ABEKAS VIDEO SYSTEMS, INC.
3180 PRESIDENTIAL DRIVE, SUITE G
ATLANTA, GA 30340
PHONE (404) 451-0637

Abekas Video Systems, Inc.

A Carlton Company

digital disk recorder **A60/64**

A60/64

Preliminary Digital Video Interface Manual

PRELIMINARY

A60/A64 Digital Video Interface Manual

Rev 1.2 21-AUG-87
Copyright (C) 1987
Abekas Video Systems, Inc.

This manual describes the formats for both the CCIR 601 4:2:2 digital component video interface and the Small Computer Systems Interface (SCSI). It covers both 525 and 625 line versions of the A60 and A64.

The SCSI interface allows the A60/A64 Digital Disk Recorders to be treated as computer peripherals using the same interface and command set as computer mass storage devices such as disks or tape drives. The A64 SCSI interface is contained in an optional interface unit, in the A60 it is integral.

The SCSI interface also provides an alternative method of remote control since all the serial protocol commands can be issued through the SCSI port.

Some of the data formats described in this manual are also used for transfers to the A60 via the Ethernet port.

Abekas Video Systems, Inc.
101 Galveston Drive
Redwood City, CA 94063
(415) 369-5111

CONTENTS

1. Component Video Application Notes

Introduction

This section gives a brief background description of 4:2:2 Component Digital sampling. It is aimed at programmers intending to render material into the native format of Abekas Digital Disk Recorders.

Although this section does not go into any lengthy mathematical explanations no attempt is made to explain words like 'phase' or 'modulation' and 'bandwidth' any reader unfamiliar with these terms is advised to read the first few chapters of a book on TV theory.

This document is not directly concerned with the NTSC and PAL standards as these are the coding schemes used to generate the composite video signals. The digital format and sampling for both TV standards is the same, they differ only in the number of lines and the field rate. The different versions of the digital standard are referred to here as 525/60 and 625/50 being the number of lines in a frame and the field frequency.

Digital video is stored in component format in both the Abekas A64 and A60. Note however that the Abekas A62 is a Composite NTSC machine which stores the video as a single channel of composite video rather than the component machines which effectively store video as three separate channels of component video.

Abekas currently offers the following digital interfaces on their disk recorder products

	A60	A64	(A62)
CCIR 601 Digital Video	X	X	
Framestore Parallel port		X	X
Built in SCSI	X		
Optional SCSI interface		X	X
Offline Storage option	X	X	X
Ethernet TCP/IP	X		

The Offline storage option is a high speed SCSI streaming tape drive that permits transfer by tape from any computer that can

write 9 Track Magtape in the required format.

The following methods of Remote control are available
- Serial Ports : RS422 or RS232
  - Abekas Keyboard Protocol
  - Editor Protocols : Sony Ampex CMX
- Parallel Port (A64)
- SCSI
- Ethernet (A60)
- Timecode Trigger
- GPI (Contact Closure)

Relative speed of transfer

Ethernet : raw YUV frames transfer in less than 5 seconds, converting from RGB and filtering will take about 10 times longer. SCSI can achieve transfer rates of 1.1 MBytes but the data has to be in YUV format with syncs and extra line number information in the data stream. The A64 Parallel port should allow transfer rates up to 7MHz but the handshake requires custom hardware.

Ethernet Data Formats

At time of writing two data file formats will be offered, YUV and RGB. YUV is the native format of the Disk Recorder, transfers will be much faster and the data will not be changed between writing and reading.

YUV stores 16 bits per pixel, non-YUV images of greater resolution (eg 24 bit RGB samples) are converted as they are written to and read form the disk and in so doing there is an inevitable loss of information. So for example RGB data written to the A60 and then read back will not neccesarily have the same values it gets rounded to the nearest value in YUV space and filtered to limit its bandwidth.

The disk Recorders can be used for storage of Non-YUV image data, for instance temporarily buffering RGB images while compositing several layers. The only restriction is that the data passed through the SCSI port should not contain the values 00 and FF since these are reserved for syncs.

1.1 Lines Fields and Frames

The following table shows the fundamental timings for the two TV standards.

	525/60 (NTSC)	
Line Rate	15.735264 KHz	(2Fsc/455)
Lines per Field	262.5	
Field Rate	59.94 Hz	(2/525 * H)
	625/50 (PAL)	
Line Rate	15.625 KHz	(4Fsc/(1135+4/625))
Lines per Field	312.5	
Field Rate	50 Hz	(2/625 * H)

Fields and Frames

The 525 line picture is scanned as two interlaced fields of 262.5 lines each The first eight and a half lines in the field are taken up with vertical sync and the next sixteen lines are blanked (carry no video information) to allow for the vertical retrace period. This leaves 243 lines per field for the active picture. The 625 line picture has 312.5 lines per field of which 288 are active.

Spatial separation of fields

It is possible to render the same information into both fields of a frame but this effectively halves the vertical resolution of the image.

Temporal separation of fields

Beware that if two fields are frozen (eg there was some motion of the subject between them) This movement appears as an annoying frame rate flicker.

Animation Sequences are better rendered as separate fields to give a smoother motion. Indeed to take this one step further note that the top line of a field is actually sampled and displayed at a different time to the bottom one.

Although the field rates for the two standards are different, so is the vertical resolution, the reduction in perceptible flicker in the 60Hz system is traded off against number of vertical lines.

The line rates for both standards are almost the same giving a line period of 64 uS.

The different quality of NTSC video relative to PAL can be attributed to the simpler color encoding scheme of NTSC coupled with the narrower broadcast channel bandwidth.

In terms of the CCIR 601 spec the bandwidth and data rate of both systems are the same.

1.2 Color Encoding

In order to encode a composite video signal for transmission the Red Green and Blue signals from a color camera are converted into a luminance Y signal and two color difference signals R-Y and B-Y. The luminance signal is intended to retain compatibility with the earlier monochrome standard and is transmitted as an amplitude modulated signal. The two color difference signals are superimposed on the luminance signal in the form of a quadrature phase encoded sub carrier.

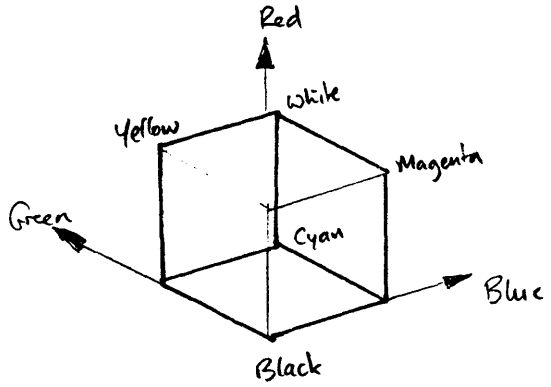
Sampling frequency

The 4:2:2 digital component video standard takes these Y, B-Y and R-Y components and digitizes them. The luminance channel is sampled at 13.5 MHz and the two color difference channels are sampled at 6.75MHz. This is the origin of the 4:2:2 ratio, which expresses the relative bandwidth ofthe YUV components, for every two luminance samples there is only one pair of color difference samples.

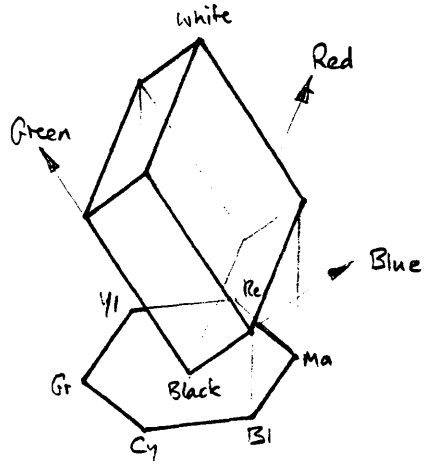
Luminance Saturation and Hue

RGB color space can be viewed as a cube standing on one of its

corners the bottom corner is black the top corner diagonally opposite it is white. The red green and blue points are at the end of the edges directly connected to the black corner and the secondary colors are at the remaining three points, Yellow between Red and green Cyan between Green and Blue and magenta between Blue and Red.



RGB Cube diagram



LSH Lozenge diagram

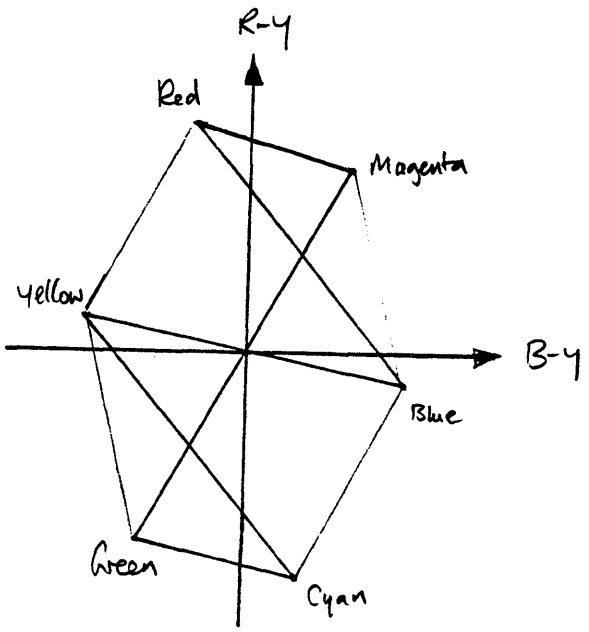
If the colored corners of the cube are translated vertically so that their perceived luminance value on the black to white scale is equal to their vertical position the cube becomes distorted into a lozenge-like shape where all of the faces are parallelograms.

The equation for the luminance component of an RGB color is based on the luminosity coefficients (or the observed relative brightness) of Red Green and Blue.

$$Y = 0.299\ R + 0.587\ G + 0.114\ B$$

Viewed from above this cube forms a hexagonal shape similar to the vector scope display of a composite video signal. The center of the hexagon is the line joining the white and black corners viewed

end on. From this view we can express hue and saturation in polar coordinates where hue is the angle and saturation is the distance from the center of the hexagon to the color.



Vector Scope Diagram

The composite video signal consists of the luminance signal with subcarrier modulated on top of it. The amplitude of the color subcarrier is equal to the saturation and the phase of the subcarrier relative to reference is the hue.

R-Y B-Y Color Difference Signals

Algebraic juggling of the luminance equation above can yield the following two equations :

$$(R-Y) = 0.701\ R - 0.587\ G - 0.114\ B$$
$$(B-Y) = -0.299\ R - 0.587\ G + 0.886\ B$$

These two differences are adjusted as follows.

```
Cb = 0.564 (B-Y)
Cr = 0.713 (R-Y)
```

The weighting normalises the color difference signals to the range -0.5 to +0.5 when the luminance signal is in the range 0.0 to 1.0 and so allows maximum use of the dynamic range available. Note the weighting coefficients are different to those used for PAL and NTSC coding where they are used to limit the maximum excursions of the modulated signal.

Two color differences and a luminance value provide sufficient information to regenerate the RGB information at the receiver. The human eye is more sensitive to changes in luminance than changes in chrominance. In order to reduce the bandwidth of the video information to a manageable level the amount of chrominance information is halved.

Once the RGB-YUV conversion has been performed once to 'round' the RGB values to the nearest YUV value it should be possible to reverse and repeat the conversion without any progressive degradation of the video. The only obstacle in this process is that when the conversion is done the components have to be filtered to limit their bandwidth to half the sampling frequency, unless this is a 'perfect' filter the information will become smeared by successive passes through the filter.

Putting all this in to one matrix we get:

```
Y  =  0.299  * r + 0.587  * g + 0.114  * b;
Cb = -0.1686 * r - 0.3311 * g + 0.4997 * b;
Cr =  0.4998 * r - 0.4185 * g - 0.0813 * b;
```

1.3 Rendering into the Framestore

Component sampling

The samples are in a sequence of four bytes: Cb Y Cr Y. A component framestore will store two bytes per pixel but when the data is converted form digital to analog the first three bytes (ie the first luminance and both the color difference samples) are supposed to be coincident. The first pixel of each pair has samples for all three components, the second has only a luminance value.

Example conversion program

An example conversion program for RGB values ranged between 0 and 255 to A60/A64 4:2:2 component video is provided at the end - this program is not optimized - it is intended to show the individual steps in the conversion.

The example also contains a rather slow example of a FIR filter which again is not particularly efficient but limits the color difference signals to the specified bandwidths.

If the bandwidth of the color difference components is not limited when viewing the the composite signal cross-color and dot-crawl effects may be observed on sharp luminance or chrominance transitions because of overlap of the luminance and chrominance components in the frequency domain of the composite signal.

Framestore dimensions

The A60/A64 holds a frame as two fields 720 pixels per line by 243 lines for 525 line systems and 288 for 625. The SCSI port only permits the framestore to be accessed a field at a time whereas the Ethernet transfers can be either field or frame.

Component devices such as the A60 and A64 do not suffer from the same color field sequence problems present in analog videotape editing. The choice of edit points with VTRs is influenced by the need to match the color subcarrier phase and this only repeats every four fields (eight for PAL).

Of course there is still a spatial difference between the two fields and although the Disk Recorders contain an Interpolator that is capable of generating a field 1 from a field 2 it the vertical resolution of the image is reduced.

Example Component Data

The following listing shows a line of the 100% Color bars which are internally generated in the A60/A64. They are provided as an example of A60/A64 component digital video format and since these are used to line up the analog circuitry in the machine they are the best reference to work from.

The repeat counts are in decimal but the pixel values are in hex. Note that there are at least 8 pixels of transition between each color.

```
     80 10 80 10 80 46 80 B4 80 EB 80 EB
35 * 80 EB 80 EB                                        (White)
     80 EB 80 EB 75 EB 82 E4 32 D8 8D D2 10 D2 92 D2
36 * 10 D2 92 D2                                        (Yellow)
     10 D2 92 D2 1F D2 85 C7 79 B3 37 A9 A6 A9 10 A9
36 * A6 A9 10 A9                                        (Cyan)
     A6 A9 10 A9 9B A9 12 A3 57 96 1D 90 36 90 22 90
36 * 36 90 22 90                                        (Green)
     36 90 22 90 45 90 35 87 9E 73 A5 6A CA 6A DE 6A
36 * CA 6A DE 6A                                        (Magenta)
     CA 6A DE 6A BF 6A E0 64 7C 57 EA 51 5A 51 F0 51
36 * 5A 51 F0 51                                        (Red)
     5A 51 F0 51 69 51 E3 47 C3 33 95 28 F0 28 6E 28
36 * F0 28 6E 28                                        (Blue)
     F0 28 6E 28 E5 28 70 22 A2 16 7B 10 80 10 80 10
36 * 80 10 80 10                                        (Black)
     80 10 80 46 80 B4 80 EB 80 EB 80 EB
35 * 80 EB 80 EB                                        (White)
     80 EB 80 EB 80 B4 80 46 80 10 80 10
```

Blanking

Vertical blanking

Vertical blanking traditionally takes up 19.5 lines of each 525 line field and for 625 however the A60 and A64 framestore contains all but 9.5 of the video lines of the field, that is 506 lines per frame in 525 systems and 606 in 625.

Horizontal Blanking

The 720 samples per line are all active but the values should ramp up from black at the ends of the line.

Analog Blanking

The specification for the active portion of a composite analog line is slightly less than the digital standard for instance digital active line is 53.3 us long when you take 720/13.5 however the analog active line in the NSTC spec is 52.7 us and in PAL is 51.95 us. This might cause 3 to 8 pixels to be cropped at either end of the line if the video is passed through an analog device that adds blanking such as a VTR, the digital line is supposed to be centered in relation to the analog line.

Safe Areas

SMPTE standard recommends a Graphics safe area with a 5% border all round and a Title safe area with a 10% border to account for the overscanning of domestic receivers.

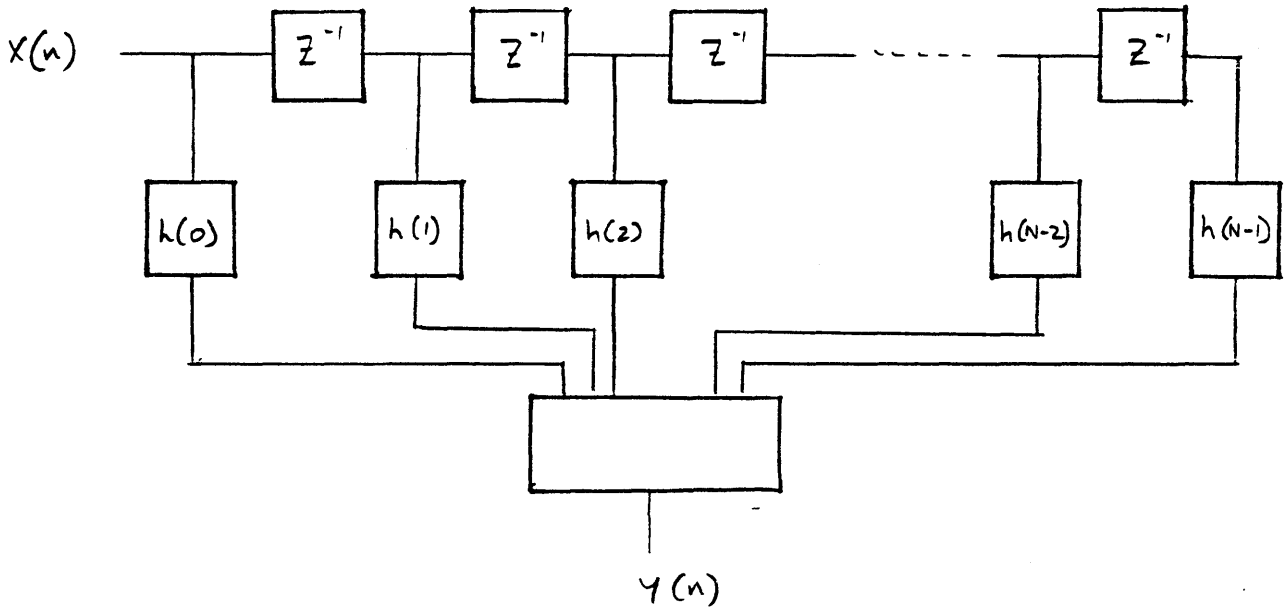
1.4 Interpolation

The A60 and A64 output hardware does contain an interpolator to allow it to generate a second field from one.

To generate an adjacent field it inevitably results in a loss of vertical resolution since the new field is spatially offset from the old one and the A60/A64 interpolates vertically to avoid an apparent picture shift.

1.5 Filtering

The simplest way to limit the bandwidth of digitally sampled data is to shift the data through a FIR (Finite Impulse Response) filter. These are also called traversal or non-recursive filters.



FIR Filter Diagram

In this diagram X(n) represents the input sample Z^{-1} are sample period delays h(N) are the filter coefficients and Y(n) is the output value equal to a the sum of the product terms.

The FIR filter If the filter has 5 taps the filtered data is not available until two more pixel values have been shifted into the filter.

The Filter registers should be filled with a known value (such as black) before the data is shifted through and in a similar way the last two pixels will have to be flushed out at the end.

Ideally the incoming data stream needs to be padded with a black border equal to half the width (aperture) of the filter, otherwise the data at the end of one line will be filtered into the start of

the next.

Filter design books such as the one mentioned in the bibliography give formulae for generating the coefficients given the number of taps and the size of the required pass band (normally expressed as a fraction of the sampling frequency).

An odd number of taps are normally chosen to give a filtered samples that are coincident with the input data.

In this application the terms either side of the center will be a mirror image.

The values of the taps are normalized such that they add up to one so that if the incoming data is flat the same value will be output.

These magic numbers are produced by the program "eqfir" found in the IEEE publication "Programs for Digital Signal Processing".

To limit 13.5 MHz luminance samples to 5.75 MHz

-0.05674 0.01883 1.07582 0.01883 -0.05674

To limit 13.5 MHz chrominance samples to 2.5 MHz (only generate alternate samples).

0.14963 0.22010 0.26054 0.22010 0.14963

1.6 Gamma Correction

The luminance signal is not linear - a cathode ray tube does not have a linear relationship between voltage applied and light output. Rather than add correction circuitry to all domestic receivers the transmitted video signal is pre-distorted. This correction function approximates to a square root of the intensity in the range 0.0 to 1.0 in fact it is equivalent to raising to the power of 1/2.2

Take the example of a white horizontal line two (frame) lines in height, there is one line in each field. If these lines are shifted up by half a line in linear terms the upper line would remain peak white and two adjacent lines in the other field have 50% luminance. This will seem to flicker because of the non linear response of the phosphor and the correct answer is to use lines of 50% raised to the power 1/2.2 or 73% luminance.

Gamma correction should be applied to the RGB values before they are converted to Y I and Q components.

1.7 Illegal Colors

Any RGB color can be encoded in YUV components but not every YUV combination is a valid RGB color, for instance YUV colors with large chrominance components and little or no luminance are outside RGB space. If YUV space is a rectangle RGB space can be viewed as the lozenge in the earlier diagram place within the rectangle. Normal TV pictures do not contain highly saturated colors, Computer rendered images displayed directly on a RGB monitor can contain any color in the RGB space broadcast standards however have a more limited color range. The range of NTSC and PAL coded colors is a subset of those available in RGB components because of restrictions on the modulation of the composite signal. The safest way to determine whether an RGB color is legal or not is to calculate its luminance and saturation and to check it against the limits for the composite signal. The saturation should be calculated using the composite weighting factors which are different from the ones in the matrix calculations above. These equations are given so that the colors of rendered objects can be chosen from the legal color space.

The PAL matrix is

```
Y  = 0.299 * r + 0.587 * g + 0.114 * b;
U = -0.147 * r - 0.289 * g + 0.437 * b;
V =  0.615 * r - 0.515 * g - 0.100 * b;
```

$$\text{Saturation} = \sqrt{U^2 + V^2}$$

For RGB values in the range 0.0 to 1.0
Maximum excursion (Y + saturation) must be less than 1.334
Minimum excursion (Y - saturation) must be greater than -0.339

The NTSC matrix is

```
Y  =  0.299 * r + 0.587 * g + 0.114 * b;
I  =  0.596 * r - 0.274 * g - 0.322 * b;
Q  =  0.211 * r - 0.523 * g + 0.312 * b;
```

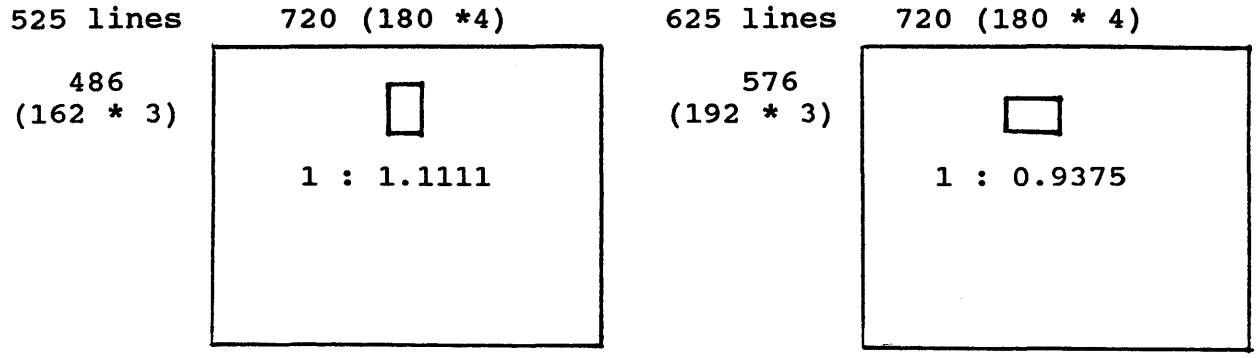
$$\text{Saturation} = \sqrt{I^2 + Q^2}$$

For RGB values in the range 0.0 to 1.0
Maximum excursion (Y + saturation) must be less than or equal to 1.0
Minimum excursion (Y - saturation) must be greater than -0.251

100% Color Bars are not considered valid for transmission in NTSC systems, the color space is normally limited to the 75% luminance (100% saturated) color bars with a 100% white this keeps the signal within the limits of +100 -16 IRE units.

1.8 Pixel Aspect Ratio

The following diagrams show the effective pixel aspect ratio for 525 and 625 line systems. The calculations are based on the unblanked video area fitting the 4:3 screen aspect ratio.



2. SCSI Application Notes

Introduction

This is a general discussion of the capabilities of the SCSI standard, intended to introduce the terminology associated with it. Not all the features and options mentioned here are supported by the A60 or the A64 SCSI adapter.

The SCSI standard has evolved from the Shugart Associates SASI interface. It uses an eight bit parallel data bus with optional parity over which data is transferred using REQ , ACK and ATN handshake lines. At any moment the bus is in one of ten 'phases' specified by the five signals BSY, SEL, C/D, I/O and MSG.

The usual sequence of phases is BUS-FREE SELECTION COMMAND DATA-IN (or OUT) STATUS BUS-FREE. The most confusing aspect of the interface is the way that transfers are controlled by the peripheral device (referred to as the Target) rather than by the host computer (the Initiator). After the Initiator has successfully selected the target it is the target that determines the the information phase by driving the C/D (command/data), I/O (in/out), and MSG (message) lines.

Messages provide an additional (optional) layer of communication in the simplest case the target only sends a command complete message at the end of the SCSI transfer. More complex implementations can use messages to control features like disconnect and synchronous transfer. The ability to support messages is indicated to the Target by the initiator during the selection phase.

Since the Target is driving the REQ line the only way for the initiator break the targets chosen sequence is to assert ATN (Attention) in response to REQ rather than ACK. This action should cause the target to change to MESSAGE OUT phase to allow the initiator to communicate its new information.

This standard encourages device independence by having a common command set applicable to most mass storage peripherals, from streaming tape drives to write-once read-multiple optical disks. The 'read' and 'write' commands issued to a peripheral on the bus deal with logical block numbers rather than cylinder and head numbers.

This allows for a new generation of intelligent peripheral controllers that are able to take care of defect mapping, local backup and even perform local data searches. The block size and the limits of the media (eg max number of blocks) will vary from device to device so the SCSI standard provides commands to allow the host computer to a obtain this information from the controller.

SCSI commands are grouped according to the class of device some commands such as the Test Unit Ready and Request Sense are supported by most devices. Commands such as Read and Write have different parameters depending on the class of the target device. The 'read' command to a direct access device such as a disk has parameters for start block and transfer length. The read command to a sequential access device such as a Tape drive has the same code but only has the transfer length parameter.

2.1 Different levels of implementation

In reality most SCSI implementations are not as generalized as the standard might suggest. Typically the host device drivers will have to be customized slightly for a particular SCSI device to enable or disable some of the optional features of the spec. There are often device dependent commands (especially the format command) which can vary depending on the capabilities of the device. Although the SCSI standard provides a mechanism for determining the limits of a device and the block sizes supported Host computers often make assumptions about the block size.

SCSI ID's

The SCSI bus can address up to eight separate SCSI devices (controllers) each of which in turn can have eight logical units connected to them. There is a proposed extension to the SCSI standard to permit the addressing of 64 devices. Before a command to a particular unit can be issued the appropriate controller has to be selected. Once communication between host and controller is established the SCSI command (such as 'read' or seek') is passed to the target. Part of the command block specifies the logical unit number on the selected device is being addressed.

Multiple Initiators

There can be more than one initiator on the bus in which case an initial Arbitration bus phase has to be completed before a prospective initiator can gain control of the bus. If two devices are both contending for control at the same time the winner will be the one with the highest device ID.

Disconnection

In multiple initiator/target situations it may be useful for the target to disconnect (ie relinquish control of the bus) mid-way through a SCSI transfer in order to allow another initiator or target to transfer data while the first target is performing a seek. Both devices have to support arbitration and messages.

Single Initiator option

In this case the initiator does not have to place its own ID on the bus during selection since there are no other Initiators there is no need for the Target to know it's ID.

SCSI Pointers

The SCSI standard expects the initiator to have some sort of context pointer as well as a data transfer pointer. If the target issues a Save Pointers message the Initiator is expected to save its current context in some way so that it can return to the same state on receipt of a Restore Pointers message. The save and restore pointers messages are normally used before and after a disconnection, they can also be used to return to a known state in the event of an error (ie restarting a transfer at the last position the pointers were saved rather than from the beginning).

Linked Commands

Linked commands enable commands to be grouped together into one SCSI transaction (ie the host does not have to intervene between the individual transfers) this is particularly useful for the search command where the initiator can request the target to search for some particular data and then transfer the block where

the data was found as two linked commands. Linked commands can also be used to ensure that commands using relative addressing do not get separated.

Synchronous Transfers

Synchronous transfer can be used to speed up data transfer phases, instead of using the rigid overlapping REQ/ACK handshake the REQ and ACK lines are pulsed without waiting for a response from the other end. The speed of the transfer can still be limited since a REQ/ACK offset is established which means the target will stop requesting if it is still waiting for the offset limit of ACKs. This permits a maximum data rate of 3.3MHz.

Synchronous mode is enabled by the Target and Initiator exchanging Synchronous Data Transfer Request messages and if necessary entering into negotiation to establish a mutually acceptable transfer period (ie the data rate) and the REQ/ACK offset.

2.2 Physical specifications

In its simplest form the SCSI bus uses 50 way flat Ribbon cable with alternate (odd numbered) lines grounded. All the bus signals are active low. The maximum cable length is 6 meters.

Drive capability

The bus drivers are intended to be open collector or tristate drivers capable of sinking 48 mA (such as 7438 or AM29864).

Termination

The daisy chained bus has to be terminated at both ends with each signal being pulled up to +5v with 220ohm and pulled down to 0V with 330ohm.

Differential drivers

Using the differential drivers option the cable length can be extended to 25 meters using 25 way twisted pair cable. Termination becomes 100ohm between differential pairs and 330ohms from the + signal to ground and 330 between the - signal and +5V. One of the grounded pins becomes a Diff sense signal to protect against plugging single ended systems into differential ones.

Remote terminator power

Up to 1.0 A at 5V for powering remote terminators can optionally be supplied.

2.3 Bus Signals

All Bus signals are active low, the RESET and BSY signals are OR-tied signals that can be driven by more than one device at a time.

Reset	Indicates a bus reset condition
BSY	Indicates the bus is in use
SEL	Indicates Selection or Reselection phase
DB 0-7	Eight bit data bus
DB Parity	Data Bus Parity
REQ	Request for data transfer by the Target
ACK	Acknowledgement of data transfer from Initiator
ATN	Driven by Initiator to indicate attention condition
MSG	Indicates message phase
C/D	Differentiates between Control and Data phases
I/O	Indicates the direction of data transfer

2.4 Bus Phases

	BSY	SEL	C/D	I/O	MSG
Bus Free	1	1	1	1	1
Arbitration	0	0	1	1	1
Selection	0	0	1	1	1
Reselection	0	0	1	0	1
message in	0	1	0	0	0
message out	0	1	0	1	0
command	0	1	0	1	1
data in	0	1	1	0	1
data out	0	1	1	1	1
status	0	1	0	0	1

2.5 Error conditions

The normal mechanism for the target to report an error to the host is for the target to return 'Check Condition' in the status phase. The Initiator is expected to respond with a Request Sense command which allows the target to describe the error condition in more detail.

2.6 Interfacing to the Abekas Disk Recorders

The SCSI interface for A60 and A64 the has been configured to allow it to be interfaced to Hosts using the simplest implementation of the SCSI protocol. For this reason it has a default block size of 512 bytes which the more flexible hosts can change using the Mode select command.

The A60 and A64 expect video data to be in a fixed format so its not possible to reliably place any volume labels or file system structure on the disk to 'fool' an operating system into mounting the A60 or A64 as a normal system disk.

The A60/A64 either has to be accessed through customized device drivers or has to be installed as a 'raw' device.

2.7 Example SCSI Transfer

The following section illustrates the sequence of bus phases and the information transferred to and from the host SCSI controller for a read of a single logical block at 1234H

- Specify Destination ID 7
- Specify Timeout Period

Arbitrate for control of bus

Select with Attention

Message out Phase

- Send Identify (logical unit 0) and enable disconnect [C0]

Command Phase

- Specify Transfer length 6 bytes
- Command block [08]
  [00]
  [12]
  [34]
  [01]
  [00]

Message in Phase (Not implemented by A60/A64)

- Save pointers message [02] (the target wants to seek)

Message in Phase

- Disconnect message [04]

Target disconnected

Reselection

- Determine Reselecting device ID 7

Message in Phase

- Reselecting LUN 0 identified [80]

Message in Phase (Not implemented by A60/A64)

- Restore pointers message [03]

Data in Phase

- Specify Transfer Count 200H bytes

- Transfer data
  (set up DMA for a 512 byte transfer)

Status Phase

- Status OK [00]

Message in Phase

- Command complete Message [00]

Target disconnected

3. CCIR 601 Reference

The CCIR 601 inputs and outputs to the A64 and A60 consist of the following implementation:

Components are referred to as Cb and Cr rather than (B-Y) (R-Y) or U and V This is to avoid confusion with the unweighted color difference signals (B-Y) and (R-Y) and the weighted (analog) difference signals U and V.

Matrices

The equations for obtaining Y, Cb and Cr from R, G and B are as follows :

$$Y = 0.299\ R + 0.587\ G + 0.114\ B$$

$$Cb = -0.1686\ R - 0.3311\ G + 0.4997\ B$$

$$Cr = 0.4998\ R - 0.4185\ G - 0.0813\ B$$

4:2:2 Sampling

The sampling rate is 13.5 MHz for the luminance and 6.5 MHz for each of the chrominance components. The samples are grouped into sequences of four as follows

Cb Y Cr Y Cb Y Cr Y

The first three samples of each group are co-sited, the second luminance sample has no corresponding chrominance information.

Data

The values 00 and FF are reserved for synchronizing the data stream and should not occur in the body of the data.

Luminance information coded as an unsigned byte in the range 10..EB (16..235) where 16 corresponds to black and 235 to peak white. Chrominance components are signed quantities offset by 80 (128) giving a range of 10..F0 (+/-112) (the EBU specification is unclear about the range for chrominance components, it claims there should be 224 quantization levels centered on 128, the problem being you can't center an even range, its either got to be 224 samples -112..+111 or 225 samples -112..+112).

A line of data has the following format

sync preamble	type	data (1440 bytes)	sync	type
FF 00 00	XX	XX XX XX ... XX XX XX	FF 00 00	XX

The type bytes have the following values.

	Video Lines		Vertical Blanking Lines	
	Start	End	Start	End
field 1	80	90	AB	B6
field 2	C7	DA	EC	F1

There are three bits to indicate first field, start of field blanking and the start of horizontal blanking, the rest of the byte is coded to protect the information in these three bits. The protection code allows single bit errors to be corrected and also detects double bit errors.

Blanking

The digital video data covers the the entire picture area, there are no half lines, both fields have the same number of lines. The 525 line system has 243 lines per field and the 625 line system 288.

Lines in the framestore are numbered from 0..607 the first active video line for a 525 line system is 18 corresponding to analog line 20 in the second field, and for a 625 line system it is framestore line 32 corresponding to analog line 23 in the first field.

The leading and trailing edges of the video data should ramp up from black but the transitions in the middle of the half lines are supposed to be generated by some analog component further down the chain and are not specified in the digital signal.

Additional Notes

A60 will free run if there is no analog reference. The A64 will not.

The digital inputs are independent of the output - there is a framestore in the machines.

The digital inputs and outputs do not generate or recognize any ancillary data.

The active video data is not modified inside the machine, data can be written in the active video areas with any value apart from 00 and FF.

4. A60/A64 SCSI Data Format

This section describes the data format of the A60/A64 SCSI port, The video data stream is a similar format to that emanating CCIR 601 digital video ports with the addition of line numbers and the omission of blanking.

The active video data passed through the SCSI interface on the A60/A64 conforms to the standards for 4:2:2 Digital Component Video referred described in the preceding section

There are no vertical blanking lines passed through the SCSI port, horizontal blanking (normally 264 bytes of black) is omitted, in its place is an Abekas format ancillary data section (8 bytes in length) which contains the line number of the following video data. This gives 1456 bytes of data per line: 8 bytes of line number, 4 bytes of sync, 1440 bytes of video and 4 bytes of sync.

The video is passed through the SCSI port a field at a time. The line numbers associated with each line are frame line numbers numbered from 0 so that for a 625 line system the first field contains all the even numbered lines and these are displayed above the corresponding odd lines in the second field.

Be warned that 525 line systems have a strange field order, the first field transmitted in the NTSC system is the lower field so the top line in a frame is actually in field 2.

The sync pattern 00 FF FF is used to characterize ancillary data as stated in the CCIR standard however the rest of the line number data does not conform to any existing standard.

ancillary	(type)	(length)	line number
00 FF FF	64	80 10	XX XX

The 10 bit framestore line number is split over two bytes as follows (to avoid using 00 and FF).

	1 0 X X X X X 0	1 0 X X X X X 0
bit	4 3 2 1 0	9 8 7 6 5

The following two lines are examples of the appropriate coding

line 0 (field 1)

00 FF FF 64 80 10 80 80 FF 00 00 80 [Cb Y Cr Y ...] FF 00 00 90

line 123 (field 2)

00 FF FF 64 80 10 AC 86 FF 00 00 C7 [Cb Y Cr Y ...] FF 00 00 DA

Block sizes

In order to simplify the task of transferring a field the length of a field is rounded up to make it a multiple of most of the popular block sizes.

For a 525 line system a field is 368640 bytes long (0x5A000) which gives 720 blocks of 512 bytes or 45 blocks of 8192 bytes. There are 252 lines in a 525 line field (including blanking), there is one whole line and a fraction at the end of the transfer.

In the case of a 625 line system a field is 450560 bytes long (0x6E000) which gives 880 blocks of 512 bytes or 55 blocks of 8192 bytes. There are 304 lines in a 625 line field (including blanking) so there are five lines and a fraction padding the end of the transfer.

The fractional lines just stop in the middle there is no need for a terminating sync.

Active Lines

In any field lines at the start are used for vertical sync and a further 9 are blanked for 525 lines and 16 for 625, to allow for vertical retrace leaving 243 active lines in a 525 line field and 288 for 625 lines.

	525	625
syncs	10.5	8.5
blanked	9	16
active	243	288
total	262.5	312.5

The framestores in the A64 and A60 are

capable of storing both the active and the blanked lines.

In Broadcast videotape applications the blanked lines often carry Vertical Interval Time Code (VITC) so there is a facility for one or two of them (selected on the miscellaneous menu) to be stored and replayed from disk.

The SCSI port dumps all of the lines in the framestore, that is 252 per 525 line field and 304 for 625 lines. Applications rendering fields for this SCSI format should include the blanked lines at the top of the picture, they should be black since they are only for padding, none of these lines gets recorded unless they are selected as VITC lines.

5. SCSI Reference Manual

The A64 interface will use messages if it is selected with ATN asserted. The Command Complete message is always set at the end of a transfer.

If enabled the following Messages are sent by the A64 Interface.

Identify
Disconnect

The only message out (of the initiator) supported is Identify.

Disconnection is optional and is enabled by the initiator setting the appropriate bit in the initial Identify message.

The 'Save pointers' and 'Restore pointers' messages are not transmitted before/after Disconnection/Reselection.

Disconnection (if enabled) will occur during any command that requires a seek. In the case of framestore data transfers the A64 will disconnect between each field transferred.

The A60 and A64 Interfaces support arbitration. As a Target the A60 and A64 will arbitrate for control of the bus when reselecting the initiator after a disconnection. Arbitration has not been tested with more than two devices on the bus.

The A60 and A64 Interfaces are not capable of servicing overlapping requests from two initiators.

Linked commands are not supported.

Incoming Data Bus Parity is not checked. Outgoing Data Bus Parity is generated.

In the event of a SCSI transfer hanging the SCSI interface will timeout after a period of 2 seconds inactivity and issue a bus reset.

Status returns 'status OK' except for the following errors which generate a check condition.

Illegal Length
- the parameter for a seek or space command is not a field boundary.
- the transport command data is less than 4.

End of Medium
- If a specified block is off the end of the disk.

The error is indicated by a bit in the data returned in response to a 'request sense' command.

There is a switch on the A64 Interface to allow the SCSI bus reset signal to cause a reset of the A64 Interface.

625 line field has 6E000H bytes per field 525 line has 5D000H

Block sizes 100H..1000H(256..4096) Bytes per Block, the default is 512 use the Mode Select and Mode Sense commands to change or verify the size.

Seeks commands should only be issued to field boundaries.

The last field on the disk is not accessible from the SCSI port. For a 25 or 30 second machine attempts to write past 25 or 30 seconds will not be detected by the interface and the results will be unpredictable.

For the A64 the SCSI Target Address is set using a DIP switch on the interface card, on the A60 the DIP switch is located on the Computer Card, each controller on the bus including the host has to have a unique number.

Remote control single frame recording 4-5 frames a second. The SCSI Interface is capable of a maximum asynchronous transfer rate of 1.1 Mbyte / sec.

Synchronous transfers are not yet supported.

5.1 Logical blocks

The block size can vary from 256 bytes to 4096 bytes. This information can be obtained using the SCSI Block Limits command.

The default block size is 512 bytes. The block size can be changed using the Mode Select command and confirmed using the Mode Sense command.

The only restriction on the block size is that it must be an even factor of the field size.

Note that if the power is cycled on the A64 Interface or some error causes the SCSI interface to reset, the block size will return to 512 bytes.

The block size used for the A60/A64 Offline Storage tape is 6C10H. Tapes should be compatible between A64s and A60s.

The A60/A64 SCSI interface only accepts data as fields, a frame of video has to be transferred as two fields.

Transfers less than a field in length are buffered in the A60/A64 framestore until the last block in the field is written, at which point the whole field is flushed to the disk.

For a single block transfer sequence to complete correctly it must start on field boundary (eg logical block address MOD blocks per field == 0) and end with a logical block address MOD blocks per field == block per field - 1, in other words even though it is possible to transfer units of less than a field the SCSI interface will not accept transfers that aren't grouped as fields.

5.2 SCSI Commands

The A60/A64 can be treated as either a Sequential device (eg Tape) or a Direct Access device (eg Disk). Commands are supplied that support either model. Since there is some overlap (both read commands have the same op code but different parameters) Direct Access commands should be addressed to Logical unit zero and Sequential access commands to Logical unit one.

The only occasion where the Logical Unit field is important is for the Read and Write commands which have different formats for direct and sequential access.

For sequential access the SCSI Interface requires that the user issues a rewind command (or a direct access seek command) before performing any sequential actions, this is because the SCSI interface has no way of determining the position of the disk heads it has to remember what has happened since it last issued a goto command to the A60/A64.

Similarly if the control panel is used or if serial protocol commands are issued some form of direct access command must be issued before a sequential read or write will work correctly.

The description for each command shows the structure of the command and gives the sequence of bytes that are expected - information supplied by the initiator is shown as XX. The command description also shows the SCSI Bus phases that can be expected and whether or not the target will disconnect during the course of the transfer.

Command Summary

Commands for both Device Types

Test Unit Ready
Request Sense
Inquiry

Direct Access Device Commands

Rezero Unit
Read
Write
Seek
Extended Read
Extended Write
Extended Seek

Sequential Access Device Commands

Rewind
Read Block Limits
Read
Write
Space
Mode Sense
Mode Select

Vendor Unique Command

Transport Commands

Test Unit Ready

The A64 SCSI Adapter and the A60 will always return OK Status for this command providing they are up and running normally. The SCSI Adapter cannot respond if the link between it and the A64 is broken or the A64 is powered off.

	7	6	5	4	3	2	1	0	
0	[-	Command		-	-	-	-	-]	00
1									00
2									00
3									00
4									00
5									00

Disconnect No

Status OK

Request Sense

This command should be issued by the Initiator in the event that Check Condition Status is returned by the A60/A64.

Non extended sense data format is not supported so the allocation length (Command byte 3) should be 8 or greater.

The data returned is all zeroes except for the Incorrect Length indicator or the End of Medium bits are set depending on the error condition.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-]		03
1									00
2									00
3									00
4	[-	Allocation Length				-	-]		08
5									00

Disconnect No

Data In

	7	6	5	4	3	2	1	0		
0	VAL	[-	Class	-]	[-	Code		-]	70	
1									00	
2		EOM	ILI						X0	Error Flags
3									00	
4									00	
5									00	
6									00	
7									00	

Status OK

Rezero Unit

Seek to Field Zero

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-]		01
1	[-	LUN	-]						00
2									00
3									00
4									00
5									00

Disconnect Yes

Status OK

Time to complete - Four fields.

Read (Direct)

Read up to 255 blocks of data from the A60/A64. Must be Issued to Logical Unit 0 if the direct access format is to be used.

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-	-]	08
1	[-	LUN	-]	[-	LB Addr MSB			-]	XX
2	[-	Logical Block Addr				-		-]	XX
3	[-	Logical Block Addr LSB						-]	XX
4	[-	Transfer length		-		-		-]	XX
5									00

Disconnect Yes

Data In

Length * Block Size bytes of composite video

Status OK or Check Condition

Time to complete - For each field read the A64 will disconnect for four fields and then take at least six fields to DMA the data out of the Store. If single block transfers are used the A60/A64 will disconnect before transferring each block but only the first block of each field will disconnect for longer than a few milliseconds.

Write (Direct)

Must be Issued to Logical Unit 0 if the direct access format is to be used.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-	-]	0A
1	[-	LUN	-]	[-	LB Addr MSB			-]	XX
2	[-	Logical Block Addr				-		-]	XX
3	[-	Logical Block Addr LSB						-]	XX
4	[-	Transfer length			-	-		-]	XX
5									00

Disconnect Yes

Data Out

Length * block size bytes of composite video

Status OK or Check Condition

Time to complete - For each field written the A60/A64 will disconnect for nine fields and then take at least six fields to DMA the data in to the Store.

Seek

Causes a seek to the given Logical Block address

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-]		0B
1	[-	LUN	-]	[-	LB Addr MSB		-]		XX
2	[-	Logical Block Addr				-	-]		XX
3	[-	Logical Block Addr LSB					-]		XX
4									00
5									00

Disconnect Yes

Status OK or Check Condition

Time to complete - Four fields.

Read (Direct Extended)

The Extended Read command allows for longer transfers and larger Logical Block addresses than the 6 byte command.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-]		28
1	[-	LUN	-]						00
2	[-	Logical Block Addr MSB					-]		XX
3	[-	Logical Block Addr				-	-]		XX
4	[-	Logical Block Addr				-	-]		XX
5	[-	Logical Block Addr LSB					-]		XX
6									00
7	[-	Transfer length MSB			-		-]		XX
8	[-	Transfer length LSB			-		-]		XX
9									00

Disconnect Yes

Data Out

Length * block size bytes of composite video

Status OK or Check Condition

Time to complete - For each field read the A60/A64 will disconnect for four fields and then take at least six fields to DMA the data out of the Store.

Write (Direct Extended)

The Extended Write command allows for longer transfers and larger Logical Block addresses than the 6 byte command.

Command

	7	6	5	4	3	2	1	0	
0	Command								2A
1	LUN								00
2	Logical Block Addr MSB								XX
3	Logical Block Addr								XX
4	Logical Block Addr								XX
5	Logical Block Addr LSB								XX
6									00
7	Transfer length MSB								XX
8	Transfer length LSB								XX
9									00

Disconnect Yes

Data In

Length * block size bytes of composite video

Status OK or Check Condition

Time to complete - For each field written the A60/A64 will disconnect for nine fields and then take at least six fields to DMA the data in to the Store.

Seek (Extended)

Causes a seek to the given Logical Block address the extended seek command allows a larger addressing range which is necessary for the smaller sizes of logical block.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-	-]	2B
1	[-	LUN	-]						00
2	[-	Logical Block Addr MSB						-]	XX
3	[-	Logical Block Addr					-	-]	XX
4	[-	Logical Block Addr					-	-]	XX
5	[-	Logical Block Addr LSB					-	-]	XX
6									00
7									00
8									00
9									00

Disconnect Yes

Status OK or Check Condition

Time to complete - Four fields.

Rewind

Rewind to Field Zero

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-]		01
1	[-	LUN	-]						00
2									00
3									00
4									00
5									00

Disconnect Yes

Status OK

Time to complete - Four fields.

Read Block Limits

Check the range of available block lengths.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-]		05
1	[-	LUN	-]						00
2									00
3									00
4									00
5									00

Disconnect No

Data In

	7	6	5	4	3	2	1	0	
0									00
1	[-	Max Block Length MSB						-]	03
2	[-	Max Block Length						-]	00
3	[-	Max Block Length LSB						-]	00
4	[-	Min Block Length MSB						-]	01
5	[-	Min Block Length LSB						-]	00

Status OK

Read (sequential)

Must be Issued to Logical Unit 1 if the sequential access format is to be used.

Command

	7	6	5	4	3	2	1	0		
0	[-	Command	-	-	-	-	-	-]	08	
1	[-	LUN	-]						20	(LUN 1)
2	[-	Transfer Length MSB						-]	XX	
3	[-	Transfer Length				-		-]	XX	
4	[-	Transfer Length LSB						-]	XX	
5									00	

Disconnect Yes

Data In

Length * block size bytes of composite video

Status OK or Check Condition

Time to complete - For each field read the A60/A64 will disconnect for four fields and then take at least six fields to DMA the data out of the Store.

Write (sequential)

Must be Issued to Logical Unit 1 if the sequential access format is to be used.

Command

	7	6	5	4	3	2	1	0		
0	[-	Command	-	-	-	-	-]		08	
1	[-	LUN	-]						20	(LUN 1)
2	[-	Transfer Length MSB						-]	XX	
3	[-	Transfer Length				-		-]	XX	
4	[-	Transfer Length LSB						-]	XX	
5									00	

Disconnect Yes

Data Out

Length * block size bytes of composite video

Status OK or Check Condition

Time to complete - For each field written the A60/A64 will disconnect for nine fields and then take at least six fields to DMA the data in to the Store.

Space

Space by 'offset' in either direction - where 'offset' is a twos complement signed number. Only steps in Blocks all 'codes' other than 0 are ignored.

Command

	7	6	5	4	3	2	1	0		
0	[-	Command		-	-	-	-	-]	11	
1	[-	LUN	-]				[Code]		20	(LUN 1)
2	[-	-	Offset MSB		-	-	-	-]	XX	
3	[-	-	Offset		-	-	-	-]	XX	
4	[-	-	Offset LSB		-	-	-	-]	XX	
5									00	

Disconnect Yes

Status OK or Check Condition

Time to complete - Four fields.

Mode Select

Select the target block size. The other parameters specified for this command - density code and number of blocks are not supported.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-	-]	15
1	[-	LUN	-]						00
2									00
3									00
4	[-	Parameter List length						-]	0C
5									00

Data Out

	7	6	5	4	3	2	1	0		
0									00	
1	[-	-	Medium Type				-	-]	00	
2									00	
3	[-	Block Descriptor Length						-]	08	
4									00	Density code and
5									00	Number of blocks
6									00	Ignored by A64
7									00	
8									00	
9	[-	-	Block Length MSB				-	-]	XX	
A	[-	-	Block Length				-	-]	XX	
B	[-	-	Block Length LSB				-	-]	XX	

Mode Sense

Mode sense returns information on the size and number of blocks on the device.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-	-]	1A
1	[-	LUN	-]						00
2									00
3									00
4									00
5									00

Data Out

	7	6	5	4	3	2	1	0	
0	[-	- Sense Data Length					-	-]	0B
1	[-	-	Medium Type				-	-]	00
2									00
3	[-	Block Descriptor Length						-]	08
4	[-	-	Density Code				-	-]	00
5	[-	Number of Blocks MSB						-]	23
6	[-	Number of Blocks					-	-]	28
7	[-	Number of Blocks LSB						-]	00
8									00
9	[-	- Block Length MSB					-	-]	00
A	[-	- Block Length					-	-]	01
B	[-	- Block Length LSB					-	-]	00

Bugs : The number of blocks is based on the assumption that the A60/A64 is a fully loaded (100 sec) system. The interface has no way of checking.

Transport Commands

Up to seven A60/A64 RS232 Port Commands can be sent in the Data portion of this Command (refer to the separate document "A64 External Protocol Manual"). Note that for RS232 protocol parameters of more than one byte the significance of the bytes is reversed from the normal SCSI ordering.

There is no returned Status information available from the A64.

The Data Length in byte 4 of the command should indicate the total number of bytes in the data block (eg 4 x number of commands).

There is no need to terminate the command sequence with a Carriage Return character as specified in the Parallel Port interface Manual.

Command

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-	-]	70
1	[-	LUN	-]						00
2									00
3									00
4	[-	Data Length		-	-	-	-	-]	XX
5									00

Disconnect Yes

Data Out

	7	6	5	4	3	2	1	0	
0	[-	Command	-	-	-	-	-	-]	XX
1	[-	Param 0	-	-	-	-	-	-]	XX
2	[-	Param 1	-	-	-	-	-	-]	XX
3	[-	Param 2	-	-	-	-	-	-]	XX
n+0	[-	Command	-	-	-	-	-	-]	XX
n+1	[-	Param 0	-	-	-	-	-	-]	XX
n+2	[-	Param 1	-	-	-	-	-	-]	XX
n+3	[-	Param 2	-	-	-	-	-	-]	XX

Status OK or Check Condition if Illegal Length is given

Time to Complete Always Four Fields

6. A60/A64 Offline Storage Data Format

The following section defines the format of the data on tapes generated by and readable with the offline storage option. The video data is identical to that for the SCSI interface (see the SCSI Data Format section).

6250 bpi 9 Track GCR Format (ANSI X3.54-1976)

Block Size 27664 (6C10H) bytes - 16 blocks per field 625 line system (14 for 525). The active video data is stored in the same format as the SCSI data format in the preceeding section, there are padding lines of garbage to fill out the last block after the end of active video.

One Segment per tape.

Approx 7 secs on a 2400 ft reel 10 secs on 3600 ft.

Two File marks denote the end of segment.

7. IBM PC Interface

This section describes how the A60/A64 SCSI interface can be connected to an IBM PC-XT.

The Advanced Storage Concepts ASC-88 (see below) is an example of an inexpensive SCSI Host adapter that is supplied with software tools to allow it to be interfaced to most SCSI peripherals and was used to test the A62 SCSI protocol.

The SCSI transfer is controlled with a Command Block structure which contains the body of the SCSI command, a pointer to the data (if there is any to be transferred) and the SCSI status byte. The command block also contains a transfer count, the destination ID and LUN and various flags to indicate the direction of the transfer, whether messages are supported and whether disconnect is supported.

Invoking a SCSI transfer is simply a matter of loading a pointer to the Command block into the appropriate registers and calling the SCSI driver through an interrupt.

The maximum size of a Transfer was limited to 64K bytes by the host adapter drivers (and the PC's DMA controller).

Example programs are available from Abekas.

Manufacturers of PC Host Adapters

Advance Storage Concepts Inc ($199)

9660 Hillcroft Ave.,
Suite 325,
Houston, Texas 77096
(713) 727-6388

Micro Design International Inc ($190)

6985 University Blvd.,
Winter Park,
Florida 32792
(305) 677-8333

8. Sun Microsystems Interface

Introduction

This section shows how an Abekas disk recorder can be connected to a Sun Microsystems 3/160 or a 3/50 Workstation. Unless there is already an unused SCSI disk configured on your system the UNIX kernel on the Sun has to be reconfigured and device files have to added.

Sun 3/160 The SCSI devices in a 3/160 pedestal are located in the top, there is no external interface connector so some way has to be found to daisy-chain the the 50-way SCSI cable from the existing SCSI devices to the Abekas SCSI interface. The proper way is probably to add an extra SCSI Host Adapter VME card which would give a proper connector on the rear of the pedestal.

Sun 3/50 the SCSI interface is via a 50 way D-Type connector on the back. If you are making up your own connector note that the pin numbers are arranged for ribbon cable with an insulation displacement connector rather than the usual D-Type order.

The Abekas Disk Recorder SCSI Interface allows it to mimic the operation of a regular Sun SCSI disk. In particular when the Sun reads block zero on the raw device the Disk Recorder returns a valid disk label. There is no file structure on the disk, data is transferred to and from the Disk Recorder by lseek(), read() and write() calls to the raw device (in this case '/dev/rsd2c').

There is no need to format the disk or write a label to it, indeed since video data written to the Disk Recorder has to be in a video format there is no guarantee that normal (ie directory) data written to the Disk Recorder will not be 'changed'.

8.1 SCSI Bus Addresses

In the example given here the Disk Recorder will be configured as /dev/sd2 (which has SCSI id 1; Logical unit 0). The Sun assignment of SCSI bus addresses is given below

	SCSI id	SCSI lun	Sun Drive #
host	7		
sd0	0	0	0
sd1	0	1	1
sd2	1	0	8
st0	4		32
st1	5		40

Note that the Sun drive numbers (used in the configuration file below) are actually (ID * 8) + LUN

The Disk Recorder cannot be installed as sd1 since it should be accessed as a logical unit 0.

If an Abekas streaming Tape drive is also on the bus it should be given SCSI id 2,3 or 6 so as not to conflict with any of the Sun devices.

8.2 Disk Labels

The Sun device drivers expect the disk label to be the first 512 bytes on the disk so the Abekas Disk Recorder returns a disk label (which is actually stored in the SCSI prom) when it recieves a request to read the single logical block 0.

If the Disk Recorder receives a read command for more than one block starting at 0 it returns data from the disk.

The disk label for an A60/A64 describes the disk as having all its storage in the 'c' partition. Most I/O transfers from the Sun appear to be in groups of 16 blocks at a time.

The Disk Recorder appears as a large raw device to a program running under UNIX so the video data can be accessed by first opening the file /dev/rsd2c and then using lseek() and write() calls.

8.3 Machine Control

In the same way that single block reads to block 0 are treated differently, single block writes to block 0 can be used to pass a set of remote commands to the Disk Recorder. This allows Sun users to access the Transport Commands without having to use the Vendor unique SCSI command.

The format of the single 512 byte block is as follows: up to 7 four byte Remote Protocol Command packets terminated by carriage return character. The rest of the block can be padded with zeroes.

8.4 Reconfiguring the kernel

If there is no /dev/rsd2c in /dev the kernel has to be reconfigured by following the procedure in chapter 8 of the Sun "Installing UNIX" manual. The notes here are intended only as a guide - if in doubt consult the Sun Documentation !

First locate the configuration file, it should be in the /usr/sys/conf directory as SYSTEM_NAME in capitals. Where system name is the name printed in brackets when UNIX is booted:

```
~~~~
Boot: xx(0,0,0)vmunix
Size nnnnn+nnnnn+nnnnn bytes
Sun Unix 4.2 Release 3.2 (SYSTEM_NAME) #1 : date created
~~~~
```

If the default configuration was used when the system was setup the name might be "GENERIC" in which case the conf file is going to be something like SDST50 (for a stand alone 3/50 with a SCSI disk and tape) rather than /usr/sys/conf/GENERIC which is the example of all of the possible options.

If there is no SCSI controller in the config file the following has to be added for a 3/50:

```
controller      si0 at obio ? csr 0x140000 priority 2
```

for a 3/160:

```
controller      sc0 at vme24d16 ? csr 0x200000 priority 2 vector
scintr 0x40
```

Once the scsi controller is specified the following line has to be added for a 3/50:

```
disk sd2 at "scsi controller" si0 drive 8 flags 0
```

and for a 3/160:

```
disk sd2 at "scsi controller" sc0 drive 8 flags 0
```

then execute:

```
# /etc/config SYSTEM_NAME
# cd ../SYSTEM_NAME
# make
```

which produces a kernel file /usr/sys/SYSTEM_NAME/vmunix

Before intalling this as the boot kernel it is best to save the current one as vmunix.old

```
# cp /vmunix /vmunix.old
# mv /usr/sys/SYSTEM_NAME/vmunix /vmunix
```

Now we have to install the device special files by using mknod or the /dev/MAKEDEV script

```
# cd /dev
# MAKEDEV sd2
```

This actually installs both raw and blocked special files for partitions a-h of SCSI disk even though we only intend to use one of them.

We also have to allow reads and write to the raw disk - not normally encouraged

```
# chmod 666 /dev/rsd2c
```

8.5 Configuring the Abekas SCSI interface

Note that in this configuration the Abekas SCSI interface is at bus id '1' so the DIP switches on the interface card have to be set accordingly (SW1 position 1 'on' 2 and 3 'off').

If the Abekas SCSI Interface is not at the end of the SCSI cable the terminating resistors will have to be removed.

If the link between the Abekas SCSI interface and the Disk Recorder Framestore Data Port is broken or the Disk Recorder main chassis is powered down but the Interface is running the SCSI bus will become really confused.

Now when you reboot the Sun with the Abekas SCSI Interface connected you should see the Disk Recorder Label in the opening messages:

```
~~~~
sd2 at si0 slave 8
sd2: <Abekas A64 Digital Disk Recorder>
~~~~
```

This proves the Disk Recorder has successfully returned the disk label.

Now you should be able to use the cppic.c proggy to transfer pictures to the Disk Recorder.

Bugs

There is a problem involving accesses to the later fields on the

disk. This is because the Sun drivers only use regular rather than extended reads and writes and they have a maximum block addressing range of 0x1FFFFF. Fields number 2912 or 48.16 on 525 line systems (2383 or 47:16 on 625) are inaccesible from the Sun.

The SCSI interface can confuse the SCSI bus if the A62 is powered off or incorrectly connected and a transfer is attempted (this includes rebooting the Sun).

Warning

Care should be exercised when running UNIX off a disk on the same SCSI cable as the Abekas Interface - In particular cycling the power on the SCSI Interface box can cause unexpected SCSI Bus Resets.

8.6 Example program

```
/************************************************************
 * cppic - copy pic to A64
 ************************************************************
 * copyright (c) 1987 Abekas Video Systems Inc              */

#include <stdio.h>
#include <sys/file.h>

#define FIELD_LEN 368640
#define BUFF_LEN 368640 /* there's no shortage of RAM here */

main(argc, argv)
int argc;
char **argv;
{
long tmp;
int i, a64, field;
static char buff[BUFF_LEN];

if(argc != 3)
    {
    printf("Usage : cppic file field_num\n");
    exit(0);
    }
if((a64 = open("/dev/rsd2c", O_RDWR)) == -1)
    {
    printf("Unable to open /dev/rsd2c\n");
    exit(1);
    }
if((field = open(argv[1], O_RDONLY)) == -1)
    {
    printf("Unable to open %s0, argv[1]);
    exit(1);
    }
tmp = atoi(argv[2]);
tmp *= FIELD_LEN;
lseek(a64, tmp, L_SET);
for(i=0; i<(FIELD_LEN / BUFF_LEN); i++)
    {
    read(field, buff, BUFF_LEN);
    write(a64, buff, BUFF_LEN);
    }
}
```

9. A60/A64 SCSI Interface Hardware

9.1 External Connections

The A64 framestore data port is linked to the SCSI Adapter with a 50 way Ribbon cable connector. This cable should be as short as possible ideally the SCSI Adapter should be placed directly above the Signal Chassis.

Be sure to observe the correct polarity when connecting the flat cables between the A64 and the Interface. The edge of the cable with the red stripe shows which end is pin 1, there is also an arrow on the connector to indicate pin 1.

If the Cable between the A64 and the SCSI Interface is connected incorrectly the Message "No Streamer Connected" will appear when the backup menu is selected.

If the "No Streamer connected" message only appears once a backup has been started The SCSI configuration (either Cables or Terminators or Addresses) could be at fault.

Both the A64 SCSI Interface and the A60 have a 50 Way ribbon connector for the SCSI port. If the Abekas box is not at the end of the bus the 50 way connector will have to be crimped in the middle of the cable. The Maximum length for the SCSI bus cable is specified as 6 metres. Remote terminator power is not provided.

9.2 Termination

If there are more than two devices on the SCSI Bus only the devices at the ends should have terminating resistors. The location of the terminating resistors in the Abekas devices is as follows. (When replacing the resistor packs be sure to check the polarization, pin one should be marked both on the resistor and the Silk Screen on the board).

A64 SCSI Adapter

RP2 and RP4 220ohm pull-ups
RP3 and RP5 330ohm pull-downs

A60 Computer

RN15 RN16 RN21 (all the same 220/330ohm pull-up/downs)

Kennedy Streaming Tape Drive

Rear Card RN3 RN9 RN10

9.3 SCSI Connector assignment

Odd pins
1 - 23 27 - 49 Ground 25 No Connection

Even pins

2	DB0	28	Ground
4	DB1	30	Ground
6	DB2	32	ATN
8	DB3	34	Ground
10	DB4	36	BSY
12	DB5	38	ACK
14	DB6	40	Reset
16	DB7	42	MSG
18	DB Parity	44	SEL
20	Ground	46	C/D
22	Ground	48	REQ
24	Ground	50	I/O
26	No Connection		

9.4 SCSI Address Settings

The SCSI bus ID's of both the Disk Recorder and the Offline Storage Tape Drive are configured as three bit binary numbers on DIP switches. The 3 switches indicate a number between 0 and 7. The factory settings are for the SCSI Disk recorder to be at ID=0 and the Tape Drive to be ID=7. Note that the binary sense of the Tape and SCSI ID is reversed.

If there are other devices on the SCSI bus these settings may have to be changed as no two devices on the same bus can have the same address.

The SCSI ID of the Abekas Interface should be a different number from the Tape ID

The Tape ID should be the same as the number set on the SCSI Board in the Tape Drive.

When using the Abekas Disk recorder as a SCSI Target (eg remote control from another computer) be sure that the SCSI ID of the host computer is set to something other than the ID of both the Abekas Interface and the the Tape Drive.

If the Interface is being used purely for remote control (No Tape drive connected) then the Tape ID setting on the Abekas interface is not relevant.

A64 SCSI Adapter
SW1 is a DIP switch to configure the following :

SW	off	on	Function
1	0	1	SCSI ID bit 0
2	0	1	SCSI ID bit 1
3	0	1	SCSI ID bit 2
6	1	0	Tape ID bit 0
7	1	0	Tape ID bit 1
8	1	0	Tape ID bit 2

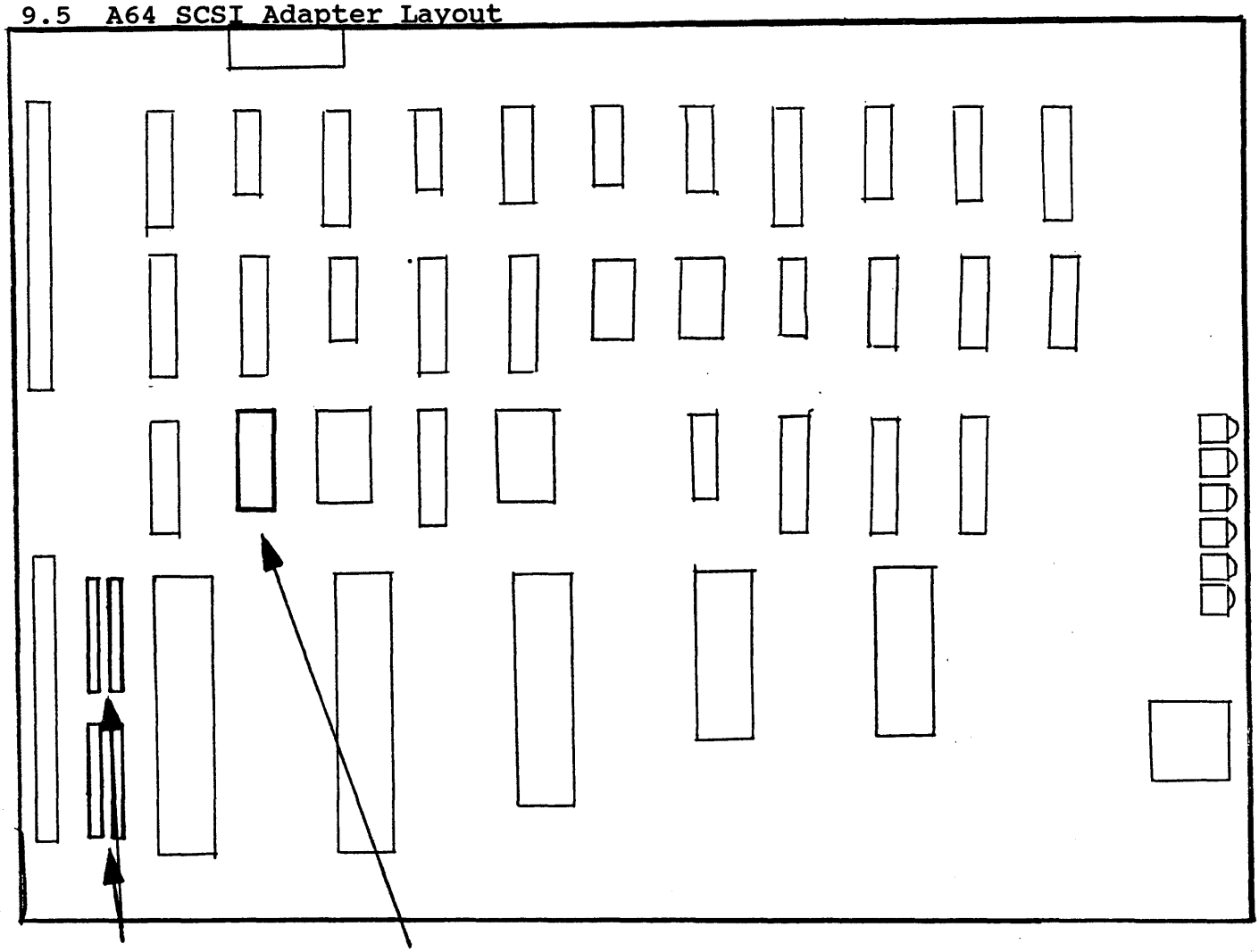
A60 Computer
DIP Switch at position 15C

1	0	1	SCSI ID bit 0
2	0	1	SCSI ID bit 1
3	0	1	SCSI ID bit 2
4	1	0	Tape ID bit 0
5	1	0	Tape ID bit 1
6	1	0	Tape ID bit 2

Kennedy Streaming Tape Drive
Rear board SW1

6	0	1	Tape ID bit 0
7	0	1	Tape ID bit 1
8	0	1	Tape ID bit 2

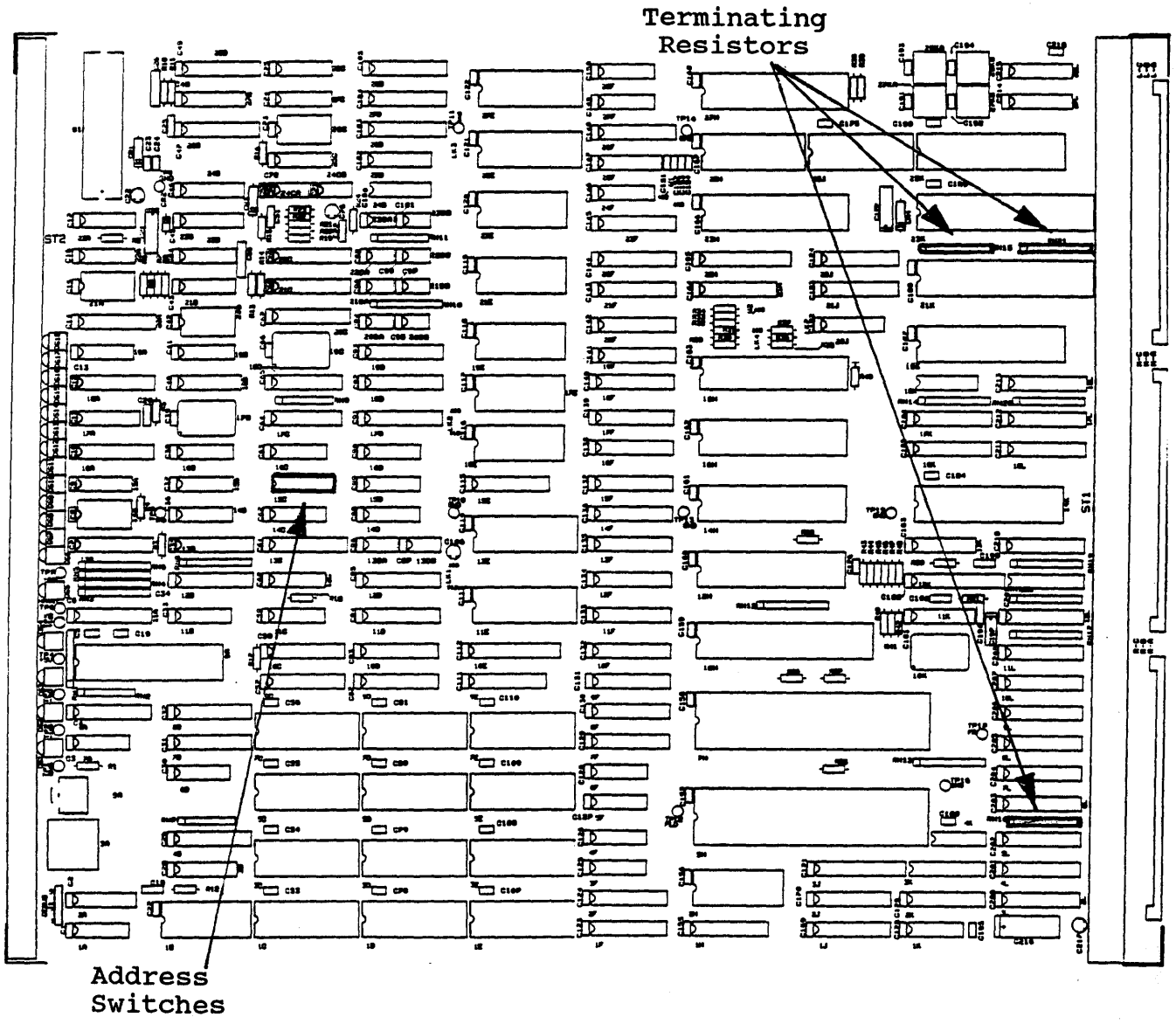
9.5 A64 SCSI Adapter Layout



The A64 interface is Factory set SCSI ID = 0; Tape = 7

1	2	3	4	5	6	7	8
off	off	off	ON	off	off	off	off

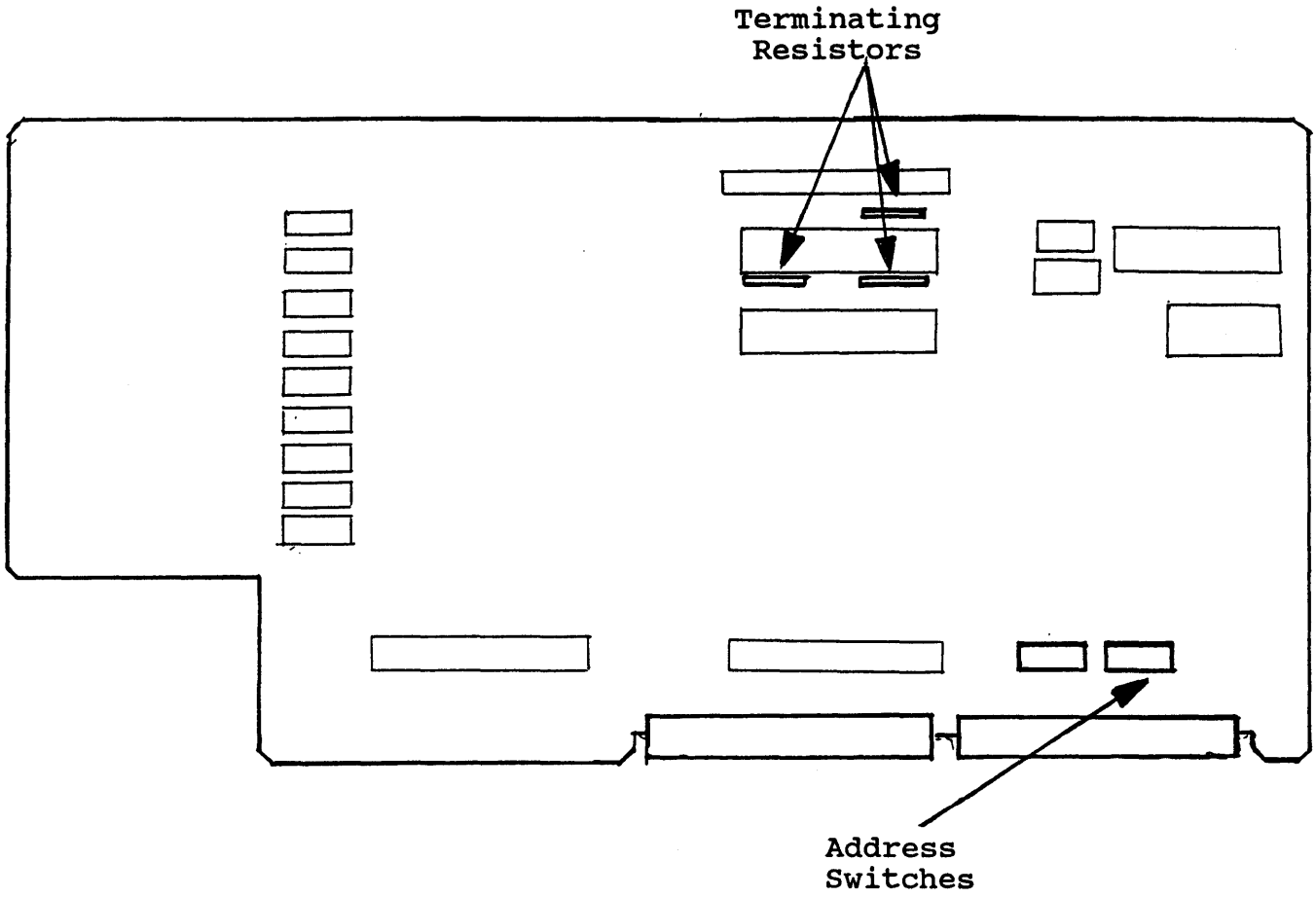
9.6 A60 Computer Layout



Factory setting SCSI ID = 0; Tape = 7

1	2	3	4	5	6	7	8
off	off	off	off	off	off	off	off

9.7 Kennedy SCSI Controller Layout



The Kennedy drive has a factory setting of ID = 7

SW2							
1	2	3	4	5	6	7	8
off	ON	off	off	off	off	off	off

SW1							
1	2	3	4	5	6	7	8
off	off	off	off	off	ON	ON	ON

10. Example Conversion Program

```
/*
 * Program to generate 4:2:2 Component video from RGB
 *
 * copyright (c) 1987 Abekas Video Systems Inc.
 *
 * Note that efficiency has been traded in favor of clarity
 *
 * For instance its fairly easy to use fixed point integer
 * math for most of the calculations
 *
 * The Filtering delays the output data
 * by 3 pixels, it also runs the end of a line into
 * the start of the next
 */

#include <stdio.h>
#include <math.h>

#define LINE_LENGTH 720
#define FRAME_LENGTH 486

int get_rgb(rgbfile, R, G, B)
FILE *rgbfile;
int *R, *G, *B;

{
if(feof(rgbfile)) return(0);

*R = fgetc(rgbfile);
*G = fgetc(rgbfile);
*B = fgetc(rgbfile);
return(1);
}

shift_up(val, buff)
double val, buff[];
{
int n;
for(n=3; n>=0; n--)     /* shift up existing contents */
    buff[n+1] = buff[n];
buff[0] = val;          /* add latest value */
}
```

```
main(argc, argv)
int argc;
char **argv;
{
float r, g, b, y, u, v;
int R, G, B, tmp, n, line, pixel;
FILE *rgbfile, *yuvfile;
                        /* buffers for filtering data */
double y_buff[5];       /* Y buff is one sample longer to delay it */
double u_buff[5];       /* in line with the chroma */
double v_buff[5];       /* which is sampled at half the frequency */

if(argc != 3)
    {
    fprintf(stderr, "Usage : convert from_file to_file\n");
    exit(0);
    }

if((rgbfile = fopen(argv[1], "rb")) == NULL)
    {
    fprintf(stderr, "Unable to open file %s\n", argv[1]);
    exit(1);
    }

if((yuvfile = fopen(argv[2], "wb")) == NULL)
    {
    fprintf(stderr, "Unable to open file %s\n", argv[2]);
    exit(1);
    }

line = FRAME_LENGTH;
while(line)
    {
    /* deal with an U Y V Y sequence each time round */

    pixel = LINE_LENGTH / 2;
    while(pixel)
        {
        /* first pixel gives Y and both chroma */

        get_rgb(rgbfile, &R, &G, &B);

        r = R / 255.0;
        g = G / 255.0;
        b = B / 255.0;
```

```
y = 0.299 * r + 0.587 * g + 0.114 * b;

shift_up(y, y_buff);

u = -0.1686 * r - 0.3311 * g + 0.4997 * b;

shift_up(u, u_buff);

v = 0.4998 * r - 0.4185 * g - 0.0813 * b;

shift_up(v, v_buff);

u =  (0.14963 * u_buff[0]) /* filtered value */
    +(0.22010 * u_buff[1])
    +(0.26054 * u_buff[2])
    +(0.22010 * u_buff[3])
    +(0.14963 * u_buff[4]);

u *= 224.0;              /* -112 .. +112 range */
tmp = floor(u) + 128;    /* centered on 128 */
putc(tmp, yuvfile);

y = -(0.05674 * y_buff[0])
    +(0.01883 * y_buff[1])
    +(1.07582 * y_buff[2])
    +(0.01883 * y_buff[3])
    -(0.05674 * y_buff[4]);

y *= 219.0;              /* 16 .. 235 range */
tmp = floor(y) + 16;     /* offset by 16 */
putc(tmp, yuvfile);

v =  (0.14963 * v_buff[0])
    +(0.22010 * v_buff[1])
    +(0.26054 * v_buff[2])
    +(0.22010 * v_buff[3])
    +(0.14963 * v_buff[4]);

v *= 224.0;
tmp = floor(v) + 128;
putc(tmp, yuvfile);
```

```
        /* second pixel just yields a Y */

        get_rgb(rgbfile, &R, &G, &B);

        r = R / 255.0;
        g = G / 255.0;
        b = B / 255.0;

        y = 0.299 * r + 0.587 * g + 0.114 * b;

        shift_up(y, y_buff);

        u = -0.1686 * r - 0.3311 * g + 0.4997 * b;

        shift_up(u, u_buff);

        v = 0.4998 * r - 0.4185 * g - 0.0813 * b;

        shift_up(v, v_buff);

        y = -(0.05674 * y_buff[0])        /* only filter a Y */
            +(0.01883 * y_buff[1])
            +(1.07582 * y_buff[2])
            +(0.01883 * y_buff[3])
            -(0.05674 * y_buff[4]);

        y *= 219.0;
        tmp = floor(y) + 16;
        putc(tmp, yuvfile);

        pixel--;
        }
    line--;
    }
}
```

11. SCSI format program

```
/*
 * toscsi.c - add all the extra junk to a yuv file for A60/A64
 * takes in a frame and spits out two seperate fields
 *
 * copyright (c) 1987 Abekas Video Systems Inc
 *
 */

#include <stdio.h>
#include <sys/file.h>

#define LINES_525 1  /* this is a 525 line system */

#define LINE_LEN 1440
#define JUNK 16

#ifdef LINES_525
#define BLANK_LINES 9   /* size of vertical blanking */
#define FIELD_LEN 243   /* lines per field */
#define PAD_LINES 1     /* complete lines to pad to end of block */
#define FRAC_LINE 272   /* number of bytes to pad to end of block */
#endif

#ifdef LINES_625
#define BLANK_LINES 16  /* size of vertical blanking */
#define FIELD_LEN 288   /* lines per field */
#define PAD_LINES 5     /* complete lines to pad to end of block */
#define FRAC_LINE 656   /* number of bytes to pad to end of block */
#endif

int field_1, field_2;
```

```
putline(line, buff)
int line;
char buff[];
{
if(!(1 & line)) /* test for even numbered line */
    {
    /* sync type bytes for field 1 */

    buff[11] = 0x80;
    buff[1455] = 0x90;

    /* even line number */

    buff[6] = ((line & 0x1F) << 1) | 0x80;
    buff[7] = ((line & 0x3E0) >> 4) | 0x80;

    write(field_1, buff, LINE_LEN + JUNK);
    }
else {
    /* sync type bytes for field 2 */

    buff[11] = 0xC7;
    buff[1455] = 0xDA;

    /* odd line number */

    buff[6] = ((line & 0x1F) << 1) | 0x80;
    buff[7] = ((line & 0x3E0) >> 4) | 0x80;

    write(field_2, buff, LINE_LEN + JUNK);
    }
}
```

```
main(argc, argv)
int argc;
char **argv;
{
long tmp;
int i, line, src_file;
static char buff[LINE_LEN + JUNK];
char *video_data;
char str[20];

if(argc != 3)
    {
    printf("Usage : toscsi src_file dst_file\n");
    exit(0);
    }
if((src_file = open(argv[1], O_RDONLY)) == -1)
    {
    printf("Unable to open %s\n", argv[1]);
    exit(1);
    }
sprintf(str, "%s.f1", argv[2]);
if((field_1 = open(str, O_WRONLY | O_CREAT | O_TRUNC)) == -1)
    {
    printf("Unable to open %s\n", str);
    exit(1);
    }
sprintf(str, "%s.f2", argv[2]);
if((field_2 = open(str, O_WRONLY | O_CREAT | O_TRUNC)) == -1)
    {
    printf("Unable to open %s\n", str);
    exit(1);
    }
```

```
/* fill out all the stuff that doesn't change */

buff[0] = 0;    /* line number information */
buff[1] = 0xFF;
buff[2] = 0xFF;
buff[3] = 0x64;
buff[4] = 0x80;
buff[5] = 0x10;

buff[8]  = 0xFF; /* start of video sync */
buff[9]  = 0;
buff[10] = 0;

buff[1452] = 0xFF; /* end of video sync */
buff[1453] = 0;
buff[1454] = 0;

/* fill the active video line with black */
video_data = buff + 12;
for(i=0; i<720; i++)
    {
    *video_data++ = 0x80;
    *video_data++ = 0x10;
    }
```

```
video_data = buff + 12;
line = 0;
for(i=0; i<BLANK_LINES; i++) /* Pad the start with the blanked lines */
    {
    putline(line++, buff);
    putline(line++, buff);
    }
for(i=0; i<FIELD_LEN; i++)
    {
    read(src_file, video_data, LINE_LEN);
    putline(line++, buff);
    read(src_file, video_data, LINE_LEN);
    putline(line++, buff);
    }
for(i=0; i<PAD_LINES; i++) /* Pad to the end with whole lines */
    {
    putline(line++, buff);
    putline(line++, buff);
    }
/* now put out the remaining bytes */
buff[11] = 0x80;
buff[1455] = 0x90;

buff[6] = ((line & 0x1F) << 1) | 0x80;
buff[7] = ((line & 0x3E0) >> 4) | 0x80;
write(field_1, buff, FRAC_LINE);
line++;
buff[11] = 0xC7;
buff[1455] = 0xDA;

buff[6] = ((line & 0x1F) << 1) | 0x80;
buff[7] = ((line & 0x3E0) >> 4) | 0x80;
write(field_2, buff, FRAC_LINE);
line++;
}
```

12. Bibliography

Related Abekas Documents

Abekas A64 External Control Protocol Manual
Abekas A60 Ethernet Manual

Digital Video Standards

CCIR 601

SMPTE Recommended Practice 125 - 1984
Society of Motion Picture and Television Engineers
862 Scarsdale Avenue
Scarsdale, NY 10583
(914) 472 6606

EBU Tech. 3246-E
EBU Parallel Interface for 625 line Digital Video Signals
August 83

SCSI Information

The A60/A64 SCSI Implementation is based on the ANSI Proposed Standard X3.131-198x Small Computer Systems Interface (SCSI) Currently available in draft form (for $30) from :

X3 Secretariat/CBEMA
311 First Street, N.W., Suite 500
Washington, D.C. 20001

The latest copy we have is :
X3T9.2/82-2 Rev 17B 12/16/85

Background information on the SCSI standard
Byte Magazine May 86 pp85-94 June 86 pp107-114

Kennedy 9612 Streaming Tape Drive
8124 SCSI Controller Application Notes

Television Standards

525 line NTSC color specification RS 170A

(Never officially published) RS 170 is the Original Monochrome standard, Industrial Electronics Tentative Standard 1 is intended to supplement it with Color information.

both are available from Electronic Industries Association
2001 Eye Street, N.W. Washington DC 20006 (202) 457-4966

625 line PAL colour specification
BBC Engineering Information Dept.
Broadcasting House
Portland Place
London W1A 1AA

Digital Filtering

Digital Signal Processing

Alan V Oppenheim, Ronald W Schafer
Prentice Hall 75

Programs for Digital Signal Processing

IEEE Publications

Abekas Video Systems, Inc.

A Carlton Company

Abekas Video Systems, Inc.
101 Galveston Drive
Redwood City,
CA 94063

Tel. 415·369·5111
Telex. 592712
Fax. 415·369·4777