal guide Miss. K.KAVITHA RAJALAKSHMI, who guided us through the right path of completing this project sucessfully with her timely conceptions.

We are greatly indebted to Mr. B.SUGUMAR, Senior Hardware Consultant, CMC, Trivandram, who guided us externally in spite of his heavy schedules, without whom this project wouldn't be a reality. Special thanks to you Sir !

We express our whole hearted thanks to our lab technicians for their timely services.

We are thankful to all the Staff Members and student friends of all the department who gave valuable suggestions with great care and love at various junctures.

INTRODUCTION

With the field of Electronics extendig into never ending new frontiers at an incredible speed and microprocessor setting new trends everyday this project has micrprocessor as its core of the heart.

This project named μC-WAO [MICROCOMPUTER - WITH ADD-ON UTILITIES] uses Intel's versatile microprocessor 8085 with 2716 [EPROM], 6116 [RAM], 8212 [I/O port cum latch] with simple AND, OR, NAND, NOT and NOR gates to achieve the task.

Intel's 8085 has a simple set of software instructions which could be easily understood. In addition, much of the software programs for experimental use and for general purpose industrial control applications can be written early with 8085 instruction set. Depending on one's interest, the kits capability can be easily extended. This performs all the arithmetic operations, can store, can copy and so it can be rightly called as a Micro Computer.

The Add-on utilities which are included in this project are

1) Audio Cassette Interface and

2) ADC and DAC.

Audio cassette interface enables the user to make use an audio tape as a temporary storage medium which is helpful in case of lengthier programs or programs which are still on the roads of development. This interfacing is achieved with CD 4051 [Analog Mux], CD 4093 [Schmitt NAND] used with an envelop detector and a simple software routine.

The second Add-on utility is the ADC. This has numerous applications wherever an analog signal is to be converted to digital signal. Then utility uses 0804 chip with a simple software. This also includes DAC using 0800 chip.

The project being the basic block of larger computer systems, can be extended to perfrom any task, with sufficient modifications.

CHAPTER I

KEYBOARD DESCRIPTION

The microprocessor receives a series of instructions which are stored in memory. The EPROM memory stores some important instructions necessary for the user to load and read data into the other memory, the RAM. Inorder to make the microprocessor do some specific work, it must have atleast one input port and one output port.

Let us now look into a, microcomputer, or even a simple calculator for that matter, made by using a microprocessor. Its input port is the Keyboard, which consists of a set of keys that can be used to load data or execute commands such as +, -, x, clear etc. The display of a calculator is the external output port. But when we have an eight digit decimal display, it may use in practice atleast two 8 - bit binary output ports. Thus a simple calculator, which can be called a special purpose microprocessor unit, houses two output and one input ports.

The larger the number of input and output ports, the greater is the versatility of the system. The 8085 has provision for 256 input and output ports each. These are provided and accessed by an 8-bit address, using its eight address lines called A0 to A7.

1.1 PARALLEL INPUT / OUTPUT :-

In the parallel I/O data mode, the 8085/8085A accepts eight bits of data on its data bus from peripherals such as switches, hex keyboards and A/D converters. Similarly, it sends out eight bits of data on its data bus to output devices such as LED's, Seven Segment LED's and D/A converters. Each I/O device is assigned a binary address, called a device address or port number, through an appropriate interfacing logic circuit. When the microprocessor executes a data transfer data instruction for an I/O device, it sends the appropriate address on the address bus, sends the control signal, enables the device and transfers data. When these I/O tasks are accomplished by means of Input/Output (IN/OUT) instructions, the process is called peripheral I/O.

When I/O tasks are accomplished by means of memory-related data transfer functions (LDA, STA etc), the process is called memory ma'pped I/O.

1.2 TYPES OF INTERFACING DEVICES :-

Interfacing devices may be classified into two categories:

1. General Purpose Peripherals

2. Special Purpose Peripherals
   (also known as dedicated function peripherals)

We use the term peripheral for these devices as well as for the other I/O devices, such as printers and floppy drives that would be interfaced to the uP through them.

General purpose peripherals are devices that perform a specific task but may be used for interfacing a variety of I/O devices to the uP. Examples of general purpose peripherals are ;

a) An I/O port

b) Programmable Peripheral Interface (PPI), also known as Peripheral Interface Adapter (PPA)

c) Programmable Interrupt Controller

d) Programmable DMA controller

e) Programmable Communications Interface

f) Programmable Internal Timer.

Special Purpose peripherals are devices that may be used for interfacing a uP to a specific type of I/O device. There are peripherals much more complex and therefore relatively more expensive than the general purpose peripherals. Some of the special purpose peripherals are,

a) Programmable CRT controller

b) Programmable Floppy Disk Controller

c) Programmable Hard disk Controller

d) Programmable Keyboard and Display Interface.

The above description describes the different types of interfacing devices. Interfacing circuit for an I/O device is determined primarily by the instructions to be used for data transfer. An I/O device can be interfaced with the 8085/8085A microprocessor either as a peripheral I/O or as memory mapped I/O. In the peripheral, the instructions IN/OUT are used for data transfer, and the device is identified by an 8-bit address.

1.3 PERIPHERAL I/O INSTRUCTIONS :-

The 8085/8085A microprocessor has two instructions for data transfer between the processor and the I/O devices. The instruction IN (code DB) inputs data from an input device (such as keyboard) into the accumulator, and the instruction OUT (code D3) sends the contents of the accumualtor to an output device such as LED display. These are 2-byte instructions with the second byte specifing the address or the port number of an I/O device. For example, the OUT instruction is typically written as follows.

Memory	Machine	Mnemonics	Comments	Address	Code
2050	D3	OUT 01h	Output accumulator contents to the port with device address 01h.		
2051	01				

If the output port with the address 01h is designed as an LED display, the instruction OUT will display the contents of the accumulator at the port. The second byte of this OUT instruction can be any of the 256 combination of eight bits, from 00h to FFh. Therefore, the 8085/8085A can communicate with 256 different output ports with device addresses ranging from 00h to FFh. Similarly, the instruction IN can be used to accept data from 256 different input ports.

1.4 OUT INSTRUCTION (8085) :-

In the first machine cycle, M1 the 8085 place the high order memory address 20h on A15-A8 and the low order address 50h on AD7-AD0. At the same time, ALE goes high and IO/$\overline{M}$ goes low. The ALE signal indicates the availability of the address on the AD7-AD0 and it can be used to demultiplex the bus. The IO/$\overline{M}$, being low, indicates that it is a memory related operation. At T2, the microprocessor sends the $\overline{RD}$ control signal which is combined with IO/$\overline{M}$ to generate the $\overline{MEMR}$ signal, and the processor fetches the instruction code D3 using the data bus. When the 8085 decodes the machine code D3, it finds out the instruction is a 2 byte instruction an and that it must read the second byte.

In the second machine cycle, M2 (Memory read), the 8085 places the next address, 2051h, on the address bus and gets the device address 01h via the data bus.

In the third machine cycle, M3 (I/O Write), the 8085 places the device address 01h on the low order(AD7-AD0) as well as the high order (A15-A8) address bus. The IO/$\overline{M}$ signal goes high to indicate that it is an I/O operation. At T2, the accumulator contents are placed on the data bus, followed by the control signal $\overline{WR}$. By ANDing the IO/$\overline{M}$ and $\overline{WR}$ signals, the $\overline{IOW}$ signal can be generated to enable an output device.

1.5 IN INSTRUCTION :-

When the microprocessor executes the instruction IN to accept data from an input port, events similar to the execution of the OUT instruction takes place. During the machine cycle M1, the address of input port is placed on the address bus, the control signal IOR is sent to enable the port and data are transferred from the input device (such as keyboard) to the accumulator via the data bus.

1.6 DEVICE SELECTION AND DATA TRANSFER :-

In general, peripherals are connected in parallel on the data and address buses. To select an appropriate peripheral the device address on the address bus and the control signal during the M3 cycle can be used as follows:

1. Decode the address bus to generate a unique pulse corresponding to the device address on the bus. This is called the device address pulse.

2. Combine (AND) the device address pulse with the control signal to generate a device select pulse that is generated only when both signals are asserted.

3. Use the device select pulse to activate the interfacing device (I/O port)

The various interfacing units that we have used are:

i. Input / Output ports
ie., Keyboard and display unit

ii. EPROM programmer and copier

iii. RAM-memory board (upto 10K)

iv. Ports and memory expandable board

v. A/D and DAC (mulitchannel board) etc., The project helps us to built any electronic control circuitaries for ant applications from simplest I/O intergface to robotics and various A-I units. Let us explain each item quoted above one by one in a simpler way. The project includes few interfacing units. The report describes about each unit assuming to some of the following.

What is its necessary for such interfacing device.

Explain about the hardware.

Explanation about the software and various control signals.

Typical troubleshooting problems:

What improvement that can be done over the existing.

Let us start with keyboard interfacing.

1.7 THE INPUT BOARD :-

The input data which consists of sets of 8-bits binary information, can pertain to the software instructions or to number of data. For inputting data in the form of numbers, a keyboard consisting of a set of small push to close keys or switches are necessary. In microprocessor terminology, numbers are not dealt within the decimal way but in binary form.

1.8 KEYBOARDS :-

Sometimes small keyboards with only a fewkeys (similar to touch-one telephone dialing keyboards) are used in industrial applications. These small assemblies of keys are generally called keypads.

When an operator depress a key, electric signals must be generated which will unable the device (or other device) to determine which keys was depressed. This is called encoding. The encoding process is dependent on the mechanism used to make the induvidual keys in the keyboard.

The most desired method for encoding is based on the use of keyboard swutches, which contain a switch similar to the push button switch used in many electric devices. When the plunger is depressed, the contacts of the switch in the housing are closed, and the two terminals at the output are effectively connected. When the plunger is up, the switch in the housing is open, and the terminals are not electrically connected.

It is a good idea to load the values on the output line into a flip-flop register before the device reads the outputs. This has the advantage of storing the values until the device can read them, particularly if the keyboard operator raises the key before the device can respond.

In which a flip-flop is used on each output, and a strobe is generated to load the flip-flops, by using a delayed inverted pulse signal generated whenever anyone of the output lines from the encoder goes high. The delay is inserted to compensate the signal slieuing, where signals arrive at the output lines at different times because of differing delays through the wires in the system. The delay must be adequate to accomadate the largest delays that may occur. Also, the length of the strobe pulse should be short compared to the shortest line a key might be depressed. (A delay of 1ms and a pulse of 1ms would be reasonable).

Here we are assuming that the switch contact do not bounce, as is the case with some switch. If the contacts do bounce, the output signals must be "smoothed" and various circuits are available.

[MICRO-COMPUTER WITH ADD-ON UTILITIES

The basic division of keyboard is (1) the electromechanical keyboard, which includes the switch type just explained and (2) the solid state keyboard.

There are several basic mechanisms for solid-state keyboards. Capacitor type have mechanisms which vary the capacity of a capacitor when a key is depressed. These are low-cost keys, often used in keypads and other cost-conscious keyboards. Hall-effect keyboards are more expensive, but have long life and good key feel, as do ferrite core and photo optic keyboards. Each of the basic mechanisms has different problems with regard to encoding the keyprinters output into a coded form usable by a computer.

IC packages for encoding are made by several manufacturers and can include smoothing or debouncing for contacts and sometimes key rollover protection, which protects against two keys being depressed at the same time.

1.9 KEY DEBOUNCE :-

It is necessary that the bouncing of the key should not be read as an input. The key bounce can be eliminated from input data by the key-debounce technique, using either hardware or software.

1.10 KEY DEBOUNCE USING SOFTWARE :-

In the software technique, when a key closure is found, the microprocessor waits for atleast 10ms before it accepts the key as an input.

The uP begins by reading the input port and checks whether the previous key has been released. This will eliminate the problem that would otherwise occur if someone were to press a key and hold it for a long time. Once the key is released, it is debounced by waiting 10ms. The program reads the keyboard and checks for a key closure. If a key closure is found, it is debounced again by waiting 10ms and the binary code corresponding to the key pressed found either by the table look-up procedure or by setting up a counter.

Our project uses a hexadecimal keyboard with discrete IC encoding process. It uses no matrix type or, programmable I/O interface. Our circuit is simpler and easy. Let us see how it works.

The keys or switches in the keyboard, when pressed, will indicate the corresponding data word for that key to appear at the 8-lines entrance of the inout port. For this a decoding circuit is necessary. For instance, if the key numbered 3 is pressed, the word must be decoded as 0000 0011. For the key A, the word should be 0000 1010.

The keyboard can be hexadecimal or an octal type. Industry standard is now using hex format. Hence a hex keyboard, which needs, 16 keys, is recommended for the kit. Sixteen push switches and one - shift - key like extra push switch (command key) are required. Five more switches are also mounted on the keyboard and these are to be connected to the microprocessor board later. One end of all the (16+1) switches are connected together to Earth terminal. The other ends, which consists of 16 wires and one extra wire from the command key are to be wired by a flat ribbon cable to the circuit board.

The 16 keys are for selecting the hex characters 0 to F. The command key is not to be pressed while introducing these hex characters. If the command key is pressed together with any one of the keys 0 to 5, then it inputs the word that is interrepted (by the EPROM software on the microprocessor board) as one of several command functions such as "data store and increment memory", "set high address", "set low address", go to execute the program" etc.,

These commands are necessary to load the program from the keyboard by keying the instruction words one-by-one, checking it as it gets stored in RAM memory and finally asking the microprocessor chip to execute them either in full or part. Of course, the results of whatever program that is executed will show up on the display via the output ports.

1.11 THE WORKING OF THE INPUT/OUTPUT PORT CIRCUIT :-

The decoder consists of eight - input NAND gate ICs (7430). For example, gate A has 8 input lines connected to the keys, 1-3-5-7-9-B-D-F. A look at the hexadecimal table will show that all these numbers give the last bit as '1'. Hence the output of the NAND gate becomes hicg whenever any one of these keys is pressed.

Like wise, the remaining 8-input NAND gates B, C and D decode the second, third and fourth bits of the hexadecimal 4 - bit word. The keyboard generates one hexadecimal word. It might be recalled that the data word is an 8 bit word and the hex word from the keyboard gives only a 4 - bit word. So to get the 8 - bit word, say pertaining to an instruction we must use the keyboard two lines. By making use of a suitable shifting instruction in the microprocessor software program, the 8-bit word will be generated as two "nibbles".

The four outputs from gates A,B, C and D are led to the entrance of the input port IC 74126. In addition, the four invertor gates of 7404 and the "0" key are all given to the inputs of the NAND gate E. So when anyone of the outputs A, B, C and D go high, or if the "0" key is pressed, the gate E has its output going high. This is connected to the other 74126 IC of the input port to be output as the bit D7 during the Read pulse.

Further, if the command key (CK) is pressed, the bit D6 will be low, otherwise it remains high. Thus when the D6 data bit is high, the 4-bit word D0-D3 that is input is considered as a hex number. When D6 is low, ie., when command key is pressed, the 4 - bit word D0-D3 is considered to be representing any of the command word instructions.

The bit D7 will always go high if any one of the 16 Keys is pressed. Thus it indicates to the microprocessor whether at all a key has been pressed. Such an information is sometimes called as a "Strobe" bit.

Since the remaining bits D4 and D5, are not used, they are left free. These bits will be read as 1's since their inputs (to the 74126) are left free.

The two 74126 IC's are used to create an 8-bit input port. The 74126 is only a buffer IC. Each of the four buffers in a 74126 has a control input line, which enables the buffer to transfer the bit at the input to the output when the control lines goes high momemtarily. Thus, these control lines of all the buffers in these two IC's are connected together and brought out for giving the read pulse from the microprocessor.

The buffers isolate the microprocessor from the input lines except during a reading operation. This read pulse must be a positive going pulse for the 74126 IC's. In fact, this pulse would be generated by the circuitry in the output port using the $\overline{I/OR}$ pulse from the microprocessor, together with the address lines, using suitable gates.

The input port in this kit is given the address 02. Thus, the instruction IN 02 would cause the data from the keyboard to be read. Any other, say IN 05 instruction, would choose only an address-5 port (if such a port is actually constructed and wired-up), and it would read from that port then.

Additional input ports can be constructed by the user using similar 74126 or 74LS126 IC pairs, and interconnected to decoded keyboards, DIP switches, analog to digital converters etc. An anlog to digital converter is useful to convert any analog signal such as voltages into number for subsequent processing in the microprocessor by a suitable program.

[MICRO-COMPUTER WITH ADD-ON UTILITIES

1.12 CONSTRUCTION :-

The keyboard is arranged on a flat Hylam sheet, using a pattern of 5x4 keys arranged in 4 coloumns and 5 rows, with a spacing of 2 cm in between them. The command key (CK) is fitted at the bottom to the right and the reset key (to go to the microprocessor board) to the bottom left. One end of all the 16 number keys and CK key go to ground. The other ends are connected using a multi-wire flat ribbon cable to the PCB. The four extra keys are needed for Interrupt functions, and they are used with the microprocessor board.

This PCB can be easily wired using either a general purpose IC type of printed board by point-to-point linking sockets are not necessary for the TTL ICs used in this board but it is advisable to provide sockets for all.

1.13 TROUBLE SHOOTING :-

A simple LED probe is needed for testing. This can be made by connecting a small 330 ohm resistor (¼ W) in series with a simple LED. The 5V supply is connected to the board and to the probe. Press any one key and see if the entrance points of the 74126 get the data, according to the key pressed. The LED probe will be lit if it is a "0" and will not be lit for a "1".

An alternative probe using 7404 invertor can be used if one wants to get the indication of LED lit with logic "1" and not lit when logic is "0". Then ground the read pulse point of the input port. The data will not appear at the data bus points for the key pressed, because the 74126 ICs are then in the open-circut state or the so called "Tri-state" condition.

CHAPTER II

THE INPUT OUTPUT BOARD

After considering the circuit diagram of the keyboard and the Input board, let us now study the circuit diagram of the output port.

2.1 THE OUTPUT PORTS:-

It is possible to have 256 output ports using the 8-bit address lines A0-A7 from the microprocessor. Usually, such a large number of ports for Input/Output are not needed for simple microprocessor applications.

We shall have three output ports in the unit described for construction of these, two will be used for the display that indicates data and address using the monitor program. The third port, which is extra is meant for experiments using the kit.

So far the inside details of the program have not been discussed sufficiently. We shall do this step by step, in a practical way.

We have already discussed that every 8-bit data which goes into or out of the microprocessor passes via the eight lines of the data bus. This bus has a bi-directional character because it passes data into or out of a microprocessor. The data is passed as per instruction whether it is an input or read instruction or an output or write instruction. Note that "Input and Output" refer to the ports while "Read and Write" refer to the memory. Both ports and memory are external to the microprocessor, and the data bus links of of them.

The address lines of the microprocessor enable the proper memory address to the decoded, and hence the data can be written into or read from that particular memory location. For example, the instructions

LDA 16 04

cause the accumulator to be loaded with the contents of the memory whose where address is 0416 (hex) or 0000, 0100, 0001, 0000 (16 bit binary address) A15..A0.

Thus, a memory of upto 64 K Words (1K=1024) can be accessed. Such a memory size is really needed if one builds a microcomputer of some power using 8085. Simpler systems either for a microcomputer or industrial control, or other uses, can do with lesser memory. For quite simple and devoted applications, ordinarily 2K work memory is satisfactory.

2.2 CIRCUIT DIAGRAM OF THE OUTPUT PORTS :-

An output port latches (Catches and stores) the data bus word the moment an OUT instruction is execuated by the microprocessor. For this purpose, the $\overline{IO/W}$ line (input or output write) coming from the microprocessor is used in conjunction with the address of the port to generate a pulse that causes D-flip-flops of a latch to catch and store the data word from the data bus.

The data on the microprocessor data bus keeps on changing as it executes instruction after instruction at a fast pace. As it executes, say the OUT 01 instruction, the relevant data appears on the data bus and the address 1 appears as high on the address bus. The output port 1 must be wired to get a strobe pulse when the address line 1 and also when the $\overline{I/OW}$ pulse are active.

Likewise the OUT 02 instruction will make address bit A1 high along with the active $\overline{I/OW}$ low signal. These two must together give a strobe pulse to the port 2 latch to catch and store the data. If we donot have a port 2 wired to the system, then the data will just appear on the data bus while executing the out 02 instruction but it will be lost, as it cannot be latched or stored.

We have to decode the address lines for port selection. In our kit, as mentioned earlier, there are basically three ports and one may add some more ports later. We are not however, going to use the full provision of 256 ports in any case. Hence we need not use all the address bits. So we may decode only the bits A7,A2,A1 and A0 and forget about the bit A3-A6. We therefore use only 4 active address bits here for decoding and selecting the I/O ports, the remaining are left undecoded.

With A7 low, we can have a combination of eight ports using the three (A0, A1 and A2) lines. Thus, with A7 low and A0=0, A1=1, A2=0 we get port 2. with A7 low and A0=1, A1=1, A2=0 we get port 3 and so on. With A7 high, we can get eight more possible address for accessing another set of eight ports. For instance, the EPROM programmer board, uses the A7=high address combination.

The output ports used in the hardwiring of the kit uses only A7 low combination. The A7 high combination is left free for the user to expand the kit by adding more ports by an exactly similar circuit like this, but with A7 high instead of low. An inverter from the A7 line to feed pin no.12 of the 7442 is to be added then.

Referring to the circuit of the output port, the 7442 is used for decoding the address lines A7, A0, A1 and A2. It has ten outputs, and one of these goes low for each selected address. For example, when A7-A2-A1-A0 are 0010 (address 2), pin no.3 of the 7442 goes low. This going low information is combined with $\overline{I/OW}$ low going pulse from the microprocessor in one NOR gate (a quarter of 7402) whose output goes momentarily high. This high going pulse is connected to the clock output of the four D-flip-flops connected in the 7475 ICs. A 7475 contains such latches in it and so two 7475's are needed to latch the eight bits of the 8-bit data bus. The data bus lines are connected to the

D-inputs of the 7475's flip-flops. Six such 7475's are needed for the three output ports on the kit.

The port addresses selected are 1, 2 and 4 for these ports wired in the kit. Port 1, is used for wiring to the digit cathodes of the eight of digits of the multi-digit LED display. Of course, only six of these digits are used, because we have to provide two gaps in between.

Hi Add		Low Add	Data
0 4	-	0 0	C 2

The flip-flops of the 7475s used in port 1 sink the currents from the cathodes of these digits through eight BC 147B transitors. Port 2 is used for wiring to the seven segments (and decimal point) of each LED display, ie., the anodes. For example, if port 1 has a latch information as

```
0 0 0 0   0 0 0 1
```

and port 2 has the information

```
0 0 0 0   1 1 0 0
g f e d   c b a .
```

(a to g denote the segments of LED).

then the display will be blank for all but the last digit and it will show a 1 as below (segments b and c alone are lit).

- - - -   - - - 1

By using software in our monitor program (kept stored in the EPROM at 0000 to 01FF), we make repeated (one be one) display of the relevant information in all the eight digits. Due to persistence of vision, however, we find the display's all eight digits to be glowing continously. This requires a scanning program that sequentially presents the bits one by one to the digits and does it repeatedly at a rate which shows no flicker. Such a display can be called a "Software Scanned Display".

For converting the binary data into the hexadecimal format also we make use of software in the monitor. A table stores the information as to which segments should glow for each hex digit. For example, for number 1, segments b and c may glow. And for A, all but segment d may glow, and so on. By having a table the data (of 16 rons) the data "nibble" can be converted into hex form, before outputting it through 2, that causes the LED segments to be lit accordingly.

Output port 4 is used for lighting the eight LED's current limit resistors R1 are used for each LED. The $\overline{Q}$ outputs of the 7475 are used in this case so that the LEDs may glow for a '1' bit and be dark for a '0' bit seven current limiting resistors (18 to 220 ohm, 1 W each) are used for the segements of the multi digit display.

2.3 SEVEN SEGMENT LED :-

A seven segment LED consists of seven light emitting diode segments and one segment for the decimal point. To display a number, the necessary segments are lit by sending an appropriate signal for current flow through diodes. For example, to display an 8, all the segments must be lit.

Seven segment LED's are available in two types: Common Cathode and Common Anode. Current flow in these diodes should be limited to 20 milli amps.

The seven segments A through G, are usually connected to data lines D0 through D6 respectively. The binary code required to display a digit is determined by the type of the seven segment LED, the connections of the data lines, and the logic required to light the segment. For example to display digit 7, at the LED the requirements are as follows.

1. It is a common anode seven segment LED and a logic '0' is required to turn on the segment.

2. To display digit seven, segement A, B and C should be turned on.

3. The binary code should be,

Data lines	D7	D6	D5	D4	D3	D2	D1	D0	
Bits	X	1	1	1	1	0	0	0	= 78 + 1
Segments	NC	G	F	E	D	C	B	A	

The code for each digit can be determined by examining the connections of th data lines to the segements and logic requirements.

To display ASCII characters the following code words are used.

? A6 , A EE , B F8 , C 72 , D BC , E F2 , F E2 ,

G 7A , H EA , I 08 , J 3C , K EC , L 70 , N A8 ,

O B8 , P E6 , Q B9 , R A0 , S DA , T F0 , U 7C ,

V 38 , X A4 , Y DC , Z B6 , - 80 , b̸ 00 ,

0 7E , 1 0C , 2 B6 , 3 9E , 4 CC , 5 DA , 6 FA ,

7 0E , 8 FE , 9 CE .

CHAPTER III

THE MAIN BOARD :-

The microprocessor cum memory board can now be assembled and used along with them. In this main board, we have the microprocessor and its associated IC's and the memories-the EPROM and RAM.

A Glass Epoxy type printed circuit board is necessary for the main board. Full size PCB layout has been drawn so that readers may follow the same conveniently. Since a double sided PCB layout (which is common in microprocessor kits) has not been employed, a good number of jumper connections are required to be make, as indicated on the component side diagram shown. Sockets are to used for all the IC's, including the 74-Series IC's to avoid trouble later while checking up.

After inserting the sockets, the jumpers shown should be wired with thin hook-up wires. Then the power supply pins of the IC's should be connected to the +5V and Ground.

The 4 MHz or 3.5 MHz crystal must be soldered between pins 1 and 2 of the microprocessor IC 8085. The crystal must be handled with care. Overheating its pins or dropping on the floor will surely damage the crystal

A 0.02 mfd ceramic capacitor must be connected on the PCB between +5V and Ground pins. It may be necessary to connect a 22pf ceramic capacitor between pin 2 and Ground of the 8085.

The ICs used for this board are all quite expensive. They should be handled with care while inserting into their sockets, after completing the board wiring.

3.1 MICROPROCESSOR ARCHITECTURE AND ITS OPERATIONS :-

The microprocessor is a programmable logic device designed with registers, flip-flops and timing elements. The microprocessor has a set of instructions designed internally, to manipulate data and commnunicate with peripherals. This process of data manipulation and communication is determined by the logic design of the microprocessor called the architechture.

All the various functions performed by the microprocessor can be classified in three general categories.

1. Microprocessor initiated operations.
2. Internal data operations.
3. Peripheral initiated operations.

The microprocessor functions listed above are explained here in relation to 8085 MPU.

MICROPROCESSOR INITIATED OPERATIONS :-

The MPU performs primarily four operations:

a.	Memory Read	: Reads data from memory
b.	Memory Write	: Writes data into memory
c.	I/O Read	: Accepts data from input devices
d.	I/O Write	: Sends data to output devices

The 8085 MPU performs these functions using three sets of communication line called buses; the address bus, the data bus and the control bus.

ADDRESS BUS :-

The address bus is a group of sixteen lines generally identified as A0 to A15. The address bus is unidirectional; bits flow in one direction - from the MPU to peripheral devices. The MPU uses the address bus to perform the first function identifying a peripheral or a memory location.

In a computer system, each peripheral or memory location is identified with a binary number, called an address and the address bus is used to carry a 16-bit address. The 8085 MPU with its sixteen address lines is capable of addressing 2 16 = 65536 (generally known as 64K) memory locations. Most 8-bit microprocessor have sixteen address lines.

DATA BUS :-

The data bus is a group of eight lines used for data flow. These lines are bidirectional - data flow in both directions between the MPU and peripheral devices. Transferriing the data is performed bu the MPU using the data bus.

The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (2 8 = 256 numbers). The largest number that can appear on the data bus is 1111 1111. The data bus determines the word length and register size of a microprocessor. Thus the 8085 microprocessor is called an 8 bit microprocessor.

CONTROL BUS :-

The control bus is comprised of various single lines that carry synchronization signals. It provides the timing signals. These are induvidual lines that provide a pulse to indicate an MPU operation. The MPU generates specific control signals for every operation it performs. These signals are used to identify a device type with which the MPU tends to communicate.

3.2 INTERNAL DATA OPERATIONS AND THE 8085 REGISTERS :-

The internal architecture of the 8085 microprocessor determines how and what operations can be performed with the data. These operations are,

1. Store 8-bit data
2. Perform arithmetic and logic operations
3. Test for conditions
4. Sequence the execution of instructions
5. Store data temporairily during exection in the defined R/W memory locations called the stack.

To perform these operations, the microprocessor requires registers, an arithmetic logic unit and control logic and internal buses.

REGISTERS :-

The 8085 has six general purpose registers to perform the first operation, that is to store 8-bit data during a program execution. These registers are identified as B, C, D, E, H and L. They can be combined as register pairs, BC , DE and HL - to perform some 16-bit operations.

These registers are programmable, meaning that a programmer can use them to load or transfer data from the registers by using instructions. For example, the instruction MOV B, C transfers data from register C to register B.

ACCUMULATOR :-

The accumulator is a 8-bit register that is part of the Arithmetic Logic Unit. This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

FLAGS :-

The ALU includes five-flops that are set or reset according to data conditions in the accumulator and other registers. It is used to perform the testing for data conditions.

For example, after an addition of two numbers, if the sum in the accumulator is larger than eight bits, the flip-flop is used to indicate a carry, called the carry flag is set to one. When an arithmetic operation results in zero, the flip-flop called the zero-flag (Z) is set to one. The 8085 has five flags to indicate five different types of data conditions. They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxillary Carry (AC) flags. The most commonly used flags are Zero and Carry.

PROGRAM COUNTER :-

This 16-bit register deals with the fourth operation, sequencing the execution of instructions. This register is a memory pointer. Memory location have 16-bit address, and that is why this is a 16-bit register. The microprocessor uses the register to sequence the execution of instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched.

STACK POINTER :-

The stack pointer is also a 16-bit register used as a memory pointer, initially, it will be called the stack pointer register to emphasize that it is a register. It points to a memory location in R/W memory, called the stack. The begining of the stack is defined by loading a 16-bit address in the stack pointer.

3.3 PERIPHERAL OR EXTERNALLY INITIATED OPERATIONS :-

External devices (or Signals) can initiate the following operations;

RESET:- When the reset is activated, all internal operations are suspended and the program counter is cleared. Now the program execution can again bégin at the zero memory address.

INTERRUPT :- The microprocessor can be interrupted from the normal execution of instructions and as fed to execute some other instructions called serive routine. The microprocessor resumes its operation after completing the serive routine.

READY :- The 8085 has a pin called READY. If the signal at this READY pin is low, the microprocessor enters into a wait state.

HOLD:- When the HOLD pins is activated by an external signal, the microprocessor relinquishes control of buses and allows the external peripheral to use them.

MEMORY :- Memory is an essential component of a microcomputer system. It stores binary instructions and data for the microprocessor. The 8085 microcomputer has two types of memory R/W M (Read / Write Memory) and EPROM (Erasable Progammable Read Only Memory).

The R/W memory is made of registers, and each register has a group of flip-flops that stores bit of information.

The second type of memory, the ROM, stores information permanently in the form of diodes; a group of diodes can be viewed as a register. The MPU can only Read information from the ROM; it cannot write into this memory.

8085 employes EPROM in which the information stored in semipermanent. All the information can be erased by exposing the memory to ultraviolet light through a quartz window installed on the chip.

[MICRO-COMPUTER WITH ADD-ON UTILITIES

INPUT / OUTPUT :-

The remaining components of the microcomputer system are Input / Output devices. The MPU communications with the "Outside World" through such devices. The MPU accepts binary data as input from devices such as keyboard or floppy diskettes and sends data to output devices such as LED's or printers. The two methods by which the MPU identifies and communications with the I/O devices are peripheral I/O and Memory-Mapped I/O.

In peripheral to direct I/O the MPU uses eight address lines to send the address of an I/O device. And in case of Memory Mapped I/O the MPU uses sixteen address lines to identify the I/O device.

3.4 RANDOM ACCESS MEMORY:-

Read / Write Memory is popularly known as Random Access Memory. This memory is volatile, meaning that when the power is turned off, all the contents are destroyed. Two types of R/W memories are available, static and dynamic.

Static memory is made up of flip-flops and stores a bit as a voltage. Dynamic memory is made up of MOS transistor gates, and it stores a bit as a charge.

The eight address lines A7-A0 are connected directly to the address lines on the memory chip to identify 256 memory locations. The address lines A15 to A8 are used to select the memory chip through a 3 to 8 decoder and logic gates. Identifying the memory map is a two step process. The first step is to recognize the logic levels required on the address lines A15-A8 to select the memory chip. The second step is to examine the possible logic level combinations that can be assumed by the address lines A7-A0.

When R/W M chip is selected, the logic levels on the address lines A15-A8 should be as follows.

A15	A14	A13	A12	A11	A10	A9	A8	
0	0	0	0	0	1	1	1	= 07h

The memory chip has eight address lines that can assume 256 different combinations from 00h to FFh. Therefore, the memory map of the chip ranges from 0700h to 07FFh.

To read from and write into this memory, one control signal $\overline{MEMW}$ is necessary,. When the $\overline{MEMW}$ is high, data can be read just by selecting the chip. The 8085 places the address of the memory location into which it intends to write on the address bus at T1, and causes the IO/$\overline{M}$ signal to go low at the same time. At period T2, it places the data on the data bus and sends the $\overline{WR}$ signal. During T2 and T3, the memory location is identified and the data are written into the location. The memory write cycle is in many ways similar to memory read cycle.

3.5 CIRCUIT DESCRIPTION :-

The 8085 IC is heart of the circuit. It is the LSI microprocessor chip, and needs only single +5V supply for its working. It has a built-in timing oscillator and works by connecting a crystal between its pin Nos 1 and 2. The frequency upto which it can work is generally 6 MHz, but we are using a 4 MHz crystal (or even an inexpensive (TV) 3.7MHz crystal used for clock ICs). Even though the system, in this way runs at a lesser speed in one sense, we can use slower speed memory ICs, like 6116 RAM and the 2716 EPROM.

The 40 pins of the 8085 are for address lines, data lines, serial input and output, interrupt pins, Hold and Hold Acknowledge. Resetting input and output as well as status signals and control signals for accessing the memory ICs and Input/Output ports.

The lines AD0 to AD7 carry both address and data information together on a time sharing basis. The moment during which address information is present on the lines is synchronously given by the pulse coming from pin No.30 - the Address Latch Enable pin (shown as ALE in circuit diagram). So by latching this information at this instant on an 8-bit latch consisting of 8-D flip-flops the address information is continously available on the eight outputs of the flip-flops. THe 8212 IC is used for this purpose. The inputs to this are the lines AD0 to AD7. The latched address information A0 to A7 comes out as eight lines from the 8212. The AD0 to AD7 are now useful as data lines D0 to D7 which go to the data bus.

The address lines A8 to A15 are coming continously from the pins 21,22,23,24,25,26,27 and 28. We are not going to use all the address lines but confine to only 16K of memory for the basic kit and its possible expansion. (A 1116 address lines can deal with 64 K of memory). So the address pins A15 to A14 are not used.

The main board also contains the essential EPROMs and RAM, each of 2K. The EPROM is a 2716, and it must have been pre-programmed with data representing the monitor program. The monitor program must be programmed into a fresh 2716 and then only inserted into the socket of the board. An EPROM programmer should be available for this purpose. The program listing which represents the monitor software is given later, and this must be programmed into the first 512 locations of its memory (i.e., 0000 to 01FF). The remaining locations are not used, and need not be programmed with any data; they have FF in the unprogrammed state. They can be used for other utility programs subsequently.

The purpose of the monitor program is to enable the keyboard to work. The data and programs are loaded using the program only.

The control output pins of the 8085 are IO/$\overline{M}$ (pin34), $\overline{RD}$ (pin32), $\overline{WR}$ (pin31) and other control output pins S0 and S1 (pins 29 and 33 respectively) are not used in the kit. The IO/$\overline{M}$ pin tells whether at any moment, the data refers to a port (ie input or output) or to a memory location. If this pin voltage is high, it is the former, if it is low, then it means the latter. The $\overline{RD}$ and $\overline{WR}$ signals are low while some data is being read or is output to either a port or a memory location.

These three signals are seperated for convenience into the following four signals.

$\overline{IO/R}$ - the signal which goes low for the moment any port is being addressed for a "reading from port" operation.

$\overline{IO/W}$ - the signal which goes low for the moment any port is being addressed for outputting a data word from the microprocessor's accumulator.

$\overline{MR}$ - the signal which goes low for the moment any memory location is being addressed for reading the data stored in the memory location, and

$\overline{\text{MW}}$ - the signal which goes low for the moment any memory location (8-bit) is being addressed for storing the data from the microprocessor's internal register into that location.

For separating the above signals from the I/OM, RD and $\overline{\text{WR}}$ signals, the two ICs 7402 and 7404 are used.

The 74LS138 is used for selecting the EPROM or RAM, depending on the address. We assign the address coding to these in the following way. The lines A0 to A7 represent 256 words. A8 and A9 allow upto 1K; A10 gives upto 2K.

A12	A11	Comments
0	0	First 2K
0	1	Second 2K
1	0	Third 2K
1	1	Fourth 2K

The additional line A13 can then be used to select two such sets of 8K memories. Here A13 is used to select four more 2K memory groups. Thus totally eight 2K memory groups are divided by IC 74LS138, whose eight outputs Y0 to Y7 are used, one for each 2K memory chip. The chip can be either a 2716 EPROM or a 6116 (2K) RAM. Provision for three 2716 EPROM's is made on the PCB.

2716 I	First 2K	(Y0)
2716 II	Second 2K	(Y1)
2716 III	Third 2K	(Y2)

The rest of the outputs, Y3-Y7 brought out to the edge connector.

So, the address decoding is done with the 3-line to 8-line decoder 74LS138. The three lines A11, A12 and A13 are seperated into eight select - outputs, each connectable to one memory chip. There are three chip - enable input pins also in this 71LS138 IC. These three are connected to A15, A14 and IO/$\overline{M}$ signals from the 8085. The 74LS138 is enabled only for memory access signal (not for I/O) and for A15 low, A14 low.

Note that C3 gets $\overline{A14}$, $\overline{CS2}$ gets A15 and hence the 74LS138 is enabled if both A14 and A15 are low.

A15	A14	A13	A12	A11	A10	A9	A8	Address page
0	0	0	0	0	X	X	X	Y0 00 to 07
0	0	0	0	1	X	X	X	Y1 08 to 0F
0	0	0	1	0	X	X	X	Y2 10 to 17
0	0	1	1	1	X	X	X	Y7 38 to 3F

Thus the Y0 pin selects EPROM 1 at address range 0000 to 07FFF; Y1 selects EPROM 2 at address ranging from 0800 to 0FFF; Y2 pin selects the 6116 RAM chip at address range 1000 to 17FF. Y3 to Y7 are brought out to the edge connector. The other pins of the 8085 which are brought out via the edge connector are:

TRAP, RST 5.5, RST 6.5, RST 7.5 (interrupt pins) and Reset out pin.

The four interrupt pins are to be normally kept low (ie., '0' level). When they are made high ('1' level), any program that is running in the microprocessor is interrupted. Then the program commences from the locations 000 044 (octal add) or 000 055 or 000 065 or 000 075 respectively. These four pins are brought via the edge connector and they must be connected to the four switches provided on the keyboard. When any of one of these four swithces is pressed, the program goes to the concerned location and begins executing the program stored from that location onwards. These locations 000 044 etc, there are written already in the 2716 only JUMP instructions which cause the program to jump to the following address:

023	044	023	054	023	064	023	074	(octal words)
13	24	13	2C	13	34	13	3C	(in hex)

We can write any needed program starting from these RAM locations, and after attending to this interrupted job, return to the main program by writing a return instruction at the end of the interrupt service routine. The interrupt pins are very useful. These are called "Hardware Interrupts".

The Reset output pin proves useful if, later any others 8085 family of ICs are added. AT present, it is not used. The "Int" and "Hold" pins of the 8085 are permanently kept low and these pins are not brought out.

The "Int pin" makes provision for the additional interrupts of a different mode. Since we already, have four interrupts, this Int pin is not brought out. Also, using this Int pin for providing interrupt programs is not quite easy.

The Hold pin, if kept high will keep the address and data buses of the 8085 in the high impedence or floating state. No data or address information can go out or get in. So, the bus lines are freed from the processor. These lines cam then be used for externally connected high or low signals so that the memory ICs connected to the bus lines can be accessed for storage or read out of data. This is called Direct Memory Accessing. We do not require this facility and so the Hold pin is keyt low by a jumper to ground and is not brought out of the board either.

The SID and SOD pins are used for inputting a 1 bit data into the 8085 and outputting a 1-bit data from the 8085. These lines are vey useful. Normally, a data word can get into or come out of a microprocessor only via an input or an output port. But where only a single - bit data is to be input or output, the SID and SOD lines are used, in conjunction with RIM and SIM instructions of the 8085 respectively.

CHAPTER IV

EPROM PROGRAMMING BOARD

4.1 INTRODUCTION :-

The 2716 (or Texas instruments TMS2516) is a third generation Erasable and Programmable Read Only Memory chip. It has 16,384 bits of programming sites on the metal oxide FET's integrated into it. It has a 24 pin IC, approximately 6 cm by 2.5 cm in size.On the centre of this IC, at the top, there is a small 0.7cm X 0.7cm quartz glass window. Through this window, ultraviolet can be shone onto the chip to erase completly all the bits.

The byte access time of the normal 2716 chip is 450 ns, and a faster 350 ns is also available. Since the chip can store 2K bytes of information, it is normally more. The 2716 is simple to program, it does not need high voltage pulses or multiple power supplies for programming. It is possible to program the chip easily on - borad the system, without the need for a special EPROM Programming machine.

The method of programming a 2716 is as follows:

1. Give the address information to the address pins A0 to A10.

2. Give the data bytes onto the data pins after making the chip select pin high.

3. Apply 25 V to pin No.21

4. Apply a single 50-millisecond pulse to the programming pin. It will be high going pulse for this duration.

5. Disconnect the 25 V supply from pin 21, and connect 5 V supply to it.

6. Remove the data input.

7. Make the chip select pin low now, and you get the stored data out.

4.2 PROGRAMMING, USING THE 8085 KIT

Data to be programmed and can be stored on the RAM of the kit and transferred to the EPROM sequentially. The steps 1 to 4 mentioned earlier will have to be repeated for all the bytes to be programmed. And because of the 50 millisec pulse required for each byte, a total time of a few minutes only is required. It is advisable to program the EPROM 2716 page by page (A page here means one block of 256 bytes). Note that the address lines of the 2716 go from A0 to A10. Of these the lines A8, A9 & A10 are high order address lines. These three can vary from 000 to 111, ie., totally there are eight pages starting from 000 page and ending with 111 page. The remaining address lines A0 to A7 give 256 locations for each such page. We may choose the 0th page (page 000) first, program all its 256 locations, then choose the first page (page 001) and program its 256 bytes and so on. This is convenient, even though all the 2K bytes could be programmed together. By programming page data (256 bytes only) into the RAM for programming the particular page.

It is also advisable to program from different locations of RAM running to various pages or to program few bytes alone in a EPROM. The above is achieved only through software developments.

In order to program, the data should be given to the data pins and the data should stay there on these pins during the period of programming. So an output port which can latch and keep the data is required. But for reading the EPROM after programming, the data pins of the IC act as output points, and they should be needed from the microprocessor via an input port. Thus the data pins should be both on an output port and also to an input port.

For situations like this, the device called Programmable Input Output port (also called Peripheral Interface Adapter - PIA) is a useful one. The Intel 8255 also available from other sources, is therefore used here for the EPROM programming board which we are describing here. This 8255 has actually three ports - A, B and C. We call them simply "Ports" because each one can be configured to be an input or an output port at any particular time. Thus by making the ports A as input, we can read the EPROM, and by making it as the output port, we can program the EPROM without changing any connections.

4.3 CIRCUIT DESCRIPTION

In our EPROM Programmer we have two sockets for programming two EPROMS simultaneously. We use live drivers 741s245 and 741s244 for data lines and address lines so that the current capabality for programming two chips are achieved. 741s245 is a bi-directional data bus driver and 741s244 is a uni-directional address line driver. The directional select of 741s245 pin 1 is shorted with PCI (Pin 15) chip select signal of 8255. During programming the EPROM PCI is high which makes 741s245 to pass data from pin 2 to 18 and respectively for all pins. Thus data from 8255 is output to EPROM chip. When the EPROM make reading then PCI is low and data transfered from EPROM to 8255.

Port A is assigned for data lines

Port B is assigned for EPROM's low address lines

Port C is assigned for EPROM's page address

While programming Vpp is connected to a well stabilized 25v power supply with current capabality of 500 ma. The power supply should be very accurate as EPROM chip may get damaged due to over voltages even by 0.5 to 1volt. Reset pin of 8255 is connected to reset pin of 8085. Whenever 8085 is active low therefore it is passed through an inverter. The chip select pin is activated low only when A1 and A7 are high. When 8255 is I/O mapped then the address bit have to be greater than CO as

CO selects 8255 and its A port register

C1 selects B port register

C2 selects C port register

C3 selects control register

Thus OUT CO means data is outout through A port to Eprom.

4.4 SOFTWARE DESCRIPTION

Usually EPROMs are programmed page wise. In our kit it is possible to program the EPROM location wise. The start of location of programming EPROM can be any thing. The data is fetched from RAM location or from EPROMs in location No 0000 to 07FF or 0800 to 1FFF or 2000 to 27FF. Any number of bytes with in 2K can be programmed. In the programming routine initially some locations of RAM are utilised for storing initial address of EPROM which has to be programmed , starting and ending address of location from where data is fetched and also contains number of bytes to be programmed. Please refer key-notes for more information.

Initially data and address are placed in respective lines and status or low order portC line i placed with programming pulse and chip select signal for programming each location it takes 50 ms. Then a time delay for 50 ms is called . After 50 ms the programming pulse is reset by output on port C with previous control word.

While reading the EPROM remember that Vpp is connected to VCC and PCI is made low in software. Now datas are passed to 8095 from EPROM. This display shows the address is moved to HL pair. The keyboard subroutine is called which prints the contents of H,L & C registers. When you press 'C' key alone HL pair is incremented displaying next data when pressing 'D' key alone HL pair is decremented displaying previous data.

In verification program the contents of programmed EPROM is verified with RAM and EPROM from where the copy is made. If there is any wrong in data then the program displays 'ERROR' else it compares whole thing and displays 'CORRECT', if it is. For this also certain RAM locations are used for selecting address i/p's as in programming routine.

Blank check is the additional program which verifies whether the EPROM is programmed or not. Only EPROM's whose contents are FF can be programmed otherwise error in data programming occures. When such thing happens entire EPROM has to be exposed to Ultera voilet light of 2500 A which is waste of time . To avoid such situations are EPROM's are subjected to blank check before they are programmed. If there is any mismatch the program displays 'ERROR'.

4.5 THE 8255A PROGRAMMABLE PERIPHERAL INTERFACE

The 8255A is a widely used, programmable, paralled I/O device. It can be programmed to transfer data under variour conditions, from simple I/O to interrupt I/O. It is flexible, versatile, and economical (when multiple I/O ports are required), but some what complex. It is an important general purpose I/O device that can be used with almost any microprocessor.

The 8255A has 24 I/O pins that can be grouped primarily in two 8-bit parallel ports A and B, with the remaining eight bits as port C. The eight bits of port C can be used as induvidual bits or be grouped in two 4 bit ports as C upper (Cu) and C lower (Cl). This device is like three 8212s with many more additional features. The functions of these ports are defined by writing a control word in the control register.

The functions of the 8255A, classified according to two modes is as shown below. The Bit Set / Reset (BSR) mode and the I/O mode.

The BSR mode is used to set or reset the bits in port C. The I/O mode is further divided into three modes. Mode 0, Mode 1 and Mode 2. In Mode 0, all the ports function as simple I/O ports. Model 1 is a handshake mode where by ports A and/or B use bits from port C as handshake signals. In the handshake mode, two type of I/O data transfer can be implemented. Status check and interrupt. In Mode 2, Port A can be setup for bidirectional data transfer using handshake signals from port C, and Port B can be setup either in Mode 0 or Mode 1. In our case we use 8255 in Mode 0 operation alone.

BLOCK DIAGRAM OF 8255A :-

The block diagram in Fig shows two 8 bit ports (A&B), two 4-bit ports (Cu and Cl), the data bus buffer and control logic. Figure shows a simplified but expanded version of the internal structure, including a control register. This block diagram includes all the elements of a programmable device. Port C performs functions similar to that of the status register in addition to providing handshake signals.

CONTROL LOGIC :-

The control section has 6 lines. Their functions and connections are as follows:

$\overline{\mathrm{RD}}$ (READ) : This control signal enables the read operation. When the signal is low, the MPU reads data from a selected I/O port of the 8255A.

$\overline{\mathrm{WR}}$ (WRITE) : This control signal enables the write operation. When the signal goes low, the MPU writes data to a selected I/O port or the control register.

RESET (Reset) : This is an active high signal, it clears the control register and sets all ports in the input mode.

$\overline{\mathrm{CS}}$, A0 and A1 : These are device select signals. $\overline{\mathrm{CS}}$ is connected to a decoded address and A0 and A1 are generally connected to MPU address lines A0 and A1 respectively.

The $\overline{CS}$ signal is the master chip select, and A0 and A1 specify one of the I/O ports or the control register as given below:

$\overline{CS}$	A0	A1	SELECTED
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	8255A is not selected

As an example, the port addresses are determined by the $\overline{CS}$, A0 and A1 lines. The $\overline{CS}$ line goes low when A7=1 and A6 through A2 are at logic 0. When these signals are combined with A0 and A1, the port addresses range from 80H to 83H, as shown in Figure.

CONTROL WORD :-

Figure shows a register called control register. The contents of the register, called the control word, specify an I/O function for each port. This register can be accessed to write a control word when A0 and A1 are at logic 1, as mentioned previously. The register is not accessible for a read operation.

Bit D7 of the control register specifies either the I/O function on the Bit Set/Reset function as classified. If bit D7=1, bits D6-D0 determine I/O functions in various modes, as shown in figure. If bit D7=0, port C operates in Bit Set/Reset mode. The BSR control word does not affect the functions of port A, B and C.

To communicate with peripherals through the 8255A, three steps are necessary.

1. Determine the addresses of ports A, B and C and of the control register according to the chip select logic and address line A0 and A1.

2. Write a control word in the control register.

3. Write I/O instructions to communicate with peripherals through ports A, B and C.

MODE 0, SIMPLE INPUT OR OUTPUT :-

In this mode port A and B can be viewed as equivalent to two 8212's and port C as equivalent of two 4 bit 8212's. Each port (or half port in case of C) can be programmed to function as simply an input port or an output port. The input/output features in mode 0 are as follows.

1. Outputs are latched.

2. Inputs are not latched.

3. Ports do not have handshake or interrupt capability.

4.6 ERASING THE EPROM:-

The EPROM and PROM are read only user programmable memories that can be reprogrammed a number of times. If a mistake occurs in the data programmed the whole EPROM should be erased and reprogrammed. There are two types:

1. The UV-Erasable PROM and
2. Electrically erasable PROM.

A typical EPROM is erased by exposing it to hand (high frequency) ultra violet light for five to ten minutes. Thus returning the contents of all memory cells to zero by discharging them.

An EPROM package has a characteristic aspect. The seal on top of the chip is not opaque. It is a quartz window that allows the ultra-violet light through. Once zeroed, the EPROM can be programmed with a special PROM programmer selected locations within the EPROM can then be programmed and within a few minutes a bit pattern can be installed in the EPROM. Then the component can be inserted in the application board. If errors are detected or changes are desired, the EPROM can be plugged and reprogrammed within minutes. This process can be repeated many times.

Several technologies are used to implement EPROM's. The " floating - gate " accumulated is one of the best used. A charge is accumulated in the silicon gate "floating" above the silicon substrate but isolated from it by a silicon - dioxide layer. The charge in induced in the silicon gate by trains on pulses. Once programmed, an EPROM is expected to retain its charge for 10 years with only 30% 100s of charge. Erasure of the charge is a accomplished with hard ultra-voilet light. The photons hitting the floating silicon gate displace electrons from shallow energy levels and cause them to migrate to the silicon substrate where their charge is neutralised, the corresponding bit reverts to zero.

EPROM's are also available that are erasable by electricity.

CHAPTER V

ADC & DAC INTERFACE

5.1 INTRODUCTION :

Digital to Analog and Analog to Digital conversion form two very important aspects of digital data processing. Digital to analog conversion involves translation of digital information in to equivalent analog information. As an example, the O/P of a digital system might be changed to analog form for the purpose of pen recorder. Similiarly, an analog signal might be received for servo motors which drives the cursor arms of a plotter. In this respect, a D/A converter (DAC) is some times considered a decoding device.

The process of changing an analog signal to an equivalent digital signal is accomplished by the use of an A/D converter. For example, an A/D converter is used to change the analog output signals from a transducer into equivalent digital signals. These signals would be in a form suitable for entry into a digital system. An A/D converter is often referred to as an encoding device since it is used to encode signals for entry into a digital system.

[MICRO-COMPUTER WITH ADD-ON UTILITIES

5.2 METHODS OF ADC CONVERSION

There are many methods of A/D conversion they are

1. Simultaneous conversion or Flash converter,which is faster of all ADCs. It consists of many conparators with increasing order of reference voltages applied to other. The O/P of the comparators depends on how large the analog signal is from reference voltage. Example,:

Simultaneously A/D conversion table:

				comparator o/ps	
Input voltage			C1	C2	C3
+0	TO	+ V/4	L	L	L
+V/4	TO	+ V/2	H	L	L
+V/2	TO	+3V/3	H	H	H
+3V/4	TO	+ V	H	H	H

Number of comparators required will be 2 n-1 n-no

2. COUNTER METHOD : By this method higher resolution could be obtained using minimum number of comparators as it is costlier in simultaneous conversions. This is possible by variable reference voltages. But the time required for this type of conversion is pretty large and depends on number of bits required.

3. SLOPE TYPE A/D CONVERTER : Here a ramp is generated and compared with analog signal by this time a clockof certain frequency is passed to the O/P logic which counts the clock pulses. Here the circuit is very complex.

4. SUCCESSIVE APPROXIMATION : This is the simplest and most accurate and speedy converter, but less speed than Flash converter.

The figure shows a Successive approximation type of A/D converter. The heart of the circuit is an 8 bit Successive approximation resistor (SAR), whose O/P is applied to an 8 bit D/A converter. The analog O/P Va of the D/A converter is then compared to an analog signal Vin by the comparator. The O/P of the comparator is a serial data input to the SAR. The SAR then adjust its digital output (8bits) until it is equivalent to analog input Vin. The 8 bit latch at the end of conversion holds on to the resultant digital data output. The circuit works as follows. At the start of a conversion cycle, the SAR is reset by holding the start(S) signal high on the first clock pulse LOW - TO - HIGH transition, the most siginificant output bit Q7, which is compared with the analog input Vin, if the comparator output is low the digital Analog output isHIGH, the D/A output Vin and the SAR will keep the MSB Q7 set. In any case, on the next clock pulse LOW - TO - HIGH transition the SAR will set the next MSB Q6. Depending on the output of the comparator, the SAR will then either keep or reset the bit Q6. This process is continued until the SAR tries all the bits. As soon as the LSB Qo is tried, The SAR forces the conversion complete (CC) signal HIGH to indicate that the parallel output lines contain valid data. The CC signal in turn enables the latch and digital data appears at the output of the latch. Digital data are also available serially as the SAR determines each bit to cycle the converter or continuously the CC signal may be connected to start conversion input. The advantage of the successive approximation A/D converter is in its high speed and excellent resolution. for example, 8 bit successive approximation A/D converter (0804) requires only 8 clock pulse.

5.3 ADC 0804

ADC 0804 is a CMOS 8-bit successive approximation A/D convertor which use a modified potentiometers ladders, and are designed to operate with 8085 control bus via a 3-state outputs. These converters appear to the procesor as memory locationn or I/O ports and hence no interfacing logic is required.

The differential analog voltage input has good common mode rejection and permits offsetting the analog-Zero-Input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8-bits of resolution.

5.4 FUNCTIONAL DESCRIPTION :

A functional diagram of the ADC 0804 of A/D converter is shown in fig. The device operates on the successive approximation principle. Analog switches are closed sequentially by successive approximation logic until the analog differential input voltage Vin(+) - Vin(-) matches a voltage derived from a tapped resistor string across the reference voltage. The most significant bit is tested first and after 8 comparisions (64 clock cycles), an 8 bit binary code (1111 1111 = full-scale) is transferred to an Output latch.

The normal operation proceeds as follows. On the high-to-low transition of the $\overline{WR}$ input, the internal SAR latches and the shift-register stages are reset, and the $\overline{INTR}$ output will be set high. As long as the $\overline{CS}$ input and $\overline{WR}$ input remain low, the A/D will remain in a reset state. Conversion will start from 1 to 8 clock periods after at least one of these inputs makes a low-to-high transition. After the requiste number of clock pulses to complete the conversion, the $\overline{INTR}$ pin will make a high-to-low transition. This can be used to interrupt a processor, or otherwise signal the availability of a new conversion. A $\overline{RD}$ operation (with $\overline{CS}$ low) will clear the $\overline{INTR}$ line high again. The device may be operated in the free-running mode by connecting $\overline{INTR}$ to the $\overline{WR}$ input with $\overline{CS}$ = 0. To ensure start-up under all posible conditions, an external $\overline{WR}$ pulse is required during the first power-up cycle. A conversion-in-process can be interrupted by using a second start command.

DIGITAL DETAILS :

The converter is started by having $\overline{CS}$ and $\overline{WR}$ simultaneously low. This sets the start flip-flop (F/F) and the resulting "1" level resets the 8 bit shift register, resets the interrupt (INTR) F/F and inputs a "1" to the Q output of DFF1, which is at the input end of the 8 bit shift register. Internal clock signals then transfer this "1" to the Q output of DFF1. The AND gate, G1, combines this "1" output with a clock signal to provide a reset signal to the start F/F. If the set signal is no longer present (either $\overline{CS}$ or $\overline{WR}$ is a "1"), the start F/F is reset and the 8 bit shift register then can have the "1" clocked in, which starts the conversion process. if the set signal were to still be present, this reset pulse would have no effect (both outputs of the start F/F would be at a "1" level) and the 8 bit shift register would continued to be held in the reset mode. This allows for asynchronous or wide $\overline{CS}$ and $\overline{WR}$ signals.

After the "1" is clocked through the 8 bit shift register (which completes the SAR operation) it appears as the input to DFF2. As soon as this "1" is output from the shift register, the AND gate, G2, causes the new digital word to transfer to the 3 state output latches. When DFF2 is subsequently clocked, the $\overline{Q}$ output makes a high-to-low transition which causes the INTR F/F to set. An inverting buffer then supplies the $\overline{INTR}$ output signal.

When data is to be read, the combination of both $\overline{CS}$ amd $\overline{WR}$ being low will cause the INTR F/F to be reset and the 3 state output latches will be enabled to proveide the 8 bit digital outputs.

DIGITAL CONTROL INPUTS :

The digital control inputs ($\overline{CS}$, $\overline{RD}$ and $\overline{WR}$) meet standard TTL logic voltage levels. These signals are essentially equivalent to the standard A/D start and output enable control signals, and are active low to allow an easy interface to microprocessor control busses. For non-microprocessor based applications, The $\overline{CS}$ input (pin 1) can be grounded and the standard A/D start function obtained by an active low pulse at the $\overline{WR}$ input (pin 3). The output enable function is acheived by an active low pulse at the $\overline{RD}$ input (pin 2).

ANALOG OPERATION :

The analog comparisions are performed by a capacitive charge summing circuit. Three capacitors (with precise ratioed values) shares a common node with the input to an auto-zero comparator. The input capacitor is switched between Vin(+) and Vin(-), while two ratioed reference capacitors are switched between taps on the reference voltage divider string. The net charge corresponds to the weighted difference between the input and the current total value set by the successive approximation register. A correction is made to offset the comparision by 1/2 LSB.

ANALOG DIFFERENTIAL VOLTAGE INPUTS AND COMMON-MODE REJECTION :

This A/D gains considerable applications flexibility from the analog differential voltage input. The Vin(-) input (pin 7) can be used to automatically subtract & fixed voltage value from the input reading (tare correction). This is also useful in 4mA - 20mA current loop conversion. In addition, common mode noise can be reduced by use of the differential input.

The time intervel between sampling Vin(+) and Vin(-) is 41/2 clock periods. The maximum error voltage due to this slight time difference between the input voltage samples is given by:

$$Ve(max) = (Vp)(2-fcm)(4.5/fCLK)$$

where:

Ve is the error voltage due to sampling delay

Vp is the peak value of the common mode voltage

fcm is the common mode frequency

For example, with a 60 Hz common-mode frequency. fcm, and a 640 kHz A/D clock, fCLK, keeping this error to 1/4 LSB (app 5mV) would allow a common-mode voltage, Vp, given by:

$$Vp = \frac{[\ Ve(MAX)\ (fCLK)]}{(2\ \ fcm)\ (4.5)}$$

or

$$Vp = \frac{(5x10\ 3)\ (640x10\ 3)}{(6.28)\ (60)\ (4.5)} == 1.9V$$

The allowed range of analog input voltage usually places more severe restrictions on input common-mode voltage levels than this.

[MICRO-COMPUTER WITH ADD-ON UTILITIES

An analog input volage with a reduced span and a relatively large zero offset can be easily handled by making use of the differential input.

ANALOG INPUT CURRENT :

The internal switching action causes displacement currents to flow at the analog inputs. The voltage on the on-chip capacitance to ground is switched through the analog differential input voltage, resulting in proportional currents entering the VIN(+) input and leaving the VIN(-) input. These current trancient occures at the leading edge of the internal clocks. they rapidly decay and do not inherently cause errors as the on-chip comparator is strobed at the end of the clock period.

INPUT BYPASS CAPACITORS :

Bypass capacitors at the input will average these charges and cause a DC current flow through the output resistances of the analog signal sources. This charge pumping action is worse for continuous conversion with the VIN(+) input voltage at full-scale. For a 640kHz clock frequency with the VIN(+) input at 5V, this DC current is at a maximum of approximately 5microAmps. Therefore, bypass capacitors should not be used at the analog inputs or the VREF/2 pin for high resistance sources (1kohm). If input bypass capacitors are necessary for noise filtering and high source resistance is desirable to minimise capacitor size, the effects of the voltage drop accross this input resistance, due to the average value of the input current, can be compensated by a full-scale adjustment while the given source resistor and input bypass capacitor are both in place. This is possible because the average value of the input current is a precise linear function of the differential input voltage at a constant conversion rate.

INPUT SOURCE RESISTANCE :

Large values of source resistance where an input bypass capacitor is not used, will not cause errors since the input current settle on prior to the comparision time. If a low-pass filter is required in the system, use a low-value series resistor (1 Kohm) for a passive RC section or add an op amp RC active low-pass filter. For low-source-resistance applications, (1 Kohm),and a 0.1 mfd bypass capacitor at the inputs will minimize EMI due to the series lead induction of a long wire. A 100 ohm series resistor can be used to isolate this capacitor (both the R and C are placed outside the feedback loop) from the output of an op amp, if used.

STRAY PICKUP :

The leads to the analog inputs (pin6 and 7) should be kept as short as possible to minimize stray signal pickup (EMI). Both EMI and undesired digital-clock coupling to these inputs can cause system errors. The source resistance for these input should, in general, be kept below 5Kohm. larger values of source resistance can cause undesired signal pickup. Input bypass capacitors, placed from the analog input ground, will eliminate this pickup but can create analog scale errors as these capacitor average the trancient input switching currents of the A/D . This scale error depends on both a large source resistance and use of an input bypass capacitor. This error can be compensated by a full-scale adjustment of the A/D with the source resistance and input bypass capacitor in place , and the desired conversion rate.

REFERENCE VOLTAGE SPAN ADJUST :

For maximum application flexibility, these A/Ds have brrn designed to accomodate a 5V, 2.5V or an adjust voltage reference. This has been acheived in the design of the IC as shown in fig.

Notice that the reference voltage for the IC is either 1/2 of the voltage which is applied to the V+ supply pin, or is equal to the voltage which is externally forced at the VREF/2 pin. This allows for a pseudo-ratiomatric voltage reference using, for the V+ supply, a 5V reference voltage. Alternatively, a voltage less than 2.5V can be applied to the VREF/2 input. The internal gain to the VREF/2 input is 2 to allow this factor of 2 reduction in the reference voltage.

Such an adjust reference voltage can accommodate a reduced span or dynamic voltage range of the analog input voltage. If the analog input voltage were to range from 0.5V to 3.5V, instead of 0V to 5V, the span would be 3V. With 0.5V applied o the VIN(-) pin to observe the offset, the reference voltage can be made equal to 1/2 of the 3V span or 1.5V. The A/D will now will encode the VIN(+) signal from 0.5V to 3.5V with the 0.5V input corresponding to zero and the 3.5V with corresponding to full-scale. The full 8 bits of resolution are therefor applied over this reduced analog input voltage range. The requist connection are shown in fig. For expanded scale inputs, the circuits of fig. can be used.

REFERENCE ACCURACY REQUIREMENT :

The converter can be operated in a pseudo-ratiomatric mode or an absolute mode. In ratiomatric converter applications, the magnitude of the reference voltage is a factor in both the output of the source transducer and output of the A/D converter and therefore cancels out in the final digital output code. In absolute conversion applications, both the initial value and the temperature stability of the reference voltage are important accuracy factor in the operation of the A/D converter. For VREF/2 voltages of 2.5V nominal value, initial errors of +- 10 mV will cause conversion errors of +- 1 LSB due to the gain of 2 of the VREF/2 input. In reduced span applications, the initial value of the stability of thr VREF/2 input voltage become even more important. For example, if the span is reduced to 2.5V, the analog input LSB voltage is correspondingly reduced from 20mV (5V span) to 10mV and 1 LSB at the VREF/2 input becomes 5mV. As can be seen, this reduces the allowed initials tolerance of the reference voltage and requires correspondingly less absolute change with temperature variations. Note that spans smaller than 2.5V place even tighter requirements on the initial accuracy and stability of the reference source.

In general, the reference voltage will require an initial adjustment. Errors due to an improper value of reference voltage appears as full-scale errors in the A/D transfer function. IC voltage regulators may be used for reference if the ambient temperature changes are not excessive.

ZERO ERROR :

The zero of the A/D does not require adjustment. If the minimum analog input voltage value, VIN(MIN), is not grounded, a zero offset can be done. the converter can be made to output 0000 0000 digital code for this minimum input voltage by biasing the A/D VIN(-) input at this VIN(MIN) value. This utilizes the differntial mode operation of the A/D.

The zero error of the A/D converter relates to the location of the first riser of the transfer function and can be measured by grounding the VIN(-) input and applying a small magnitude poditive voltage to the VIN(+) input. Zero error is the difference between the actual DC input voltage which is necessary to just cause an output digital code transition from 0000 0000 to 0000 0001 and the ideal 1/2 LSB value (1/2 LSB = 9.8mV VREF/2 = 2.5V).

FULL - SCALE ADJUST:

The full-scale adjustment can be made by applying a differential input voltage which is 1 1/2 LSB down fromthe desired analog full-scale voltage range and then adjusting the magnitude of the VREF/2 input (pin 9) for a digital output code which is just changing from 1111 1110 to 1111 1111. When offsetting the zero and used a span-adjusted VREF/2 voltage, the full-scale adjustment is made by inputting VMIN to the VIN(+) input which is given by:

$$vin(+)FSADJ == vmax - 1.5 [(vmax - vmin)/256]$$

Where :

VMAX = the high end of the analog input range

and

VMIN = the low end (the offset zero) of the analog range. (both are ground referenced)

CLOCKING OPTION :

The clock for the A/D can be derived from an external source such as the CPU clock or an RC network can be added to provide self-clocking. The CLK IN (pin 4) makes use of a schmitt trigger.

Heavy capacitive or DC loading of the CLocK R pin should be avoided as this will disturb normal converter operation. Loads less than 50pF, such as driving up to 7 A/D converter clock inputs from a single CLK R pin of 1 converter, are allowed. For larger clock line loading, a CMOS or low power TTL buffer or PNP input logic should be used to minimize the loading on the CLK R pin (do not use a standard TTL buffer).

RESTART DURING A CONVERSION :

If the A/D is started ($\overline{CS}$ and $\overline{WR}$ go low and return high) during a conversion, the converter is reset and a new conversion is started. The output data latch is not updated if the conversion in progress is not completed. The data from the previous conversion remains in this latch.

CONTINUOUS CONVERSIONS :

In this application, the $\overline{CS}$ input is grounded and the $\overline{WR}$ input is tied to the $\overline{INTR}$ output. This $\overline{WR}$ and $\overline{INTR}$ node should be momentairly forced to logic low following a power-up cycle to insure circuit operation.

DRIVING THE DATA BUS :

This CMOS A/D, like MOS microprocessor and memories, will require a bus driver when the total capacitance of the data bus gets larger. Other circuitary, which is tied to the data bus, will add to the total capacitive loading, even in 3 state (high-impedance mode). Backplane bussing also greatly adds to the stray capacitance of the data bus.

There are some alternatives available to the designer to handle this problem. Basically, the capacitive loading of the data bus slows down the response time, even though DC specifications are still met. For systems operating with a relatively slow CPU clock frequency, more time is available in which to establish proper logic levels on the bus and therefore higher capacitive loads can be drivwn.

Finally, if time is short and capacitive loading is high, external bus driver must be used. These can be 3-state buffers (low power Schottky is recommended, such as the 74ls240 series) or special higher-drive-current products which are designed as bus drivers. High-current bipolar bus drivers with PNP inputs are recommended.

POWER SUPPLIES :

Noise spikes on the V+ supply line can cause conversion errors as the comparator will respond to this noise. A low-inductance tantalum capacitor should be used close to the converter V+ pin, and values of 1 mfd of greater are recommended. If an unregulated voltage is available in the system, a seperate 5V voltage regulator for the converter (and another analog circuitry) will greatly reduce digital noise on