A L B E R T S U P P O R T P A C K A G E

a product of

AMPERFAX

P.O. Box 600097 N.M.B., Florida 33160
(305) ~~935-0353~~
944-1477

Version 2.3 - August 1984

DISCLAIMER OF ALL WARRANTIES AND LIABILITIES

Amperfax makes no warranties, expressed or implied with respect to this manual or the software described in this manual -- its quality, performance, merchantability, or fitness for any particular purpose. Amperfax Albert computer support documentation, hardware, and software is sold "as is." The entire risk as to their quality and performance is with the buyer. **The only exception to this disclaimer is in the event of failure of the magnetic media upon which the software programs are recorded.** If such an event takes place within a period of 90 days following the initial purchase by the initial purchaser, Amperfax, at its discression, will replace the diskette and recorded software free of charge to the initial purchaser provided that the initial purchaser returns the damaged media to Amperfax at his/her own cost. **In no event shall Amperfax be liable for direct, indirect, incidental, or consequential damages resulting from a defect in this product, and/or the use of this product.**

Amperfax is an organization wishing to provide limited software support to purchasers of the ALBERT COMPUTER. Amperfax is in no way affiliated with ALBERT COMPUTERS, INC., CADENA ENGINEERING, and/or any affiliate of these organizations.

TABLE OF CONTENTS

SECTION VI

VBI SYSTEM
80 COLUMN DISPLAY
Z80 AND CP/M
DIAGNOSTIC DISK INTERFACE

SECTION VII

I/O ROM OPTIONS
SERIAL INTERFACE
PARALLEL INTERFACE
RS-422 INTERFACE

APPENDIX A

PATCHING APPLESOFT
KEYBOARD
DISK DRIVES
SELF-BOOTING DISKETTES
SOFTWARE AUTOMATIC MOUTH
COMPATIBILITY
PROGRAMMABLE ARRAY LOGIC
AFTERWORD

APPENDIX B

INTEGRATED CIRCUITS

APPENDIX C

READING REFERENCES
CHARACTER SET COMPARISION

* * *

INTRODUCTION

In 1983, Albert Computer, Inc. announced the unveiling of its Apple compatible Albert computer. Albert produced about 4000 prototype machines. If you own an Albert, the chances are it's one of these.

In early 1984, We purchased 5 prototype machines. None operated correctly, so we began our R & R procedure. Within 1 month, hardware problems were identified and either repaired or circumvented. We then began R & R from a software aspect, pulling and decoding eproms, so that we could gain insight into the workings of the machine. After many months of research and hard work, Amperfax is proud to be able to present this support package to the Albert public. **We hope that you will find it both useful and valuable as a tool for working with your Albert.**

When we set upon creating this package, we decided that it should touch on all aspects of the machine. Some discussions are extremely basic in nature, while others are extremely complex. Regardless of your reason for purchasing this package, there is something here for you. The potential now exists for the everyday user to make his/her computer work for them, rather than vice versa. Additionally, if you are already familiar with computers and have been held back from writing programs for the Albert because of a lack of information, this package is especially well suited for you. We know that each person has his/her own reasons for purchasing ASP, and we sincerely hope that the answers are contained herein.

After you have read through the documentation and run the software, we hope that you will use the enclosed feedback form. We are soliciting your help as an Albert user, and functioning as a focal point, we can help share your Albert experience with other users. If we accept your feedback for publication in future revisions of this package, you will, in exchange for your help, receive a copy of the updated package for free.

Thank you for your purchase.

ASP HARDWARE INSTALLATION

Check off each procedure as you complete it. **Move slowly** through each step, taking your time to insure nothing is overlooked. **Please observe anti-static handling precautions** when working with the integrated circuits. Refer to the peel-off circuit layout for locations of the various integrated circuits that need to be changed.

PLEASE NOTE THAT THE BOOT ROM & I/O ROM SOCKETS ARE LARGER THAN THE INTEGRATED CIRCUITS THEMSELVES. INSTALL THE NEW CHIPS SO THAT PIN 1 ON EACH IS IN THE THIRD HOLE ON THE LEFT, FROM THE REAR, VIEWING THE MACHINE FROM THE FRONT. OTHERWISE SERIOUS DAMAGE MAY RESULT.

STEP 1 (REQUIRED)

____ Switch off Albert power supply (if so equipped)
____ Remove top cover from main computer
____ Unplug battery backup (if so equipped)

STEP 2 (REQUIRED)

____ Locate and remove original Boot ROM (EPROM 1)
____ Insert Amperfax Boot ROM in its place (notch faces rear)
____ Confirm all pins are inserted correctly (none bent)
____ Put original Boot ROM in anti-static foam and put away

STEP 3 (OPTIONAL)

____ Locate and remove original I/O ROM (EPROM 2)
____ Insert Amperfax I/O ROM in its place (notch faces rear)
____ Confirm all pins are inserted correctly (none bent)
____ Put original I/O ROM in anti-static foam and put away

STEP 4 (OPTIONAL)

____ Locate and remove original Character Generator (EPROM 3)
____ Insert Amperfax CHAR GEN in its place (notch faces rear)
____ Confirm all pins are inserted correctly (none bent)

____ Put original Char. Gen. in anti-static foam and put away

STEP 5 (REQUIRED)

____ Check all three ROMS once again carefully
____ Reconnect battery backup (if so equipped)
____ Replace cover on main computer

STEP 6

____ Switch on video monitor
____Switch on computer and verify the following at the top of your screen.

ALBERT][

In the event you do not see the identification, **disconnect power to the machine remove the battery backup (if so equipped), and ask your dealer for assistance.**

* * *

OVERVIEW

In describing the Albert, it is difficult to determine where to begin. Perhaps the best place to start is with a description of what the machine actually is.

The described features of the machine are as follows:

VIDEO DISPLAY

Display modes:

* 40 column text, 5x7 dot matrix
* 80 column text, (optional)
* Low resolution color graphics
* High resolution color graphics
* Analog R.G.B. output
* NTSC Composite monitor output (black & white only)

Text capacity:

* 40 columns x 24 lines (standard)
* 80 columns x 24 lines (optional)

Character set:

* 96 printable ASCII characters upper and lower case

Display formats:

* Normal, Inverse, and Flashing
* Programmable text character and text background colors selected from a choice of 256 hues

Low resolution graphics:

* 16 colors at any one time from a choice of 256 hues
* 40h x 48v pixels
* 40h x 40v pixels with 4 lines of text

High resolution graphics:

* 4 colors with 280h x 192v pixels (160v with 4 lines of text)
* 6 colors with 140h x 192v pixels (160v with 4 lines of text)
* All colors selectable from a choice of 256 hues

MEMORY

* 64 kilobytes of dynamic RAM
* 2 on-board rom sockets (about 8 kilobytes)

CPU

* 6502 microprocessor
* Z-80 co-processor (optional)

INPUT/OUTPUT

* Detachable serial keyboard
* Microphone input
* Internal speaker
* External speaker jack
* Amplifier with internal volume control
* RS-232 Port
* RS-422/423 Networking port
* Centronics compatible parallel port
* 5 peripheral expansion slots
* Real-time clock
* Game ports (16 pin dip and serial type)
* A/D and D/A converters
* Battery backup (optional)
* Two 5.25" half-height disk drives
* **SOFTWARE AUTOMATIC MOUTH**
* **ARTSCI MAGIC SOFTWARE (complete series)**
* **S & H UTILITY DISKETTE (with licensed version of Applesoft)**
* Koala pad and micro-illustrator software

Not all of these features were advertised, but because of the packages that became available, most were combinations of the above.

The Albert is many computers rolled up into one. The software operating system is essentially Applesoft BASIC. Because it has the capability to display 80 columns of text with an alternate RAM card, it parallels an **Apple][e.** When we look closely at the RAM banking system, we see a direct similarity to the **Commodore 64.** The built-in peripheral system and co-processor ability is analagous to that found in the **BASIS 108.** Finally, the boot system is extremely similar to the type found in the new **Franklin CX Series Computers.**

In each case, the Albert essentially functions in a similar fashion to one or more of these machines, but the physical makeup bears very little similarity if any, at all.

Just one example of this difference is evident in the way the RAM can be write protected in both the Albert and the Franklin. Functionally similar, but the **total control scheme is entirely different.** That is why there is going to be some software incompatibility between these machines and the Albert.

Now we would like to address the most powerful feature of the Albert; a **RAM based operating system.** As described above, the system is very similar to that of the Commodore 64. If you are not familiar with that machine, then all you need to know is that essentially, you can use the BASIC POKE command (or its equivalent in machine) to set the Commodore up to be a full 64k of empty RAM. The Albert is the same way (although believe it or not, the Commodore is far more sophisticated), and because of this feature, it opens the door to some interesting possibilities. The first of these is the ability to **"patch"** or modify a version of Applesoft to suit your particular application. One example is in the situation of lower case. When you type "INT" after a cold-boot, the Albert actually installs a lower case patch into Applesoft. Actually, it does much more than this, but for now it is only important that you know the capability exists.

The second interesting possibility is for use in a laboratory, or video production room. Because the Albert's operating system (which is normally Applesoft) **can be "unlocked" and modified,** who is to say that you can't modify it to the point that it is no longer Applesoft at all, and make it an entirely new machine. In the lab situation, we can envision the Albert collecting data from its Analog to Digital converter, and then literally speaking

the results using the Software Automatic Mouth (SAM). In the video scenario, it is not difficult to see the Albert doing time coding or perhaps by using the R.G.B. output in conjunction with the control registers (discussed later), extremely sophisticated graphics could be achieved at a fraction of the cost of commercial systems. Now, we know there are alot of you that are ready to jump down my throat saying "Easier said than done", and of course you would be right, but nevertheless, **the capability or potential does exist within the Albert, where it would literally cost thousands of dollars to emulate it from another source.**

* * *

BOOTING PROCESS

When power is switched on to the Albert, the Boot ROM is engaged by one of the **PAL (Programmable Array Logic) chips.** There are many reasons for this. First, like any other computer, the Albert always needs to execute some type of program whenever the power is on. **Even when idling, any microprocessor is always executing a program.** Because the Albert is RAM based, there must be a starting program or initial entry point of some sort. That's one purpose for the **Boot ROM.**

Another reason for the Boot ROM is because the Albert has internal systems. These systems must be initialized to a known state. You see, because the Albert is RAM based, all of the functions like text color and the like are controlled by locations in memory. Moreover, because dynamic RAM (like the ones used in the Albert) don't hold data after the power is switched off, there is no way to predict what value these locations will be holding when power is switched on. **The Boot ROM takes care of this for you by storing values into these locations.** In this way, the machine is always initialized to a known state when the power is switched on. **Without it, the video display (amongst other things) would be a different color each time you turned the computer on.**

Even another reason for the Boot ROM has to do with the fact that the Albert is RAM based. The Albert idea is to emulate an Apple computer, and because the Apple is ROM based, any Apple type disk controller will require that certain key utility programs be available for use during a disk boot. **Not only does booting a disk require certain utility programs, but they must function identically to the Apple, and must even be located in the same place in memory as the Apple counterparts.** So you see, these routines have to be stored in the Albert somewhere, and once again, because the Albert is RAM based, the memory locations that normally hold these key utility programs are going to be empty when the computer is switched on. The method used to circumvent this problem is simple. **The key utility programs are stored on the Boot ROM, and the PAL chips make the RAM functionally switch places with it.** This concept is known as **BANK-SELECTING.** It is a **very important** concept in understanding the Albert.

So to review, you switch on your Albert and the Boot ROM is automatically banked in by a PAL chip. Next, the initialization program is executed and the machine is placed into a known or default state. Now that it's finished doing all that, it must look for a disk controller card in one of the slots. If it finds one, program execution continues with the ROM on the controller card and the disk drive switches on. If no card is found, the Albert will force a boot error.

I am sure you have all seen this at one time or another. Whenever it happens, the message **"BOOT UTILITY DISK"** is displayed, and the speaker beeps. We found this to be very annoying, and are sure you did too. That's one reason we wrote this package.

Now that you have a better understanding of essentially what the Boot ROM does right after a power-on, you may be asking yourself if this is all there is. Well, there's much more.

Let's assume that you've inserted the **S&H UTILITY DISK** into the drive, and booted it from a cold-start. Why does the machine load Applesoft instead of Integer? The reason is that the Albert is configured at power-on as **an old-style standard Apple][with Integer BASIC installed in ROM, and a 16k RAM card.** The **S&H UTILITY DISKETTE** is set up in such a way that it always loads the missing language into the 16k RAM card, and then begins execution of this missing language. In this case, the missing language is Applesoft, and the 16k card is the top portion of memory. Remember, the **Boot ROM looks like real Apple ROM to the S&H UTILITY DISK**, so it will switch off the ROM, and begin execution of Applesoft.

An interesting experiment you can try yourself to confirm the S&H method is to find an Apple][with Applesoft in ROM, and a 16k card in slot 0. Boot the Utility disk on it, and you will see the machine come up in Integer. The reason is because a system like that is the opposite of the Albert. Integer is missing, instead of Applesoft. To further confirm it, boot the Utility Disk on an Apple][with Integer in ROM and a 16k card in slot 0. Voila' instant Applesoft, just like the Albert. But wait, is it really just like the Albert? The answer is of course; yes and no. Read on.

* * *

THE "INT" COMMAND

As we have already explained, the Albert ends up with Applesoft in its RAM. If you have any Apple experience, you know that INT stands for Integer, and FP stands for Floating Point, or Applesoft. These are Apple DOS 3.3 commands, and they function identically in the Albert, except that the Albert has no true language ROM.

If you have an Apple][with Integer in ROM, and a 16k card in slot 0, boot the **S&H UTILITY DISK.** The machine comes up in Applesoft BASIC, as previously explained. Now type the word "INT", and the language will switch from Applesoft to Integer. Essentially, the RAM card holding the missing language (Applesoft) is switched off, and the ROMS holding Integer are switched on.

In the Albert, we know that the same is true with the exception of the last part. We can't turn on Integer ROMS with the "INT" command if we don't have them, so what happens ONCE per bootup session, is the Boot ROM is called one final time.

The routine that is executed is located exactly where DOS 3.3 expects Integer BASIC execution to begin. You know this routine, because it is the one that displays the boot options menu.

```
1-Boot DOS 3.3
2-Boot Pascal
3-Exit
```

Up until this point, the Albert is holding standard Applesoft. But once this menu appears, it is a signal that many other things have taken place. First of all, when you type INT, **the Boot ROM completely wipes out the cassette routines in Applesoft, and replaces them with a program that changes the color control registers** (discussed in detail later). **It also modifies the input routine in Applesoft to allow lower case input.** Next it sets another not so important hardware switch that **defeats your ability to display flashing and inverse lower case.** Finally, it takes the entire Boot menu displayed above, and moves it to low RAM. This is because as soon as you select 1-3, the Boot ROM

gets banked out of memory for good, and if the program weren't moved first, it would crash as soon as the selection was made.

So now you know how the entire boot system works. The only last point to mention here, is that when you make the selection, and the Boot ROM is finally moved out, **RAM replaces it.** That RAM is the Albert's real RAM card. It can be selected just like an Apple's. As a matter of fact, the only main difference between the first and second options in Albert's Boot Menu, is that PASCAL needs a RAM card. So truthfully, you could say that **option 1 actually sets up a 48k system, while option 2 sets up a 64k system.** Now, I know that you may be wondering about what happens when you boot up Applesoft, and switch to Integer. Well, you may already be aware that typing "FP" from Integer yields exactly the same result in the Albert as typing "INT" from Applesoft. The same programs are called on the Boot ROM. As one final note, once you have made the selection, **a repeated call to another language other than the one you are in will produce a "Language Not Available" error, because it just isn't there to switch to.**

* * *

OUR BOOT ROM

As you all know, Albert Computers never had a chance to finish the job they set out to do. In today's world, this is not an uncommon occurrence. Albert had a good idea, and they really tried their best to make it work. If you look at the machine from the engineer's point of view, you will see that alot of things were designed so that modifications could be made easily. One of these areas was the use of **Eraseable Programmable Read-Only Memory, or EPROM.**

The Boot ROM is just one out of four included in the Albert. The others are the I/O ROM, Character Generator ROM, and Keyboard Encoder ROM. This package contains special replacements for the first three only because the Keyboard Encoder ROM is ok as it is.

Looking at it from Albert's perspective, you can easily see that they were undoubtedly under a tremendous amount of pressure to complete their machines and get them to the marketplace. Therefore, it is certainly not unreasonable to assume that the programs written for the Boot ROM were basic in nature, just enough code to make the machines workable. **In many respects, it really is an interesting system**, and it certainly represented a challenge to figure out how it worked.

After finding out what had to be done, minimally, to make the machine operable, I set out on writing a completely new source code with the user in mind.

One of the first things incorporated into the ROM is intelligence. **Many prior problems that would have generated boot errors on the old system will now be handled by the appropriate routines.** Additionally, the main menu has been expanded to include the following:

1. Boot DOS 3.3
2. Boot PASCAL or CP/M
3. 48k System
4. 64k System
5. Diagnostics
6. Monitor

Each of these new boot options is described in the next section, so let's digress for a moment and go back to the concept of "INT", or the method of activating the menu as described in the last section.

As you know from reading, the old Boot ROM makes various modifications to the version of Applesoft that is loaded into the Albert's memory. Well, this is extremely convenient, except that it causes problems. For one thing, if you have found a way to call the diagnostics on the controller card, and you have run the ROM test, or have even booted any of the diagnostic software packages available, you undoutedly got a **checksum error** on some of the ROMS. Since **these ROMS are really RAM with a write-protect switch added,** many of you might be thinking that you have bad RAMS. This is not the case.

The Albert uses an interleaving method of RAM addressing that is different than the Apple's. **If any of the 4164 RAMs are bad in the machine, it won't work at all.** So where does this checksum error come from? Well, it's really quite simple. The **S&H UTILITY DISK** loads a standard version of Applesoft. The Boot ROM then modifies it for the Albert when you type "INT". **So, now the version of Applesoft is non-standard, and will not give the same checksum results when tested.** This also sets the stage for another problem.

Many programs, especially copy-protected versions with a special DOS on them will not boot under the old system. The reason for this is because commercial software must ascertain what machine it is working on. The typical method is to locate the value at memory location 57344 decimal ($E000 hexidecimal). If it is a 32 ($20), then the machine is in INTEGER BASIC. If it is a 76 ($4C), then the machine is in APPLESOFT BASIC. Additionally, if it can store a new value there after BANK-SELECTING, then it indicates the machine has a RAM card. The problem arises when a program attempts to store a value before the RAM is write-protected. It causes BASIC to cease proper operation. Moreover, even if your disk gets past this problem, you are still faced with the checksum error.

Much of today's software performs a checksum operation on the BASIC that is installed on the machine it is booting on. The reasoning may appear that whoever wrote it is just being nice by

running a quick diagnostic on your machine, but the real reason is because they don't want their program to crash unexpectedly. If a checksum error is detected, then the software just doesn't boot, and if it doesn't boot, then it never has the opportunity to crash.

So the first thing we made sure of, is that **our Boot ROM won't touch Applesoft at all**, and that the **RAM gets locked automatically on a DOS 3.3 boot.** This in itself will cut down on the number of unbootable disks that you might have, and will insure that diagnostics function properly.

The next idea we decided to address was the initialization of the machine. All of the Annunciators, color registers, and bank select registers are set to hold their proper values.

After that, we went further by examining the rationale of what else you would need. The "necessary" routines for booting have been expanded, so even more software with its own DOS will work on your machine. Additionally, you now have control over the machine at power-on. You can press (control-shift [) once, to fall into a simple monitor program, or you can press it twice to enter a simple assembler. You probably won't have any need for this capability, but it does allow you more flexibility than you had before. That's about it for the most part, so now let's discuss those new options.

* * *

NEW BOOT OPTIONS

The new boot options listed in the previous section are available to you when you select either "INT" from Applesoft BASIC, or "FP" from Integer BASIC. **The menu is available once per power-on.** When prompted for a selection, **only the answers 1-6 are considered valid.** Please note that **selection response is immediate,** so have that disk ready in the drive. As one last point to remember, all 48k options will lock RAM automatically. All 64k options must leave RAM unlocked to operate properly.

OPTION 1 - Boot DOS 3.3:

As mentioned before, this option is used primarily to boot a copy-protected disk or a standard DOS 3.3 disk.

OPTION 2 - Boot Pascal or CP/M:

This option will allow you to boot PASCAL directly. **CP/M is now also available if you have a compatible Z-80 card installed.** Unfortunately, the system will not operate properly with some cards, so **be sure to read the section on Z-80 and CP/M before attempting to use the system or before making any purchases.** This option also sets up a 64k system with a RAM card available. You may also use this with certain copy-protected programs such as **APPLEWRITER][** by Apple Computer to make use of the additional space.

OPTION 3 - 48k System:

You may be wondering why this option is included. The reasoning behind it is that you may just want to lock the RAM in place, and continue working with your system. After all, you already have Applesoft installed along with DOS, so why boot another disk? After selecting this, you may insert diskettes and go directly to work.

OPTION 4 - 64k System:

This option was created for the same reason as option 3, but it makes the RAM card available for you to work with. Use standard Apple commands to bank the RAM card in and out of the system.

OPTION 5 - Diagnostics:

This option will call the diagnostics on the disk interface card. **If you are using a standard interface, then avoid using this option**, or the system will have to be rebooted.

OPTION 6 - Monitor:

This final option is **equivalent to a BASIC "Call - 151".** It is included as a matter of convenience for those of you who wish to make custom modifications to the resident BASIC prior to locking the RAM. If you use this option, don't forget to set the RAM locking register (discussed in detail later) to the protected state.

* * *

CONTROL REGISTER THEORY

In the previous section, I broached upon the concept of a control register. Simply put, **a control register is a location in the memory of the computer that controls the operation of whatever it is "mapped" or connected to.** In the Albert there are many of these control registers. Some have just one specific purpose, and others are multi-purpose.

When using these control registers, all you need to do is place a number between 0 and 255 to cause a change in the operation of the computer. **From either Integer or Floating Point BASIC,** you can do this with the **POKE** command, where the format would be as follows:

(Line Number) POKE (REGISTER MEMORY ADDRESS), (VALUE 0-255)

In this example, you can see that the statement starts with a line number. This is only necessary if you wish to change a control register from a BASIC program. The reason this is pointed out, is because **the line number is optional when dealing with control registers.** This means that you can enter the POKE command from the **immediate mode** also. For those of you who understand this, please bear with me. If you don't fully uunderstand, then let me explain.

The BASIC language is simply a big machine level program that is constantly running. You can cause things to happen in BASIC from one of two modes. **The first is known as the "DEFERRED EXECUTION" mode.** This simply means that you preceed your BASIC statements with a line number, and when you type "RUN", the computer will THEN execute them. **The second is known as the "IMMEDIATE EXECUTION" mode.** It simply means that if you type a certain BASIC statement without a line number, then the computer will immediately execute the instruction you have given it. In the BASIC language, there are many instructions of which some are meant to be used in one mode, and some in another. POKE is one type of instruction that can be used from either mode. If you still haven't caught up, don't worry. Try reading the **APPLESOFT TUTORIAL** by Apple Computer for a good understanding of BASIC, and then come back to this.

Now, to continue with control registers, it is important to note, that these are special locations in memory that are known as **write-only registers.** These are relatively new (past few years) to the computer world. They first made their appearance on the Apple-scene, when they started being incorporated into 80 column cards. The reasoning behind it was that the use of 80 columns on the Apple or its compatibles, sometimes gets complicated by a mismatch between the display monitor and the computer. So what manufacturers started doing, was to incorporate these control registers into the 80 column cards so that people like you or me could place numbers between 0 and 255 into these registers, and actually change the operation of the 80 column card to suit our particular hardware. On an 80 column card, they essentially provide you with the ability to change the display using a program, much as you would by turning one of the little knobs in the back of a television to provide a better picture.

Now, imagine this concept installed into your Albert, **but instead of using them to match your computer and display monitor, they are now used to select on-board peripherals like your clock, RS-232, RS-422, Digital to Analog converter, RGB color control, 80 column auxillary RAM card, and much more.**

As previously mentioned, each of these is a write-only register, which means that you can only place a number there, but you can't read the SAME number back. That is one limitation of this type of system, but certainly one that is easy to live with.

Now, I stated at the beginning of this topic that some of these registers have just one purpose, and others have more than one. The reason for this is because there is unfortunately, not enough space in the Albert to give each peripheral individual control registers in different areas in memory. The way this problem was circumvented was by taking each peripheral that required more than one or two registers, and by using the BANK-SELECTING method described earlier to essentially "OVERLAY" them into the same areas. Then, all that would be needed is a special purpose "DEVICE-SELECT" register, whose setting would determine which of the peripherals was to be "MAPPED IN" for use.

In the Albert, the main control registers are located in a group of 16 sequential areas in memory. These are outlined in the **FUNCTIONAL ADDRESS INDEX** at the end of the book, and are also discussed in greater detail later. It is not necessary that you

understand which register performs which function right now; rather, **it is much more important that you only understand the basic concept.** Now we will look at **BANK SELECTION.**

* * *

BANK SELECTION EXPLAINED

You were first introduced to bank-selection when I told you the Albert "banks" the Boot-ROM in and out of the system. Additionally, when I described the theory behind control registers, I briefly mentioned how the Albert goes one step further, by "mapping" three known peripherals into the Albert's memory. Each of the peripherals is either "Banked-in" or "Banked-out", depending upon the setting of what I term the **DEVICE-SELECT REGISTERS.**

There are 4 device-select registers in the Albert, and **to select a certain peripheral requires storing a number in two of these at a time.** Therefore, the Albert may select up to six on-board peripherals; although, only three are used at this time (to our knowledge).

These three peripherals are:

1) R.G.B. color control system
2) Digital to Analog converter
3) Real-Time clock

As stated, each of these peripherals may be "Banked" in or out by you easily. If you are confused; don't worry. **It is only important that you recognize how each of these peripherals is controlled.** This method is detailed on the following page.

* * *

DEVICE-SELECT REGISTERS

As stated in the previous section, the Albert uses device-select registers to control the bank-selection of the various on-board peripherals. **These bank-select registers are nothing more than dedicated areas in memory where you place a number to switch the peripherals on and off.** These locations are as follows:

NAME OF REGISTER	DECIMAL ADDRESS	HEXIDECIMAL ADDRESS
Device-select 1	49254	$C066
Device-select 2	49255	$C067
Device-select 3	49256	$C068
Device-select 4	49257	$C069

When the machine is powered-on, **the device-select registers default to the RGB color control system.** We know that turning on each of the three peripherals is simply a matter of storing a value into the proper registers, so here are all the combinations:

COMBINATION	PERIPHERAL
49254 (C066) 49255 (C067)	UNKNOWN/UNUSED
49254 (C066) 49256 (C068)	RGB COLOR CONTROL (DEFAULT SETTING)
49254 (C066) 49257 (C069)	DIGITAL TO ANALOG CONVERTER
49255 (C067) 49256 (C068)	REAL-TIME CLOCK
49255 (C067) 49257 (C069)	UNKNOWN/UNUSED
49256 (C068) 49257 (C069)	UNKNOWN/UNUSED

The method for switching on a desired peripheral is this:

```
FROM BASIC:        POKE (ADDRESS 1), 0
                   POKE (ADDRESS 2), 0
```

Where ADDRESS 1 and ADDRESS 2 are the two **DECIMAL** locations listed as a combination for the desired peripheral.

```
FROM ASSEMBLER:    LDA #$00
                   STA ADDRESS 1
                   STA ADDRESS 2
                   RTS
```

Where ADDRESS 1 and ADDRESS 2 are the two **HEXIDECIMAL** locations listed as a combination for the desired peripheral.

The method is really simple, and perhaps the only thing you need to remember is that **switching a peripheral on always turns off any peripheral that was previously selected.** Therefore, you can't have SAM changing colors and speaking at the same time.

Once any of these peripherals has been switched on, 16 additional locations become available for use. These locations contain what I call the **CONTROL REGISTERS** (previously mentioned), and they are the places in memory from which you may literally control each peripheral; hence, the term "CONTROL REGISTERS."

The control registers are described under each of the individual peripherals, because they may be the same locations, but they do have different functions. The first of these peripherals is the **RGB COLOR CONTROL SYSTEM**, and it is probably the most fun to use, so read on.

* * *

RGB COLOR CONTROL

The Albert's RGB Color Control system is easy to use. Using the following method, you can bank-in the RGB Color Control, which will make 16 control registers available to you. **These 16 locations can hold values from 0 to 255 decimal, or $00 to $FF hexidecimal.** These values represent the color that the Albert will display in each of the modifiable modes of operation.

Although the RGB Color Control system is by default; banked-in, it will undoubtedly be necessary in the future for you to select it manually. The general method is as follows:

FROM BASIC:

```
10 POKE 49254,0
20 POKE 49256,0
```

FROM ASSEMBLER:

```
LDA #$00
STA $C066
STA $C068
```

Both programs serve the same purpose, and once run, will insure the RGB Color Control system is available. As stated, you will have a choice from between 0 and 255 for colors. It is important to note a few points here. **First, not all 255 values are different colors. There are only some "pure" colors with the balance being variations in intensities.** This is really not a limitation at all, as you will see when you begin to work with the system. The second point to note is that the following page lists all the possible colors and their corresponding values. **Please note that these values are not only subjective, but also depend heavily upon the sensitivity of the RGB monitor being used.** Therefore, use them as a general guide.

* * *

ALBERT COLOR SELECTION GUIDE

VALUE	COLOR	VALUE	COLOR	VALUE	COLOR
000	WHITE	008	PINK	016	PINK
001	BLUE	009	VIOLET	017	VIOLET
002	BLUE	010	BLUE	018	VIOLET
003	BLUE	011	BLUE	019	BLUE
004	BLUE	012	BLUE	020	BLUE
005	BLUE	013	BLUE	021	BLUE
006	BLUE	014	BLUE	022	BLUE
007	BLUE	015	BLUE	023	BLUE
024	PINK	032	PINK	040	PINK
025	VIOLET	033	VIOLET	041	VIOLET
026	VIOLET	034	VIOLET	042	VIOLET
027	BLUE	035	BLUE	043	BLUE
028	BLUE	036	BLUE	044	BLUE
029	BLUE	037	BLUE	045	BLUE
030	BLUE	038	BLUE	046	BLUE
031	BLUE	039	BLUE	047	BLUE
048	PINK	056	PINK	064	YELLOW
049	VIOLET	057	VIOLET	065	GREEN
050	VIOLET	058	VIOLET	066	GREEN
051	BLUE	059	BLUE	067	GREEN
052	BLUE	060	BLUE	068	GREEN
053	BLUE	061	BLUE	069	GREEN
054	BLUE	062	BLUE	070	GREEN
055	BLUE	063	BLUE	071	GREEN
072	YELLOW	080	ORANGE	088	RED
073	GREEN	081	ORANGE	089	RED
074	GREEN	082	GREEN	090	RED
075	GREEN	083	GREEN	091	PURPLE
076	GREEN	084	GREEN	092	BLUE
077	GREEN	085	GREEN	093	BLUE
078	GREEN	086	GREEN	094	BLUE
079	GREEN	087	GREEN	095	BLUE

096	RED	104	RED	112	RED
097	RED	105	RED	113	RED
098	RED	106	RED	114	RED
099	PURPLE	107	PURPLE	115	PURPLE
100	BLUE	108	BLUE	116	BLUE
101	BLUE	109	BLUE	117	BLUE
102	BLUE	110	BLUE	118	BLUE
103	BLUE	111	BLUE	119	BLUE
120	RED	128	GREEN	136	YELLOW
121	RED	129	GREEN	137	GREEN
122	RED	130	GREEN	138	GREEN
123	PURPLE	131	GREEN	139	GREEN
124	BLUE	132	GREEN	140	GREEN
125	BLUE	133	GREEN	141	GREEN
126	BLUE	134	GREEN	142	GREEN
127	BLUE	135	GREEN	143	GREEN
144	ORANGE	152	ORANGE	160	ORANGE
145	ORANGE	153	ORANGE	161	ORANGE
146	GREEN	154	ORANGE	162	ORANGE
147	GREEN	155	GREEN	163	ORANGE
148	GREEN	156	GREEN	164	GREEN
149	GREEN	157	GREEN	165	GREEN
150	GREEN	158	GREEN	166	GREEN
151	GREEN	159	GREEN	167	GREEN
168	ORANGE	176	ORANGE	184	ORANGE
169	ORANGE	177	ORANGE	185	ORANGE
170	ORANGE	178	ORANGE	186	ORANGE
171	ORANGE	179	ORANGE	187	ORANGE
172	ORANGE	180	ORANGE	188	ORANGE
173	BLUE	181	PURPLE	189	PURPLE
174	BLUE	182	BLUE	190	BLUE
175	BLUE	183	BLUE	191	BLUE
192	GREEN	200	YELLOW	208	ORANGE
193	GREEN	201	GREEN	209	YELLOW
194	GREEN	202	GREEN	210	GREEN
195	GREEN	203	GREEN	211	GREEN
196	GREEN	204	GREEN	212	GREEN
197	GREEN	205	GREEN	213	GREEN
198	GREEN	206	GREEN	214	GREEN
199	GREEN	207	GREEN	215	GREEN

216	ORANGE	224	ORANGE	232	ORANGE
217	ORANGE	225	ORANGE	233	ORANGE
218	YELLOW	226	ORANGE	234	ORANGE
219	GREEN	227	YELLOW	235	ORANGE
220	GREEN	228	GREEN	236	ORANGE
221	GREEN	229	GREEN	237	GREEN
222	GREEN	230	GREEN	238	GREEN
223	GREEN	231	GREEN	239	GREEN

240	ORANGE	248	ORANGE
241	ORANGE	249	ORANGE
242	ORANGE	250	ORANGE
243	ORANGE	251	ORANGE
244	ORANGE	252	ORANGE
245	ORANGE	253	ORANGE
246	BLACK	254	BLACK
247	BLACK	255	BLACK

TEXT CHARACTER COLOR

One of the interesting features of the Albert is the ability to change the color of the text characters. When you turn on the machine, the text characters should appear white.

Once you have banked-in the RGB color control as described in the previous section, you may proceed to modify the text character color. There are 4 control registers out of 16 dedicated to text character color. Each register may contain a value between 0 and 255 decimal, or $00 to $FF hexidecimal. Here they are:

NAME	DECIMAL LOCATION	HEXIDECIMAL LOCATION	DEFAULT VALUE
TC-1	49186	$C022	0 ($00) White
TC-2	49190	$C026	0 ($00)
TC-3	49194	$C02A	0 ($00)
TC-4	49198	$C02E	0 ($00)

Of the four registers available, two are known as **main registers,** and the remaining two are known as **auxillary registers.** The main registers control the color of the text characters, while the auxillary registers take care of a timing problem in the display hardware. This timing problem is evident in many different modes. Please note that because there are main and auxillary registers, I will term them **MAIN** and **SYNC** respectively for the sake of clarity. SYNC was coined, because the timing problem is a syncronization mismatch in nature. It is not necessary that you understand how this problem comes about; rather, just note that **if you change the MAIN character color, it would be wise to change the SYNC to the same color.** Now let's try an example program, so you have a better understanding of how to do it.

Boot the **S&H UTILITY DISK**, and get into BASIC.

Now type in the following program:

```
 10 POKE 49254,0 : REM BANK-IN RGB CONTROL
 20 POKE 49256,0 : REM BANK-IN RGB CONTROL
 30 C = 160 : GOSUB 100
 40 HOME : PRINT "THIS IS ORANGE TEXT"
 50 GET A$ : PRINT A$
 60 C = 0 : GOSUB 100
 70 HOME : PRINT "THIS IS WHITE TEXT"
 80 GET A$ : PRINT A$
 90 GOTO 30
100 POKE 49186,C : REM SET TC-1
110 POKE 49190,C : REM SET TC-2
120 POKE 49194,C : REM SET TC-3
130 POKE 49198,C : REM SET TC-4
140 RETURN
```

This program will alternate the text color betweeen ORANGE (160), and WHITE (0), each time you press a key. You may experiment with different values for C in line 30. Each value can be chosen from between 0 and 255; but remember, some of these values will be of different intensities, and may appear as BLACK on your monitor. If you lose readability due to the color, just press [CONTROL] [HELP] when the text is white. This will force a system reset while you can still see the text characters, thus making it easier for you to place a new value in line 30. Experiment for a while, and if you have BASIC experience, fiddle around with the program.

As you have seen, text character color can be changed with just a few lines of a BASIC program. It really is that simple. Here's a much simpler **assembler** program that will change the characters to either WHITE or ORANGE, depending on the call.

```
  DS1 EQU $C066   ; DEVICE-SELECT 1
  DS3 EQU $C068   ; DEVICE-SELECT 3
  TC1 EQU $C022   ; MAIN CHARACTER COLOR REGISTER 1
  TC2 EQU $C026   ; MAIN CHARACTER COLOR REGISTER 2
  TC3 EQU $C02A   ; SYNC CHARACTER COLOR REGISTER 3
  TC4 EQU $C02E   ; SYNC CHARACTER COLOR REGISTER 4
SETUP LDA #$00    ; LOAD A WITH ANY VALUE
      STA DS1     ; BANK-IN RGB CONTROL
      STA DS3     ; BANK-IN RGB CONTROL
      RTS         ; RETURN TO CALLER
ORANGE LDA #$A0   ; PUT VALUE FOR ORANGE IN A
```

```
        JSR SETREG ; SUBROUTINE TO SET COLOR
        RTS        ; RETURN TO CALLER
WHITE   LDA #$00   ; PUT VALUE FOR WHITE IN A
        JSR SETREG ; SUBROUTINE TO SET COLOR
        RTS        ; RETURN TO CALLER
SETREG  STA TC1    ; SET MAIN REGISTER 1
        STA TC2    ; SET MAIN REGISTER 2
        STA TC3    ; SET SYNC REGISTER 3
        STA TC4    ; SET SYNC REGISTER 4
        RTS        ; RETURN TO MAIN PROGRAM
```

Once you assemble this program, you first call **SETUP** to insure the RGB Control is banked-in. Then you may call ORANGE or WHITE at will. Other values may be substituted for ORANGE (A0), and you may wish to expand the program even more. In any event it is a very basic assembler program, whose purpose is for instruction only. The software support disk has oodles of goodies, so later you may wish to try them, but now please read through the balance of the color system documentation so you will gain familiarity with it.

* * *

TEXT BACKGROUND COLOR

The next logical discussion in the area of the Albert's RGB Color Control system is that of text background color. The text background color has 4 registers just like the text character color, and they are used in identical fashion. They are:

NAME	DECIMAL LOCATION	HEXIDECIMAL LOCATION	DEFAULT VALUE
TB-1	49184	$C020	255 ($FF) Black
TB-2	49188	$C024	255 ($FF)
TB-3	49192	$C028	255 ($FF)
TB-4	49196	$C02C	255 ($FF)

As you can see, because these registers default at power-up to holding 255, the equivalent color is BLACK. You may change the background to display any color you wish, and for instructional reasons, let's do that now. This program is a little more complex than the last one, but it does more.

FROM BASIC TYPE:

```
10 POKE 49254,0 : REM BANK-IN RGB CONTROL
20 POKE 49256,0 : REM BANK-IN RGB CONTROL
30 TC = 255 : TB = 160 : GOSUB 110
40 HOME
50 PRINT "THIS IS BLACK ON ORANGE"
60 GET A$ : PRINT A$
70 TC = 160 : TB = 255 : GOSUB 110
80 HOME
90 PRINT "THIS IS ORANGE ON BLACK"
100 GET A$ : PRINT A$ : GOTO 30
110 POKE 49186,TC : POKE 49190,TC
120 POKE 49194,TC : POKE 49198,TC
130 POKE 49184,TB : POKE 49188,TB
140 POKE 49192,TB : POKE 49196,TB
150 RETURN
```

When you run this program, **pressing any key will alternate the text and background colors between BLACK and ORANGE.** Again, you are encouraged to experiment with the program by substituting values for T(ext) C(olor) and T(ext) B(ackground). Use [CONTROL] [HELP] to stop the program.

Here is a similar program in **assembler**, that doesn't require a key to be pressed.

```
     DS1 EQU $C066    ; DEVICE-SELECT 1
     DS3 EQU $C068    ; DEVICE-SELECT 3
     TC1 EQU $C022    ; MAIN CHARACTER COLOR REGISTER 1
     TC2 EQU $C026    ; MAIN CHARACTER COLOR REGISTER 2
     TC3 EQU $C02A    ; SYNC CHARACTER COLOR REGISTER 3
     TC4 EQU $C02E    ; SYNC CHARACTER COLOR REGISTER 4
     TB1 EQU $C020    ; MAIN BACKGROUND COLOR REGISTER 1
     TB2 EQU $C024    ; MAIN BACKGROUND COLOR REGISTER 2
     TB3 EQU $C028    ; SYNC BACKGROUND COLOR REGISTER 3
     TB4 EQU $C02C    ; SYNC BACKGROUND COLOR REGISTER 4
    HOME EQU $FC58    ; ROM ROUTINE TO CLEAR TEXT SCREEN
    WAIT EQU $FCA8    ; ROM ROUTINE TO CAUSE A DELAY
    COUT EQU $FDED    ; ROM ROUTINE TO OUTPUT A CHARACTER
    TEMP EPZ $FF      ; TEMPORARY STORAGE LOCATION
         JSR HOME     ; CLEAR THE SCREEN
   SETUP LDA #$00
         STA DS1      ; BANK-IN RGB CONTROL
         STA DS3      ; BANK-IN RGB CONTROL
         STA TEMP     ; START AT ASCII CODE 0
    LOOP LDA TEMP     ; GET CURRENT ASCII CODE
         CMP #$FF     ; IS IT THE LAST ONE?
         BEQ READY    ; IF YES THEN START COLOR SWITCH
         INC TEMP     ; INCREMENT ASCII CODE
         JSR COUT     ; OUTPUT ASCII TO SCREEN
         JMP LOOP     ; GO BACK FOR MORE
   READY JSR COUT     ; OUTPUT LAST CHARACTER
 SCREEN1 LDA #$A0     ; PUT VALUE FOR ORANGE IN A
         JSR SETBACK  ; MAKE BACKGROUND ORANGE
         LDA #$FF     ; PUT VALUE FOR BLACK IN A
         JSR SETTEXT  ; MAKE TEXT BLACK
         LDA #$40     ; PUT DELAY VALUE IN A
         JSR WAIT     ; DELAY
 SCREEN2 LDA #$FF     ; PUT VALUE FOR BLACK IN A
         JSR SETBACK  ; MAKE BACKGROUND BLACK
         LDA #$A0     ; PUT VALUE FOR ORANGE IN A
         JSR SETTEXT  ; MAKE TEXT ORANGE
         LDA #$40     ; PUT DELAY VALUE IN A
         JSR WAIT     ; DELAY
         JMP SCREEN1  ; DO IT AGAIN
 SETBACK STA TB1      ; SET MAIN BACKGROUND REGISTER 1
         STA TB2      ; SET MAIN BACKGROUND REGISTER 2
```

```
        STA TB3     ; SET SYNC BACKGROUND REGISTER 3
        STA TB4     ; SET SYNC BACKGROUND REGISTER 4
        RTS         ; RETURN TO CALLER
SETTEXT STA TC1     ; SET MAIN TEXT REGISTER 1
        STA TC2     ; SET MAIN TEXT REGISTER 2
        STA TC3     ; SET SYNC TEXT REGISTER 3
        STA TC4     ; SET SYNC TEXT REGISTER 4
        RTS         ; RETURN TO CALLER
```

This program will display the entire ASCII set at the top of the screen, and then proceed to alternate the text character and background colors between ORANGE and BLACK. Again, you are encouraged to modify this source code to experiment with the system. You will have to press [CONTROL] [HELP] to exit. Now we will proceed onto **HIGH RESOLUTION COLORS.**

* * *

HIGH RESOLUTION COLORS

When the Albert is powered-on, the ROM 2.0 automatically places the correct values for Apple][color emulation into the High Resolution Color registers. There are 4 registers available, and for color, there is no sync required except in the background mode discussed in the next section. The 4 registers are:

NAME	DECIMAL LOCATION	HEXIDECIMAL LOCATION	DEFAULT VALUE
HC-1	49187	$C023	207 ($CF) GREEN
HC-2	49191	$C027	026 ($1A) PURPLE
HC-3	49195	$C02B	145 ($91) ORANGE
HC-4	49199	$C02F	023 ($17) BLUE

Now, in the overview, you read that **the Albert can display up to six colors.** Here we have only four, so where are the other two? Well, this takes a little bit of explaining.

Due to hardware reasons, the Apple can display four colors plus two whites, and two blacks. Therefore, Applesoft is configured to use one of these colors during an HPLOT. If you don't understand this, then please read the **"APPLESOFT TUTORIAL"** by Apple Computer. To continue, we know the Apple breaks down these four basic colors into two groups.

Group 1 colors are composed of: HCOLOR 0 = Black 1
HCOLOR 1 = Green
HCOLOR 2 = Violet or Purple
HCOLOR 3 = White 1

Group 2 colors are composed of: HCOLOR 4 = Black 2
HCOLOR 5 = Orange
HCOLOR 6 = Blue
HCOLOR 7 = White 2

Now imagine that White 1 is a combination or sum of Green and Violet, and White 2 is a combination or sum of Orange and Blue.

In other words,

HCOLOR 3 = HCOLOR 1 + HCOLOR 2
and
HCOLOR 7 = HCOLOR 5 + HCOLOR 6

Now, some of you Apple Vets might be a little skeptical about this system; but indeed, it is the method by which the Albert gets the two additional colors. Because i'm sure seeing is believing, let's try a quick demonstration.

Get into BASIC, and place the **SOFTWARE PROGRAMS #1 DISKETTE** in drive 1. Then type the following:

```
HGR
BLOAD PICTR.HIRES COLORS,A$2000
```

You should now see the standard Apple colors on your screen. From left to right they are;

Black 1
Green
Violet (or purple, depends on monitor)
White 1
Black 2
Orange
Blue
White 2

We know that the control register for HCOLOR 1 (green), is located at 49187 decimal, or $C023 hexidecimal. If my theory is correct, changing HCOLOR 1's register to display ORANGE (160 or $A0) will result in a change in the color of HCOLOR 3 or White 1. Let's try it.

Type: **POKE 49187,160**

See, I told you so. Now experiment with each of the registers to see what colors you can get out of your machine. If you want to restore the original colors, either place the default values listed at the beginning of this section in the proper registers, or you can type;

BRUN NORMAL HIRES

This program will reset all the registers to their power-up values.

There are a couple of points to note when you're using these registers. The first is that the **Albert ALWAYS displays NORMAL APPLE-TYPE COLORS through its hardware system.** The change that takes place **only modifies the DISPLAY INFORMATION** through the RGB Color Control System. Therefore, if you make a picture with special colors you've selected, save the picture out to the disk, turn the machine off, and then reboot and load the picture; **THE PICTURE WILL APPEAR IN NORMAL APPLE COLORS.** The Albert **NEVER** modifies the actual data in a picture. This takes a little getting used to, but it is a fun system to work with.

The second point to remember here, is that under the old Boot-ROM, you could press [CONTROL] [SHIFT] [and the colors would toggle between all white and color. This feature has been removed under **ROM 2.0** due to compatibility problems with the diagnostics as described earlier, but it can be re-installed simply by running the program **"COLOR TOGGLE."** If you are using the **MAGIC SERIES** of software with the **excellent 70 column HIRES driver,** you will want this feature. It makes the HIRES characters appear solid white, and really aids readability. The method for installing the color toggle is by typing;

BRUN COLOR TOGGLE

and then boot your **MAGIC SERIES** diskette. Operation of the toggle works just as it did before.

Another method of achieving a similar result, without modifying Applesoft is by simply putting zero in each of the HIRES control registers. The method of doing it goes like this:

FROM BASIC:

```
10 POKE 49187,0
20 POKE 49191,0
30 POKE 49195,0
40 POKE 49199,0
```

FROM ASSEMBLER:

```
LDA #$00
STA $C023
STA $C027
STA $C02B
STA $C02F
```

You may also want to either patch the **MAGIC SERIES**, or include the BASIC listing in a greeting program on a disk that you can use as a preboot. In any event, you can see how simple it is to change the HIRES colors, and of what value it can be to you. There is presently no known Apple Compatible computer with this feature. Now we will discuss another Albert gift; **HIGH RESOLUTION BACKGROUND.**

* * *

HIGH RESOLUTION BACKGROUND COLORS

The high resolution background system is very similar to the text background system. Here again, we have two main registers, and two sync registers. The sync registers are used to eliminate the problems generated when other than black is used for the background.

NAME	DECIMAL LOCATION	HEXIDECIMAL LOCATION	DEFAULT VALUE
HB-1	49185	$C021	255 ($FF) Black
HB-2	49189	$C025	255 ($FF)
HB-3	49193	$C029	255 ($FF)
HB-4	40197	$C02D	255 ($FF)

The reason the default value is set to 255, is because the background of the HIRES screen is normally equal to black.

When you modify these registers, you must use the same value in all four. Otherwise, you will experience strange video effects. Also, if you are going to be in the HGR mode using text at the bottom of the screen, **good practice dictates changing the text background registers to the same color as the high resolution background registers. This will avoid sync problems between the two display screens.** Another thought to be aware of is the hires background's effect on the plotting colors.

When you change the background color, it will affect the display of each HCOLOR. The only values that will not interfere, are black and white. To illustrate the point, power-on the machine, get into BASIC, and with the **SOFTWARE PROGRAMS #1 DISKETTE** in drive 1 type the following:

```
HGR
BLOAD PICTR.HIRES COLORS,A$2000
BRUN INVERT HIRES
```

Now the screen should be white, with a minimum of influence on the HCOLORS. To return the screen, type:

```
BRUN NORMAL HIRES
```

As you can see, this system won't be used as much as the High Resolution Color system, but it is there, and someday you may have a need for it. As described in the previous section, the program COLOR TOGGLE is available. For background changes between Black and White, we have included other programs with the name **TOGGLE** at the end to indicate how they work. Each is designed to operate with the display mentioned in the program name. They all will alternate the related display between NORMAL and INVERSE modes by using [CONTROL] [SHIFT] [. Now let's go onto the final topic in this section, **LOW RESOLUTION GRAPHICS.**

* * *

LOW RESOLUTION COLOR & BACKGROUND

I saved this section for last because it does introduce some complex ideas. We can now broach it if you have a firm foundation in control register theory.

Low resolution graphics is available via the GR statement. When executed, some soft-switches (Page 13 Apple Reference Manual) are thrown, and **the screen is cleared to BLACK.** However, on the Albert under ROM 2.0, **the screen will be BLUE.** The reason for this is really very simple.

As you know, **ROM 2.0** sets the color registers when the machine is powered on. It **installs the correct colors for both the text and high resolution display screens, but not the low resolution display screen.**

Because Applesoft is structured to handle sixteen lores colors (0-15), the Albert needs sixteen registers to control these colors. Since **there are only sixteen color control registers available, the Albert must use all of these for low resolution graphics.** Therefore, all low resolution color control registers conflict with the text and high resolution displays.

This problem is evident when, as previously stated, the GR command results in a BLUE screen on the Albert, rather than a BLACK screen as on an Apple. **This particular conflict arises out of the fact that the HC-4 (high resolution color) register has been loaded with the value for BLUE by ROM 2.0, but HC-4 is the same register that is used for LC-0 (low resolution color 0).** This means that to avoid conflicts between displays, the color control registers should be loaded with the appropriate values first.

DISPLAY REGISTER USAGE COMPARISON

DEC LOC	HEX LOC	TEXT USAGE	HIRES USAGE	LORES USAGE
49184	C020	TB-1	--	LC-15
49185	C021	--	HB-1	LC-14
49186	C022	TC-1	--	LC-13
49187	C023	--	HC-1	LC-12
49188	C024	TB-2	--	LC-11
49189	C025	--	HB-2	LC-10
49190	C026	TC-2	--	LC-09
49191	C027	--	HC-2	LC-08
49192	C028	TB-3	--	LC-07
49193	C029	--	HB-3	LC-06
49194	C02A	TC-3	--	LC-05
49195	C02B	--	HC-3	LC-04
49196	C02C	TB-4	--	LC-03
49197	C02D	--	HB-4	LC-02
49198	C02E	TC-4	--	LC-01
49199	C02F	--	HC-4	LC-00

Again, as you can see by examining this figure, **although both text and hires registers are staggered, lores does conflict with each.**

The following page contains a list of the recommended values for each control register, in order to produce Apple-compatible colors while preserving the standard white/black text screen display.

APPLE COMPATIBLE LOW RESOLUTION CONTROL REGISTER SETTINGS

REGISTER	COLOR	DEC VAL	HEX VAL	APPLE COLOR
LC-15	BLACK	255	FF	WHITE
LC-14	ORANGE	160	A0	AQUAMARINE
LC-13	WHITE	0	00	YELLOW
LC-12	LT GREEN	134	86	LT GREEN
LC-11	BLACK	255	FF	PINK
LC-10	YELLOW	136	88	GREY 2
LC-09	WHITE	0	00	ORANGE
LC-08	BROWN	154	9A	BROWN
LC-07	LT BLUE	7	07	LT BLUE
LC-06	MD BLUE	23	17	MD BLUE
LC-05	GREY	18	12	GREY 1
LC-04	DK GREEN	150	96	DK GREEN
LC-03	PINK	56	38	PURPLE
LC-02	DK BLUE	39	27	DK BLUE
LC-01	MAGENTA	186	BA	MAGENTA
LC-00	BLACK	255	FF	BLACK

If you want to sacrifice text colors, you may use either of these two methods to yield more colors. These are in no way inclusive of all the possibilities.

METHOD 1 - LOW RESOLUTION GRAPHICS WITH 4 LINES OF TEXT ADDS TWO COLORS

a. Substitute LC-15 and LC-11 with any number from 0-255.
b. Store the same number in LC-0 (avoids sync problem).
c. Store any one number from 0-255 in both LC-13 and LC-09.
d. Limit PLOT to 0-39 vertical and 0-39 horizontal.

* Make sure that LC-15/11 and LC-13/09 are holding values that differ enough to make text readable. Here you would essentially be changing text colors 1 + 2, and text background colors 1 + 2. You must make LC-0 equal to the text background color, so that the area holding the text at the bottom of the screen is the same color as the low resolution background (color 0); otherwise, you will get poor results.

The following method is a little less conservative, yet it can open the door to interesting color displays.

METHOD 2 - LOW RESOLUTION GRAPHICS NO TEXT
ADDS FIVE COLORS AND EIGHT MORE PLOT LINES

a. Place any number 0-255 in LC-15.
b. in LC-13.
c. in LC-11.
d. in LC-09.
e. in LC-00.
f. Execute a POKE 49234,0.
g. Limit PLOT to 0-47 vertical and 0-39 horizontal.

With this method, the four lines of text at the bottom of the screen will be available as an additional eight plotting lines. The only point to remember is that Applesoft will not normally clear these eight lines to the background color, so when working with standard Applesoft make sure your first program statements (after POKE 49234,0) clear these lines to the background color. This can be accomplished with the following BASIC statements:

```
COLOR = 0
FOR ROW = 38 TO 47
HLIN 0,39 AT ROW
NEXT ROW
```

It is also possible to modify Applesoft to do this for you. This topic is covered in the section on **PATCHING APPLESOFT.**

As one final note on LOW RESOLUTION GRAPHICS, as with all of the systems, you are encouraged to experiment and develop new ideas. Furthermore, if you don't know how to use the graphic commands, read the book entitled **"APPLESOFT TUTORIAL"** by Apple Computer.

Well, that's it for graphics, we hope this has helped you. Now on to Section V and the **INTERNAL PERIPHERALS.**

* * *

REAL TIME CLOCK

The Real Time Clock functions like the other peripherals in all but one respect. The clock is the only one that sends data back through the control registers. Essentially, it makes the control registers bi-directional, because you set the clock by writing to them, and you find the time by reading from them. As with the other peripherals, the clock must be banked-in first, so here's the method;

FROM BASIC:

```
POKE 49255,0
POKE 49256,0
```

FROM ASSEMBLER:

```
LDA #$00
STA $C067
STA $C068
```

Once the clock is banked into the system, we can begin our work. One bit in the register location 49191 ($C027) is the clock switch. If clear (or equal to 0), then the clock is off. If set (or equal to 1), then the clock is running. There are other registers used, and for the sake of simplicity, they are outlined for you here;

DECIMAL LOCATION	HEXIDECIMAL LOCATION	FUNCTION
49186	$C022	SECOND
49187	$C023	MINUTE
49188	$C024	HOUR
49189	$C025	DAY
49190	$C026	MONTH
49191	$C027	SWITCH

The registers can be loaded with the proper values after two conditions have been met:

a. The clock is banked-in
b. The clock is turned off

You already know how to do the first, so here's the second;

```
FROM BASIC:        TO SWITCH CLOCK OFF:  POKE 49191,3
                   TO SWITCH CLOCK ON :  POKE 49191,7

FROM ASSEMBLER:    TO SWITCH CLOCK OFF:  LDA #$03
                                         STA $C027

                   TO SWITCH CLOCK ON :  LDA #$07
                                         STA $C027
```

When the clock is off, simply put the proper value into the appropriate register. You really don't have to worry about this, because the Software Programs Diskette has a utility that will allow you to set the clock. If you want to read the time, use the following program. Note the difference between slots in the Albert version vs the Amperfax version, or your program may hang.

IF ORIGINAL ALBERT I/O ROM - SLOT = 1
IF AMPERFAX I/O ROM - SLOT = 4

```
10  D$ = CHR$ (04) : REM DOS CONTROL D
20  PRINT D$; "IN#" ; SLOT
30  INPUT " ";A$
40  PRINT D$  "IN#0"
50  PRINT "The present time is: ";A$
60  END
```

It is easy to see that this program may be modified in many ways. We included a program called MENU.FP on the Software Programs Disk. If you make a copy of your S&H Utility Disk, replace their MENU.FP with ours, and put SAM onto that disk, you will have a very interesting greeting program.

Now, just one last note on the clock: Many people have been told that this clock emulates the Mountain Computer Clock. **This is NOT true.** The format in which the time is printed IS Mountain Computer format, but the drivers on the I/O ROM (regardless of manufacturer) is not the same as Mountain's. Therefore, a program like SUPERCAT on the S&H Utility Disk will NOT recognize the Albert's clock as a Mountain Computer clock.

* * *

ANALOG TO DIGITAL CONVERTER

The Analog to Digital converter port contains various control inputs and outputs. These are outlined as follows:

DESCRIPTION	PIN
+5 VOLTS DC	01
AUXILLARY PUSHBUTTON 1	02
ANNUNCIATOR 3 OUTPUT	03
ANNUNCIATOR 1 OUTPUT	04
FRAME GROUND	05
ANALOG INPUT CHANNEL 4	06
ANALOG INPUT CHANNEL 0	07
ANALOG INPUT CHANNEL 2	08
ANALOG INPUT CHANNEL 1	09
ANALOG INPUT CHANNEL 3	10
ANALOG INPUT CHANNEL 5	11
ANNUNCIATOR 0 OUTPUT	12
ANNUNCIATOR 2 OUTPUT	13
AUXILLARY PUSHBUTTON 2	14
AUXILLARY PUSHBUTTON 0	15

AUXILLARY PUSHBUTTON STATUS AND OPERATION

Each of the Auxillary pushbuttons has a dedicated location in memory where their status may be read from. These locations are as follows:

BUTTON	DECIMAL LOCATION	HEXIDECIMAL LOCATION
AUXILLARY BUTTON 0	49257	$C069
AUXILLARY BUTTON 1	49258	$C06A
AUXILLARY BUTTON 2	49259	$C06B

Pushbuttons are activated in the same manner as the type used by Apple Game Control Paddles. The status is determined by the following values:

SET = 255 CLEAR = 127

The following is a simple BASIC program to read the status of each of the auxillary pushbuttons:

```
10 A = PEEK (49257)
20 B = PEEK (49258)
30 C = PEEK (49259)
40 IF A > 127 THEN PRINT "BUTTON 0 IS SET"
50 IF B > 127 THEN PRINT "BUTTON 1 IS SET"
60 IF C > 127 THEN PRINT "BUTTON 2 IS SET"
70 GOTO 10
```

ANNUNCIATOR OPERATION

There are four annunciator outputs. Each will drive a standard Transistor-Transistor Logic (TTL) load. If you are going to control relays or devices without TTL loads, these outputs must be fully buffered. These annunciators are low voltage switches, and can be controlled via the following locations:

ANNUNCIATOR	SWITCH ON LOCATION	SWITCH OFF LOCATION
0	49241 / $C059	49240 / $C058
1	49243 / $C05B	49242 / $C05A
2	49245 / $C05D	49244 / $C05C
3	49247 / $C05F	49246 / $C05E

It is important to note that just referencing these locations will cause a change in annunciator status. This example program should give you a start. It is included on the supplied diskettes, so **you don't have to type it in.**

```
10 REM ANNUNCIATOR SELECT UTILITY
20 TEXT : HOME
30 FLAG = 0
40 A$ = "ANNUNNCIATOR SELECT UTILITY"
50 GOSUB 1000 : POKE 34,1
60 FOR I = 0 TO 3 : AN$(I) = "OFF" : NEXT I
70 LO(0) = 49240 : REM $C058
71 HI(0) = 49241 : REM $C059
72 LO(1) = 49242 : REM $C05A
73 HI(1) = 49243 : REM $C05B
74 LO(2) = 49244 : REM $C05C
```

```
 75 HI(2) = 49245 : REM $C05D
 76 LO(3) = 49246 : REM $C05E
 77 HI(3) = 49247 : REM $C05F
 80 FOR I = 0 TO 3 : POKE LO(I),0 : NEXT I
 90 IF FLAG = 1 THEN VTAB 20 : HTAB 10 : PRINT "PRESS ANY KEY
    TO STOP"
100 VTAB 3
110 A$ = "ANNUNCIATOR STATUS" : GOSUB 1000
120 A$ = "------------------" : GOSUB 1000
130 FOR I = 0 TO 3 : HTAB 10
140 PRINT "Annunciator ";I;" is ";AN$(I);" "
150 NEXT I
160 IF FLAG = 1 THEN RETURN
170 HTAB 20 : PRINT "or"
180 HTAB 9 : PRINT "(S)equential exerciser"
190 HTAB 9 : PRINT "(E)xit the program"
200 PRINT : PRINT
210 VTAB 13 : HTAB 9
220 PRINT "Press 0,1,2,3,S, or E ";
230 GET A$ : PRINT A$
240 IF A$ = "E" THEN END
250 IF A$ = "S" THEN GOTO 700
260 IF A$ = "0" THEN GOTO 300
270 IF A$ = "1" THEN GOTO 400
280 IF A$ = "2" THEN GOTO 500
290 IF A$ = "3" THEN GOTO 600
295 GOTO 210
300 IF AN$(0) = "Off" THEN AN$(0) = "On" : POKE HI(0),0 :
    GOTO 90
310 IF AN$(0) = "On" THEN AN$(0) = "Off" : POKE LO(0),0 :
    GOTO 90
400 IF AN$(1) = "Off" THEN AN$(1) = "On" : POKE HI(1),0 :
    GOTO 90
410 IF AN$(1) = "On" THEN AN$(1) = "Off" : POKE LO(1),0 :
    GOTO 90
500 IF AN$(2) = "Off" THEN AN$(2) = "On" : POKE HI(2),0 :
    GOTO 90
510 IF AN$(2) = "On" THEN AN$(2) = "Off" : POKE LO(2),0 :
    GOTO 90
600 IF AN$(3) = "Off" THEN AN$(3) = "On" : POKE HI(3),0 :
    GOTO 90
610 IF AN$(3) = "On" THEN AN$(3) = "Off" : POKE LO(3),0 :
    GOTO 90
700 FLAG = 1
```

```
 710 GOSUB 740
 720 IF PEEK (-16384) > 128 THEN FLAG = 0 : HOME : GOTO 90
 730 GOTO 710
 740 FOR L = 0 TO 3
 750 FOR D = 1 TO PDL(0) : NEXT D
 760 POKE LO(L),0 : AN$(L) = "Off"
 770 GOSUB 90
 780 NEXT L
 790 FOR L = 0 TO 3
 800 FOR D = 1 TO PDL(0) : NEXT D
 810 POKE HI(L),0 : AN$(L) = "On"
 820 GOSUB 90
 830 NEXT L
 840 RETURN
 999 END
1000 HTAB(40-LEN(A$))/2 : PRINT A$ : RETURN
```

ANALOG INPUTS

There are six analog inputs. They can be read from the following locations:

CHANNEL	DECIMAL LOCATIONS	HEXIDECIMAL LOCATIONS
0	49264 or 49272	$C070 or $C078
1	49265 49273	$C071 $C079
2	49266 49274	$C072 $C07A
3	49267 49275	$C073 $C07B
4	49268 49276	$C074 $C07C
5	49269 49277	$C075 $C07D

As you can see, there are two locations from which to read the information from. Each is equally valid, and is a function of the Albert's hardware scheme. If you wish to read these inputs, you must first trigger a "countdown" timer as described in the Apple reference manual on page 99. After starting the countdown, you must delay at least 110 microseconds to obtain an accurrate reading. Retrieved values are in the range of 0 - 255, with 0 representing 0 volts, and 255 representing 5 volts. **DO NOT EXCEED 5 VOLTS OR DAMAGE WILL RESULT.** Here is an example program:

```
 10 HOME
 20 A = 49264
 30 POKE A,0 : REM START TIMER
 40 FOR D = 1 TO 5 : NEXT D : REQUIRED DELAY
 50 VTAB 1
 60 FOR I = 0 TO 5 : REM INDIVIDUAL PINS
 70 V = PEEK (A + I)
 80 PRINT "INPUT ";I;" = ";V;"    "
 90 NEXT I : REM READ NEXT PIN
100 FOR D = 1 TO 500 : NEXT D : REM DELAY SCREEN
110 GOTO 50 : START NEW CONVERSIONS
```

* * *

THE VBI SYSTEM

The VBI or Vertical Blanking Interval System, allows the Albert to display more than 1 screen of information at a time.

The way that this system works is related to the operation of what is known as a **RASTER SCAN VIDEO SYSTEM.** All televisions as well as standard computer monitors, are based upon this system. **A basic understanding of the video display is required to understand the VBI,** so I will discuss it briefly for the benefit of beginners.

The screen area of a television or monitor is divided up into many small dots covered with a phosphorous coating on the inside. A beam of electrons "shoots" toward this phosphorous coating, and when it hits one of these small dots, you see the dot "light" up. Now, imagine that this beam can be moved to any part of the display screen, and that it can also be varied in intensity. When this beam is moved at a high speed across the screen, so fast in fact, that your eyes cannot follow it, then you have a television picture. The electron beam moves across the screen, starting from the top, and moves down the screen a line at a time. When it reaches the bottom of the screen, the beam is turned off, and sent back up to the top of the screen so it can begin drawing a new frame. This moment of turning off the beam, and returning it to the top of the screen is known as the **VERTICAL BLANKING INTERVAL** or VBI. Your Albert has the ability to tell your programs when this VBI occurrs via a special register in memory, which is located at 49256 decimal or $C068 hexidecimal.

You may be wondering why the VBI register is so important. Its real value lies in the ability to display **FLICKER-FREE** graphic displays. Many Apple games use animation. The way most of the present high-quality arcade type games are done, is by using the two graphic screens; HGR and HG2. The method typically involves drawing a picture on screen 1, displaying it, and then doing the same thing on screen 2. This **PAGE-FLIPPING** method provides for extremely smooth animation, but there is a bit of flicker on the display when the pages are switched when the electron beam is on. **The solution to this is the VBI register.** It was not available

on the Apple][, but it was implemented on the Apple][e, and as you are now aware, on the Albert as well. Its real beauty lies in the ability to sense the moment when the beam is turned off, and at that moment, we switch a screen; therefore, when the beam is turned on again, it is going to draw the screen with the new information. This process of drawing the screen is extremely fast, and if we want, we can expand on this idea to emulate a dual display system. **Essentially, this would allow us to simulate a text screen overlayed upon either HGR or HGR2, or even the GR screen.** The method involves drawing your graphics on one screen, text on the other, and flipping them when the beam is off. This will give the illusion of a dual-purpose display screen. I know this all sounds wonderful, but there is one drawback. Flipping the screens on the Albert brings out a small timing problem in the display system. **Superimposing two screens will display EVEN HORIZONTAL LINES on one screen, and ODD HORIZONTAL LINES on the other.** It is a limiting factor, but certainly something that can be lived with by designing displays that work with it instead of against it. The typical method in which page flipping is achieved goes like this, and can ONLY be done properly from assembler:

```
        ORG $0300
  VBI   EQU $C068
LOOPA   BIT VBI       ; TEST FOR N = 0 OR POSITIVE
        BPL LOOPA     ; IF YES THEN LOOP BACK
LOOPB   BIT VBI       ; TEST FOR N = 1 OR NEGATIVE
        BMI LOOPB     ; IF YES THEN LOOP BACK
        STA SCREENX   ; TOGGLE DESIRED SCREEN
        RTS           ; RETURN TO CALLER OR PLACE
                      ; DUPLICATE CODE TO TOGGLE
                      ; ANOTHER SCREEN HERE.
```

Here, we can see that the N Flag in the 6502 is the key. Testing the value that is read from $C068 will indicate the status of the electron beam.

You should experiment with this system, but please note that timing is critical here, so any changes will cause flicker or "rolling" of the display screens.

* * *

80 COLUMN DISPLAY

The 80 column text display on the Albert is similar to that found on the Apple][e. It uses a control register to switch between the 40 and 80 column modes. To access this system, you may use either of the BASIC or ASSEMBLER programs listed here. I suggest that you do try this system now.

```
FROM BASIC:       POKE 49165,0 : REM SWITCHES ON 80 COLUMN MODE

                  POKE 49164,0 : REM SWITCHES ON 40 COLUMN MODE

FROM ASSEMBLER:   LDA #$00     ; PUT ANY NUMBER IN A
                  STA $C00D    ; SWITCH ON 80 COLUMN MODE

                  LDA #$00     ; PUT ANY NUMBER IN A
                  STA $C00C    ; SWITCH ON 40 COLUMN MODE
```

Now, after you have selected the 80 column mode by using one of these methods, you will see each character on the screen printed TWICE. The Albert is now sending double the video information to the RAM area that is mapped to the text screen, just as it does on the Apple][e. The only problem that crops up here is that **of all the machines produced, only .1 of them have the additional 80 COLUMN TEXT DISPLAY CARD that is required to handle the 80 column display.** This card is the one that "piggybacks" on the connectors surrounding the motherboard RAM chips.

Now this is very important. **If your machine doesn't have the card installed, don't go out and buy the one for the Apple][e. It won't work in the Albert. The Apple][e card uses a different RAM addressing scheme, and additionally, the card connectors are different.**

There are some Albert Video RAM cards available, but we decided not to purchase them. The reasoning is as follows:

PROBLEM: If we purchased these cards, you would be paying MORE than what a Videx compatible 80 column card would cost. This is a function of the people selling them; not Amperfax.

ANALYSIS: Amperfax does highly-specialized consulting and we would never recommend a purchase that did not give the owner equity. Our reasoning is simple. Producing 80 columns is a very desirable feature in a computer, but if your machine doesn't have the feature completely built in, then it makes no sense to purchase more hardware that can only be used on that same machine, unless it is a vital part of the computer itself. 80 column displays on the Albert are not vital. Therefore, you must protect yourself, and any subsequent purchase, by insuring that each item is fully APPLE COMPATIBLE. Then if you ever sell your Albert, the 80 column card can be sold seperately; hence, SALVAGE VALUE. If you purchase the ALBERT-SPECIFIC card, then you will be forced to sell it with the computer, and unfortunately, without an additional price for the card, because people just won't pay for it.

SOLUTION: Amperfax can sell REGISTERED OWNERS OF ASP Videx type 80 column cards that will work in ANY Albert or Apple compatible computer, at a very low price. This will give you equity, and should be the most attractive solution if you need this feature.

* * *

Z-80 AND CP/M

This is another area that is just as touchy as the 80 column display. The Albert does have the ability to use an alternate Z-80 processor. The one Albert intended to produce was a small piggyback card that was meant to sit on top of the 6502. That's why there are connector pins surrounding the motherboard microprocessor. Unfortunately, **these are not available, nor are schematics describing the wiring modifications. Therefore we propose the following alternative:**

Just as in the case of the 80 column card, Amperfax can sell REGISTERED OWNERS OF ASP the **PCPI APPLICARD** with a Z-80B processor working at 6 megahertz. This card is special, because when you are not using CP/M, you may use the **ADDITIONAL 64K of RAM on the PCPI card as a RAMDISK.** Additionally, this card supports ANY video display (40 or 80), and comes with **the same type of 70 column driver that the MAGIC SERIES of software does.** Therefore, if you purchase this card, **you will not need an 80 column card.**

It is important to note here that our choice of the PCPI card lies in the fact that yes, **there are many low-cost MICROSOFT type cards available, but they will NOT work with the Albert because they must use the Albert's motherboard RAM and DMA line.** Additionally, **with the PCPI there is not preboot required, and you get CP/M and utilities with each package.**

Regardless of whom you purchase your card from, this card is the one that we recommend you use.

* * *

DIAGNOSTIC DISK INTERFACE

Undoubtedly, some of you with computer experience have recognized what the Albert's built-in diagnostics really are. If you haven't, then read on.

The diagnostic progam is contained on the EPROM located roughly in the center of the disk interface card. The card says Albert Computer on it; however, it bears a great similarity to the Prometheus Applesurance card.

The card will work in an Apple][if the diagnostic programs are exchanged between the Albert version and the Apple][version.

To call the diagnostics in the Albert, from other than the new boot menu:

FROM BASIC: CALL 50742

FROM ASSEMBLER: JSR $C636

This assumes that your disk interface is installed in slot 6.

Please note that checksum errors will be detected if you have modified BASIC in any way prior to executing the diagnostics.

* * *

I/O ROM OPTIONS

The original Albert I/O ROM creates the following configuration:

```
PR#1 = PARALLEL PORT OUTPUT
IN#1 = CLOCK INPUT
PR#4 = SERIAL PORT & RS 422 OUTPUT
IN#4 = SERIAL PORT & RS 422 INPUT
```

The new Amperfax I/O ROM creates the following configuration:

```
PR#1 = SERIAL PORT & RS 422 OUTPUT
IN#1 = SERIAL PORT & RS 422 INPUT
PR#4 = PARALLEL PORT OUTPUT
IN#4 = CLOCK INPUT
```

As you can see, which ROM you use is a matter of preference. If you are using a parallel printer, then you would prefer staying with the original ROM, but with a serial printer you may prefer having it located in slot 1. In any event, you now have a choice of whichever system your needs require.

* * *

SERIAL INTERFACE

The serial interface on the Albert can be used to drive most any serial device, including printers and modems. The signals that are available are as follows:

SIGNAL	PIN
DATA OUT	02
DATA IN	03
REQUEST TO SEND	04
CLEAR TO SEND	05
FRAME GROUND	07
DATA TERMINAL READY	20
RxC or TxC I/O	24

When using a serial device, refer to the manufacturer's instructions as to which line goes to which pin. They can usually be connected simply by matching the signal names of the pins on both the computer and device regardless of whether or not you actually understand what each signal does. If you have problems hooking a device up, take both to a local computer store. **In any event, insure the FRAME GROUND is connected.**

Once the device is connected, the serial port can be initialized with the standard Apple IN# x and PR #x, where "x" is equal to 1 when using the Amperfax I/O ROM, and 4 when using the original Albert I/O ROM. Once initialized, the serial port may be addressed as follows:

MAIN OPERATING COMMANDS

KEYPRESS		RESULTS
[CONTROL A] [CONTROL X]	will	CANCEL RS-232 INPUT
[CONTROL A] [CONTROL F]		ENABLE FULL DUPLEX MODE
[CONTROL A] [CONTROL H]		ENABLE HALF DUPLEX MODE
[CONTROL A] [CONTROL S]		SEND A BREAK AND PAUSE FOR KEYPRESS

BAUD RATE COMMANDS

KEYPRESS		BAUDRATE
[CONTROL A] >	will set	00050
[CONTROL A] 0		00075
[CONTROL A] 1		00110
[CONTROL A] 2		00135
[CONTROL A] 3		00150
[CONTROL A] 4		00300
[CONTROL A] 5		00600
[CONTROL A] 6		01200
[CONTROL A] 7		01800
[CONTROL A] 8		02400
[CONTROL A] 9		03600
[CONTROL A] :		04800
[CONTROL A] ;		07200
[CONTROL A] <		09600
[CONTROL A] =		19200

When using these commands, type the control key and the "A" key together prior to the next character. This is known as a "LEAD IN CHARACTER." Here are more commands when using the serial port from a remote terminal:

REMOTE COMMANDS

[CONTROL R] Sets remote control mode.
When this command is issued from a remote terminal, the Albert will take all input from the system as if it were from the keyboard, and it will output to it as if it were the display.

[CONTROL T] Sets terminal mode.
When this command is issued from a remote terminal, it will restore control to the Albert. If this is issued without a preceeding IN#0, the Albert will set itself to half-duplex.

The RS-232 is addressable from both BASIC and ASSEMBLER. The following information will aid you in customizing a program for use with the Albert's serial port. When using the serial port, it may be used as previously described, or directly accessed as follows.

The Albert's serial port is a 6551 Asynchronous Communication Interface Adapter (ACIA). It can be directly addressed through four registers in memory. This section deals with the RS232 in great detail. You may never need it, but it's here.

REGISTER	DECIMAL LOCATION	HEXIDECIMAL LOCATION
DATA	49344	$C0C0
STATUS	49345	$C0C1
COMMAND	49346	$C0C2
CONTROL	49347	$C0C3

The DATA register is where information is transmitted from and recieved to.

The STATUS register is used to inform your programs as to the condition of communications (described in detail later).

The COMMAND register is used to control parity checking and generation, interrupt enabling, and the RS-232 handshaking signals.

The CONTROL register is used to control the baud rate, word length, and the number of stop bits.

The following pages contain information on the STATUS, COMMAND, and CONTROL registers on the bit level.

RS-232 STATUS REGISTER

PROGRAMMING MODEL

BIT NUMBER 7 - IRQ REGISTER	0 = NO INTERRUPT 1 = INTERRUPT HAS OCCURRED
BIT NUMBER 6 - DATA SET READY	0 = DATA READY (LOW) 1 = DATA NOT READY (HIGH)
BIT NUMBER 5 - CARRIER DETECT	0 = CARRIER DETECTED (LOW) 1 = NO CARRIER (HIGH)
BIT NUMBER 4 - TRANSMIT DATA	0 = DATA REGISTER NOT EMPTY 1 = DATA REGISTER EMPTY
BIT NUMBER 3 - RECIEVER DATA	0 = DATA REGISTER NOT FULL 1 = DATA REGISTER FULL
BIT NUMBER 2 - OVERRUN	0 = NO OVERRUN 1 = OVERRUN HAS OCCURRED
BIT NUMBER 1 - FRAMING ERROR	0 = NO FRAMING ERROR 1 = FRAMING ERROR DETECTED
BIT NUMBER 0 - PARITY ERROR	0 = NO PARITY ERROR 1 = PARITY ERROR DETECTED

Here it is important to note no interrupt will occur for OVERRUN, FRAMING, and PARITY.

	7 6 5 4 3 2 1 0
HARDWARE RESET	0 - - 1 0 0 0 0
SOFTWARE RESET	- - - - - 0 - -

RS-232 COMMAND REGISTER

PROGRAMMING MODEL

BIT NUMBERS 7 6 5 - PARITY CHECK CONTROLS

OPERATION	BIT 7	6	5
PARITY DISABLED - NO PARITY BIT GENERATED - NO PARITY BIT RECEIVED	-	-	0
ODD PARITY RECEIVER AND TRANSMITTER	-	-	1
EVEN PARITY RECEIVER AND TRANSMITTER	0	1	1
MARK PARITY BIT TRANSMITTED PARITY CHECK DISABLED	1	0	1
SPACE PARITY BIT TRANSMITTED PARITY CHECK DISABLED	1	1	1

BIT 4 - NORMAL / ECHO MODE CONTROL

0 = NORMAL
1 = ECHO

BITS 3 and 2 TRANSMITTER CONTROL

TRANSMIT INTERRUPT	RTS LEVEL	OTHER	BIT 3	2
DISABLED	HIGH	-	0	0
ENABLED	LOW	-	0	1
DISABLED	LOW	-	1	0
ENABLED	LOW	TRANSMIT BREAK	1	1

BIT NUMBER 1 - RECEIVER INTERRUPT ENABLE

IRQ INTERRUPT ENABLED FROM BIT 7 OF STATUS REGISTER = 0
IRQ INTERRUPT DISABLED = 1

BIT NUMBER 0 - DATA TERMINAL READY

DISABLE RECEIVER/TRANSMITTER (DTR HIGH) = 0
ENABLE RECEIVER/TRANSMITTER (DTR LOW) = 1

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	0	0	0	0	0	1	0
SOFTWARE RESET	-	-	-	0	0	0	1	0

THIS SPACE INTENTIONALLY BLANK

RS-232 CONTROL REGISTER

BIT NUMBER 7 - STOP BITS

1 STOP BIT = 0
2 STOP BITS = 1 IF WORD LENGTH = 8 BITS PLUS PARITY
= 1.5 IF WORD LENGTH = 5 BIT WITHOUT PARITY

BITS 5 AND 6 - WORD LENGTH

WORD LENGTH IN BITS	5	6
8	0	0
7	1	0
6	0	1
5	1	1

BIT 4 - RECEIVER CLOCK SOURCE

EXTERNAL RECEIVER CLOCK = 0
BAUD RATE GENERATOR = 1

THIS SPACE INTENTIONALLY BLANK

BITS 0 thru 3 - BAUD RATE GENERATOR

BAUD RATE	3	2	1	0
16x EXTERNAL CLOCK BAUD	0	0	0	0
50.00	0	0	0	1
75.00	0	0	1	0
109.92	0	0	1	1
134.58	0	1	0	0
150.00	0	1	0	1
300.00	0	1	1	0
600.00	0	1	1	1
1200.00	1	0	0	0
1800.00	1	0	0	1
2400.00	1	0	1	0
3600.00	1	0	1	1
4800.00	1	1	0	0
7200.00	1	1	0	1
9600.00	1	1	1	0
19200.00	1	1	1	1

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	0	0	0	0	0	0	0
SOFTWARE RESET	-	-	-	-	-	-	-	-

PARALLEL INTERFACE

The parallel port uses standard centronics protocol, and installation of various printers is both easy and straightforward. The cable that is used to connect your Albert to your printer should be a ribbon cable no longer than a few feet, or you will lose data. The cable should have the following connectors and pin connections:

PRINTER END = AMP 522-931-1 ALBERT END = AMP 499-577-7

PIN CONFIGURATION

SIGNAL	PIN NUMBER
D0	03
D1	05
D2	07
D3	09
D4	11
D5	13
D6	15
D7	17
NEGATIVE STROBE	01
NEGATIVE OR POSITIVE ACK	19
POSITIVE BUSY	21
NO CONNECTION	23,25,26

ALL REMAINING PINS SHOULD BE GROUNDED

The parallel port is initialized just as with the serial port, by using IN# x and PR# x, where "x" is equal to 4 with the Amperfax I/O ROM, and 1 with the original Albert I/O ROM. These are the commands available to you after initialization. Please note the use of a LEAD IN CHARACTER as described in the preceeding section.

MAIN OPERATING COMMANDS

KEYPRESS		RESULTS
[CONTROL I] A	will	INSERT LINE FEEDS AFTER CARRAIGE RETURNS
[CONTROL I] B		SEND A BELL CHARACTER TO DEVICE
[CONTROL I] C		MASK OUT BELL CHARACTERS
[CONTROL I] H		ALLOW THE MOST SIGNIFICANT BIT TO BE PASSED TO THE DEVICE - MAY ENABLE GRAPHICS ON SOME PRINTERS
[CONTROL I] K		DISABLE LINE FEEDS AFTER CARRAIGE RETURNS
[CONTROL I] [CONTROL]?		THIS COMMAND WILL SET THE LEAD IN CHARACTER TO WHATEVER CHARACTER YOU USE IN THE PLACE OF THE QUESTION MARK USE ANY LETTER FROM A-Z UNDER DOS 3.3 **NEVER EXECUTE THIS COMMAND IN PASCAL.**

The following is a list of printers known to work with the Albert and their associated settings. If you can add to this list, please do.

COMPATIBLE PRINTERS AND RELATED SETTINGS

CANON COLOR A-1210: NO PROBLEM - PLUG COMPATIBLE

RADIO SHACK COLOR INK JET 26-1268: NO PROBLEM - PLUG COMPATIBLE

ANADEX 9500 SERIES: ALL SWITCHES ON S-2 SHOULD BE OFF. SET S-1 AS REQUIRED.

CENTRONICS 739-01 : ALL 4 SWITCHES SHOULD BE TURNED OFF.

EPSON MX70 : CONNECT SHORTING PLUG TO RED POST.

EPSON MX80 : ALL SWITCHES ON S-2 SHOULD BE OFF.

GEMINI : DIP SWITCH S-1 #3 SHOULD BE OFF.
#8 ON.

EPSON MX-100 : DIP SWITCH S-2 #3 SHOULD BE OFF.
S-1 #6 OFF.
S-1 #8 ON.

IDS 440 & 445 : DIP SWITCH S-4 #5 SHOULD BE OFF.
S-4 #6 OFF.
S-4 #7 OFF.
S-3 #6 ON.
S-3 #7 OFF.

STRAP SIGNAL POLARITY AS FOLLOWS:

9-6 BUSY ACTIVE HIGH
11-5 POSITIVE DATA STROBE
14-1 ALWAYS REMAINS INSTALLED

IDS 460 & 560 : DIP SWITCH S-4 #5 SHOULD BE ON.
S-4 #7 OFF.
S-3 #6 OFF.
S-3 #7 OFF.

STRAP PRINTER INTERNALLY AS FOLLOWS:

8-7 INSTALLED FOR PARALLEL OPERATION
11-4 BUSY RISING HIGH
12-3 LOW GOING STROBE

CITOH PROWRITER : DIP SWITCH S-1 #2 SHOULD BE ON.
S-1 #5 ON.
S-2 #6 OFF.
S-2 #7 OFF.

OKIDATA M82 : SET ALL PANEL DIP SWITCHES TO OFF.

In each case, these settings should help to get your printer and computer talking to each other. If your particular printer has switches not mentioned here, i.e. character spacing, then set it for whatever works best.

RS-422 INTERFACE

The RS-422 interface port is configured as follows:

PIN	DESCRIPTION
1	REQUEST TO SEND ($\overline{\text{RTS}}$)
2	DATA TERMINAL READY (DTR)
3	CLEAR TO SEND (CTS)
4	DATA IN/OUT
5	DATA IN/OUT
6	REQUEST TO SEND (RTS)
7	DATA TERMINAL READY ($\overline{\text{DTR}}$)
8	CLEAR TO SEND ($\overline{\text{CTS}}$)

To use the networking interface, two Alberts should be connected via these ports. Each of the units can communicate using the SERIAL PORT COMMANDS. The RS-422 is selected whenever the serial interface is used via PR# or IN# commands. If you have attempted to use this port, please notify us of the results so we can share them with other users.

* * *

PATCHING APPLESOFT

As broached in previous sections, it is possible to modify Applesoft. The procedure is very easy; however, you must insure that you don't put "bugs" in the BASIC interpreter, or the system will crash. Although this section is called PATCHING APPLESOFT, you can use the same procedure to modify INTEGER or whatever language you have installed.

As you know, the ROM in the Albert is actually RAM with a write-protect or lock on it. When modifying BASIC, you must first unlock or deprotect the RAM. The methods are as follows, and you must follow the correct order of pokes for this to be effective.

FROM BASIC TO UNLOCK:

```
POKE 49249,0
POKE 49250,0
POKE 49252,0
```

FROM BASIC TO LOCK:

```
POKE 49249,0
POKE 49251,0
POKE 49253,0
```

FROM ASSEMBLER TO UNLOCK:

```
LDA #$00
STA $C061
STA $C062
STA $C064
```

FROM ASSEMBLER TO LOCK:

```
LDA #$00
STA $C061
STA $C063
STA $C065
```

We have included a program on diskette that will do this for you. Additionally, there are various other programs included that take advantage of this system.

* * *

KEYBOARD

The Albert's keyboard is a parallel type with a serial transmitting encoder board. Each keypress is coded into ASCII by the keyboard encoder; afterwhich, the keyboard EPROM determines the actual code sent to the computer. The codes are then sent via the serial transmitter to the **AY3-1015 Universal Asyncronous Reciever Transmitter (UART)** at a rate of 30K baud. The jumper next to the UART (see parts diagram) controls the actual strobe frequency; hence, it is possible to modify other keyboards for use with the Albert. We have already done this, and have been successful; however, the procedure involved is beyond the scope of this text. It is important to note that the serial transmit rate can be adjusted from within the keyboard by turning potentiometer R-1.

The function keys are hard-wired; therefore, they cannot be user defined. However, they can be re-wired to produce other ASCII codes but we do not recommend this. Configured from the factory, function keys generate control A-F, and are "bumped" up sequentially if the [CONTROL] or [SHIFT] keys are pressed simultaneously.

Pressing **[CONTROL] [SHIFT] [generates an NMI**, and vectors to the location stored at $FFFA-FFFB. Therefore, you can modify this vector for custom applications so your programs may take advantage of this feature. This normally vectors to $3FB.

Pressing **[CONTROL] [HELP] generates a hardware reset**, and vectors to the location stored at $FFFC-FFFD. You may modify this as your application requires. This normally vectors to $FF59.

When using interrupt handling as with the NMI, make sure you have enabled interrupts prior to initializing your program, and most importantly, insure that the registers are saved upon entry and restored upon exit. Return is accomplished with a RETURN FROM INTERRUPT (RTI) instruction.

* * *

DISK DRIVES

The disk drives supplied with the Albert were either TEAC or PANASONIC HALF-HEIGHT drives. They can be purchased and/or serviced at most any computer store. You should care for them by performing a standard maintainence operation of cleaning, alignment, and speed adjustment once or twice per year.

If you experience drive "lock-up" or speed fluctuations, connect a ground wire between the drive cabinet and the Albert rear panel.

If you require additional disk drives or replacements you should consider us. **We can offer REGISTERED PURCHASERS OF ASP ultra-quiet .25 tracking SLIMLINE drives at prices BELOW mail order.** These units are far superior to both the TEAC and PANASONIC. If you would like more information, contact Amperfax.

You may also wish to take advantage of our special price on diskettes; $15.00 for a box of 10 (limited quantities). These disks bear the name of a major disk drive manufacturer on them, and are of the highest quality.

* * *

SELF-BOOTING DISKETTES

Many people have inquired as to whether or not the Albert can be refitted with the Applesoft ROM. It is possible to rebuild the Albert so that Applesoft would be in ROM; however, not only is that topic beyond the scope of this text, but you would be making a mistake. The reasoning behind this is that even if the Albert were refitted, **you would still have to boot at least 1 diskette to install DOS into the machine.** Additionally, since the ability exists to make diskettes that will **AUTOBOOT** on the Albert for far less than the modifications would cost, it just doesn't make sense. **These AUTOBOOT diskettes are fully compatible** and transportable between the Apple and the Albert. **The software package that will do this for you is S&H SOFTWARE'S TDE PACKAGE. TDE will create AUTOBOOT diskettes in less than 1 minute.** Not only is this an excellent solution to the PREBOOT problem, but **Amperfax can offer REGISTERED PURCHASERS OF ASP the TDE package for only $49.95 (normally $69.95).**

This isn't just a sales pitch, it absolutely makes good sense, especially when you consider (at last quote), the Applesoft F8 ROM alone sells for about $40, and it's also important to note here that **YOU SHOULD NEVER ATTEMPT TO PLUG ANY APPLE ROM INTO THE ALBERT AT ANY TIME. DUE TO POLARITY DIFFERENCES, EXTREME DAMAGE WILL RESULT TO YOUR COMPUTER.**

We have placed information about TDE with this package. You should really give it some thought, as it is an EXCELLENT package. If you decide you would like one, phone us directly and we will arrange it for you. This is just one special service we can offer you. Another is SAM. Read on.

* * *

SOFTWARE AUTOMATIC MOUTH

One of the most amazing features of the Albert is the ability to produce high-quality speech under program control. The program is known as SAM. Many of you who have this program already know how excellent it is, but we are also aware that many of you purchased machines without SAM. This is for you. **Amperfax can offer SAM to REGISTERED PURCHASERS OF ASP for only $59.95.** If you don't have this software, your machine is lacking a feature that is simply one of a kind.

* * *

COMPATIBILITY

As of now, there are only three known operating systems that will NOT operate on the Albert. They are:

1. APPLE COMPUTER PRODOS
2. FRANKLIN COMPUTER F-DOS
3. MICROSOFT CP/M

The following is a short list of software that is compatible.

DOS TYPE SOFTWARE

APPLE LISP
APPLE LOGO
VISICALC
APPLE PASCAL
APPLE FORTRAN
APPLEWRITER][
ARTSCI ALL APPLE][SOFTWARE
ALL SIERRA ON-LINE SOFTAWARE
BRODERBUND ARCADE MACHINE
BRODERBUND GAMES (ALL)
HAYDEN COMPILER
MICROSOFT TASC COMPILER
HIGHER TEXT
ALL S&H SOFTWARE
DON'T ASK - SAM
DESKTOP PLAN
ALL MUSE SOFTWARE
BAG OF TRICKS
ALL BEAGLE BROTHERS
ALL ADVENTURE INTERNATIONAL
ALL KOALA SOFTWARE
LISA ASSEMBLER
PENGUIN SOFTWARE (ALL PRODUCTS)
SIRIUS SOFTWARE (ALL)
DON FUDGE'S HI-RES SECRETS
EZ-DRAW

CP/M

INFOSTAR
WORDSTAR
CALCSTAR
DBASE II
SUPERSORT
ZIP
MAILMERGE
QUICK CODE
DNAMES
DGRAPH
DBPLUS
NEVADA COBOL
NEVADA FORTRAN
SUPERCALC
FRANKLIN CBASIC

AUTOCODE
DIGITAL RESEARCH ASSEMBLER
DIGITAL RESEARCH MAC
DIGITAL RESEARCH RMAC
DIGITAL RESEARCH SID
DIGITAL RESEARCH ZSID

DOS SOFTWARE IS APPLICABLE UNDER ROM 2.0 ONLY.
CP/M SOFTWARE HAS ONLY BEEN TESTED FOR PCPI STARCARD.

HARDWARE

PCPI APPLICARD - STARCARD - FRANKLIN Z80 CARD
THUNDERCLOCK
APPLE DISK INTERFACE
VIDEX VIDEOTERM
VIDEX ULTRATERM
MICROTEK 80 COLUMN CARD

INCOMPATIBILITIES

WILDCARD
APPLE ROM CARDS
CCS 7710
SNAPSHOT
APPLE GRAPHICS TABLET
MICROSOFT Z-80
ORBITAL SYSTEMS Z-80

ANY PERIPHERAL THAT USES THE DMA LINE

RETROBALL AIR HOCKEY GAME, OR ANY GAME/HARDWARE THAT DOES NOT FOLLOW APPLE STANDARD INTERFACING.

Please help us add to this list.

* * *

PROGRAMMABLE ARRAY LOGIC

The Albert incorporates six integrated circuits known as Programmable Array Logic (PAL). These PALS essentially replace the equivalent of about sixty logic chips. Each one controls key functions of the computer. These circuits are **NO LONGER MANUFACTURED.** They were specifically designed to operate on the Albert. **IF ONE OF THESE CIRCUITS FAILS, YOUR MACHINE WILL BE WORTHLESS.** We have secured a limited supply of these PALS, and they are available strictly on a first-come basis. They are sold in a kit containing all six, and CANNOT be sold separately. **The price for the kit with instructions is $66.00,** and when supplies are gone; that's it. If you assign any value to your Albert, you should have a set of these.

* * *

AFTERWORD

This concludes the Albert Support Package. We hope that you have found it to be interesting.

You may find it of interest that ASP was written on an Albert with a PCPI APPLICARD using WORDSTAR under CP/M 2.2. Total time from conception to release was eight months; therefore, you have the equivalent of this work in the package. It should help you save a tremendous amount of time doing research.

Undoubtedly, there may be areas where it appears topics were overlooked. This is either because there was no source information available, or the nature of the item(s) was standard. If additional revisions become available, you will be notified if we have your mailing information. In the mean time, keep working with your Albert. We have found it to be an excellent machine. If you write software for it, or any other computer, contact us for author information.

If this package does not contain a price list for parts and services, please notify us by mail (or in the remark area of the warranty card) and we will send it to you. At the time of writing, this list was not yet complete. It describes all replacement parts, peripherals, and repair services that are offered in conjunction with ASP.

Amperfax is growing in all directions and as each day passes, more and more items are added to our product list. If you have a specific problem requiring a hardware or software solution, you might wish to note that Amperfax specializes in Management Information Systems, High Performance Hardware and Software Systems, DBMS, and Troubleshooting at **both** the Business and Personal level.

AMPERFAX

* * *

MOTHERBOARD INTEGRATED CIRCUITS

REFER TO PEEL-OFF PARTS DIAGRAM

PART #	DESCRIPTION
Y	74LS374
A5	74LS175
A6	MC1488
A7	MC3486
A8	26LS31
A10	N9334
A11	74LS37
A15	LM1886N
A18	LM3900
Z	LM386N
C5	74LS138
C6	74LS138
C8	74LS74
C10	74LS251
C11	74LS08
C12	74LS74
C13	74LS32
C14	74LS86
C15	74LS257
C16	74LS257
C17	74LS32
C18	74LS251
C19	558CP
D1	74LS367
D2	74LS367
D3	74LS367
D4	74LS133
D11	74LS138
D12	74LS00
D13	74LS04
D14	74LS08
D15	N9334
D16	74LS139
D17	74LS00
D18	74LS04
D19	74LS02

PROGRAMMED ARRAY LOGIC

C-7	F-10
E-4	F-16
E-13	G-10

PART #	DESCRIPTION
E3	DP8304
E14	74LS174
E15	74LS174
E16	74LS74
E17	74LS09
E18	74LS74
E19	NE556
F3	74LS02
F4	74LS161
F5	74LS153
F6	74LS153
F7	74LS153
F8	74LS153
F9	74LS374
F12	74LS366
F14	74LS244
F17	74LS05
G1	74LS04
G3	74LS08
G4	74LS257
G5	74LS161
G6	74LS161
G7	74LS161
G8	74LS161
G9	74LS283
G11	74LS161
G12	74LS161
G14	74LS125
G16	74LS14

SPECIAL PURPOSE CIRCUITS

CPU	6502
ADC	ADC0809CCN
E5-E12	4164 RAM
EPROM1-BOOTROM	2732
EPROM2-I/O ROM	2732
EPROM3-CHARGEN	2732
RS232	6551
UART	AY3-1015
CLOCK	1879
E-13	74S189
E-14	74S189

74LS00

QUADRUPLE 2-INPUT POSITIVE-NAND GATES

POWER: 2mw DELAY: 9.5ms POSITIVE LOGIC: $Y=\overline{AB}$

GATE	INPUTA	INPUTB	OUTPUTY	REPLACEMENTS
1	PIN 1	PIN 2	PIN 3	FAIR 74LS00
2	4	5	6	HIT HD74LS00
3	9	10	8	MOT SN74LS00
4	12	13	11	NAT DM74LS00

Gnd = PIN 7 Vcc = PIN 14

74LS02

QUADRUPLE 2-INPUT POSITIVE NOR GATES

POWER: 2.75mw DELAY: 10ns POSITIVE LOGIC: Y = A + B

GATE	INPUTA	INPUTB	OUTPUTY	REPLACEMENTS
1	PIN 2	PIN 3	PIN 1	FAIR 74LS02
2	5	6	4	HIT HD74LS02
3	8	9	10	NAT DM74LS02
4	12	13	11	MOT SN74LS02

Gnd = PIN 7 Vcc = PIN 14

```
                  ******************
                  *     74LS04     *
                  * HEX INVERTERS *
                  ******************
```

POWER: 2mw DELAY: 9.5ns POSITIVE LOGIC: $Y = \overline{A}$

INPUT	OUTPUT	REPLACEMENTS
1A = PIN 1	1Y = PIN 2	FAIR 74LS04
2A = 3	2Y = 4	HIT HD74LS04
3A = 5	3Y = 6	NAT DM74LS04
4A = 9	4Y = 8	MOT SN74LS04
5A = 11	5Y = 10	
6A = 13	6Y = 12	
Gnd = 7	Vcc = 14	

```
          ***********************************************
          *                   74LS05                    *
          * HEX INVERTERS WITH OPEN-COLLECTOR OUTPUTS *
          ***********************************************
```

POWER: 2mw DELAY: 16ns POSITIVE LOGIC: $Y = \overline{A}$

INPUT	OUTPUT	REPLACEMENTS
1A = PIN 1	1Y = PIN 2	FAIR 74LS05
2A = 3	2Y = 4	HIT HD74LS05
3A = 5	3Y = 6	MOT SN74LS05
4A = 9	4Y = 8	NAT DM74LS05
5A = 11	5Y = 10	
6A = 13	6Y = 12	
Gnd = 7	Vcc = 14	

```
*****************************************
*                 74LS08                *
* QUADRUPLE 2-INPUT POSITIVE AND GATES *
*****************************************

POWER: 4.25mw          DELAY: 12ns          POSITIVE LOGIC: Y=AB
```

GATE	INPUTA	INPUTB	OUTPUTY	REPLACEMENTS
1	PIN 1	PIN 2	PIN 3	FAIR 74LS08
2	4	5	6	HIT HD74LS08
3	9	10	8	MOT DM74LS08
4	12	13	11	NAT SN74LS08

```
    Gnd = PIN 7        Vcc = PIN 14


*************************************************************
*                          74LS09                           *
* QUADRUPLE 2-INPUT AND GATES WITH OPEN COLLECTOR OUTPUTS *
*************************************************************

POWER: 4.25mw          DELAY: 20ns          POSITIVE LOGIC: Y=AB
```

GATE	INPUTA	INPUTB	OUTPUTY	REPLACEMENTS
1	PIN 1	PIN 2	PIN 3	FAIR 74LS09
2	4	5	6	HIT HD74LS09
3	9	10	8	MOT DM74LS09
4	12	13	11	NAT SN74LS09

```
    Gnd = PIN 7        Vcc = PIN 14
```

****
* 74LS14 *
* HEX SCHMITT-TRIGGER INVERTERS *
****

POWER: 0.8v DELAY: 15ns POSITIVE LOGIC: Y=A

INPUT	OUTPUT	REPLACEMENTS
1A = PIN 1	1Y = PIN 2	FAIR 74LS14
2A = 3	2Y = 4	HIT HD74LS14
3A = 5	3Y = 6	MOT DM74LS14
4A = 9	4Y = 8	NAT SN74LS14
5A = 11	5Y = 10	
6A = 13	6Y = 12	
Gnd = 7	Vcc = 14	

****
* 74LS32 *
* QUADRUPLE 2-INPUT POSITIVE-OR GATES *
****

POWER: 5mw DELAY: 12ns POSITIVE LOGIC: Y = A+B

GATE	INPUTA	INPUTB	OUTPUTY	REPLACEMENTS
1	PIN 1	PIN 2	PIN 3	FAIR 74LS32
2	4	5	6	HIT HD74LS32
3	9	10	8	MOT DM74LS32
4	12	13	11	NAT SN74LS32

Gnd = PIN 7 Vcc = PIN 14

```
*********************************************
*                   74LS37                  *
* QUADRUPLE 2-INPUT POSITIVE-NAND BUFFERS *
*********************************************
```

POWER: 4.3mw DELAY: 12ns POSITIVE LOGIC: $Y=\overline{AB}$

LOW LEVEL OUTPUT CURRENT: 24ma HIGH LEVEL OUTPUT CURRENT: -1.2ma

GATE	INPUTA	INPUTB	OUTPUTY	REPLACEMENTS
1	PIN 1	PIN 2	PIN 3	FAIR 74LS37
2	4	5	6	HIT HD74LS37
3	9	10	8	MOT DM74LS37
4	12	13	11	NAT SN74LS37

Gnd = PIN 7 Vcc = PIN 14

```
*****************************************************
*                      74LS74                       *
* DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS *
*              WITH PRESET AND CLEAR                *
*****************************************************
```

fmax: 33mhz PWR/F-F: 10mw SETUP:20ns HOLD: 5ns

PIN	SIGNAL	REPLACEMENTS
1	FF1 CLEAR	FAIR 74LS74
2	1 D	HIT HD74LS74
3	1 CLOCK	MOT DM74LS74
4	1 PRESET	NAT SN74LS74
5	1 Q	
6	1 QBAR	
7	Gnd	
8	FF2 QBAR	
9	2 Q	
10	2 PRESET	
11	2 CLOCK	
12	2 D	
13	2 CLEAR	
14	Vcc	

****
* 74LS86 *
* QUAD 2-INPUT EOR GATES *
****

POWER: 30mw DELAY: 10ns POSITIVE LOGIC $Y=A(+)B=A\overline{B}+\overline{A}B$

GATE	INPUTA	INPUTB	OUTPUTY	REPLACEMENTS
1	PIN 1	PIN 2	PIN 3	FAIR 74LS86
2	4	5	6	HIT HD74LS86
3	9	10	8	MOT DM74LS86
4	12	13	11	NAT SN74LS86

Gnd = PIN 7 Vcc = PIN 14

****
* 74LS125 *
* QUAD BUS BUFFER GATES WITH THREE-STATE OUTPUTS *
****

DELAY: 8ns MAX SOURCE CURRENT: -1ma MAX SINK CURRENT: 12ma
POSITIVE LOGIC Y = A

GATE	GBAR	INPUTA	OUTPUTY	REPLACEMENTS
1	PIN 1	PIN 2	PIN 3	FAIR 74LS125
2	4	5	6	HIT HD74LS125
3	10	9	8	MOT DM74LS125
4	13	12	11	NAT SN74LS125

Gnd = PIN 7 Vcc = PIN 14

****
* 74LS133 *
* 13-INPUT POSITIVE-NAND GATES *
****

POWER: 0.15mw DELAY: 8ns POSITIVE LOGIC: $Y = \overline{ABCDEFGHIJKLM}$

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	A	9	Y	FAIR 74LS133
2	B	10	H	HIT HD74LS133
3	C	11	I	MOT DM74LS133
4	D	12	J	NAT SN74LS133
5	E	13	K	
6	F	14	L	
7	G	15	M	
8	Gnd	16	Vcc	

****
* 74LS138 *
* 3 TO 8 LINE DECODER OR DEMULTIPLEXER *
****

POWER: 31mw SELECT TIME: 22ns ENABLE TIME: 21ns

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	A	9	Y6	FAIR 74LS138
2	B	10	Y5	HIT HD74LS138
3	C	11	Y4	MOT DM74LS138
4	GBAR2A	12	Y3	NAT SN74LS138
5	GBAR2B	13	Y2	
6	G1	14	Y1	
7	Y7	15	Y0	
8	Gnd	16	Vcc	

```
*********************************************
*                   74LS139                 *
* DUAL 2 TO 4 LINE DECODERS OR MULTIPLEXERS *
*********************************************
```

POWER: 34mw SELECT TIME: 22ns ENABLE TIME: 19ns

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	1GBAR	9	2Y3	FAIR 74LS139
2	1A	10	2Y2	HIT HD74LS139
3	1B	11	2Y1	MOT DM74LS139
4	1Y0	12	2Y0	NAT SN74LS139
5	1Y1	13	2B	
6	1Y2	14	2A	
7	1Y3	15	2GBAR	
8	Gnd	16	Vcc	

```
*************************************************************
*                          74LS153                          *
* DUAL 4 LINE TO 1 LINE DATA SELECTORS OR MULTIPLEXERS *
*************************************************************
```

DELAY DATA TO NON-INV OUTPUT: 14ns DELAY FROM ENABLE: 17ns
POWER: 31mw

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	1GBAR	9	2Y	FAIR 74LS153
2	B	10	2C0	HIT HD74LS153
3	1C3	11	2C1	MOT DM74LS153
4	1C2	12	2C2	NAT SN74LS153
5	1C1	13	2C3	
6	1C0	14	A	
7	1Y	15	2GBAR	
8	Gnd	16	Vcc	

****
* 74LS161 *
* SYNCRONOUS 4-BIT COUNTERS *
****

POWER: 93mw COUNTER FREQUENCY: 25mhz

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	CLEAR	9	LOAD	FAIR 74LS161
2	CLOCK	10	ENT	HIT HD74LS161
3	A	11	QD	MOT DM74LS161
4	B	12	QC	NAT SN74LS161
5	C	13	QB	
6	D	14	QA	
7	ENP	15	RCO	
8	Gnd	16	Vcc	

****
* 74LS165 *
* 8-BIT SHIFT REGISTERS *
****

POWER: 105mw SHIFT FREQUENCY: 35mhz

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	SHIFT/LOAD	9	QH	FAIR 74LS165
2	CLOCK	10	SERIAL OUT	HIT HD74LS165
3	E	11	A	MOT DM74LS165
4	F	12	B	NAT SN74LS165
5	G	13	C	
6	H	14	D	
7	QBAR H	15	CLOCK INH	
8	Gnd	16	Vcc	

****
* 74LS174 *
* HEX D-TYPE FLIP FLOP *
****

FREQUENCY: 40mhz POWER PER F-F: 10.6mw DELAY SETUP: 20ns
HOLD: 5ns

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	CLEAR	9	CLOCK	FAIR 74LS174
2	1Q	10	4Q	HIT HD74LS174
3	1D	11	4D	MOT DM74LS174
4	2D	12	5Q	NAT SN74LS174
5	2Q	13	5D	
6	3D	14	6D	
7	3Q	15	6Q	
8	Gnd	16	Vcc	

****
* 74LS175 *
* QUAD D-TYPE FLIP FLOPS *
****

FREQUENCY: 40mhz POWER PER F-F: 10.6mw DELAY SETUP: 20ns
HOLD: 5ns

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	CLEAR	9	CLOCK	FAIR 74LS175
2	1Q	10	3Q	HIT HD74LS175
3	1Q BAR	11	3Q BAR	MOT DM74LS175
4	1D	12	3D	NAT SN74LS175
5	2D	13	4D	
6	2Q BAR	14	4Q BAR	
7	2Q	15	4Q	
8	Gnd	16	Vcc	

****
* 74LS244 *
* OCTAL BUFFERS / LINE DRIVERS / LINE RECEIVERS *
****

DELAY: 10ns MAX SOURCE CURRENT: -15ma MAX SINK CURRENT: 24ma

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	1G BAR	11	2A1	FAIR 74LS244
2	1A1	12	1Y4	HIT HD74LS244
3	2Y4	13	2A2	MOT DM74LS244
4	1A2	14	1Y3	NAT SN74LS244
5	2Y3	15	2A3	
6	1A3	16	1Y2	
7	2Y2	17	2A4	
8	1A4	18	1Y1	
9	2Y1	19	2G BAR	
10	Gnd	20	Vcc	

****
* 74LS251 *
* DATA SELECTORS/MULTIPLEXERS *
****

DELAY DATA TO INV OUTPUT: 17ns DELAY FROM ENABLE: 21ns
DELAY DATA TO NON-INV OUTPUT: 21ns POWER: 35mw

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	D3	9	C	FAIR 74LS251
2	D2	10	B	HIT HD74LS251
3	D1	11	A	MOT DM74LS251
4	D0	12	D7	NAT SN74LS251
5	Y	13	D6	
6	W	14	D5	
7	G BAR	15	D4	
8	Gnd	16	Vcc	

```
**************************************
*              74LS257               *
* QUAD DATA SELECTORS/MULTIPLEXERS   *
**************************************
```

DELAY TO NON-INV OUTPUT: 11ns DELAY FROM ENABLE: 19ns
POWER: 60mw

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	A/B BAR	9	3Y	FAIR 74LS257
2	1A	10	3B	HIT HD74LS257
3	1B	11	3A	MOT DM74LS257
4	1Y	12	4Y	NAT SN74LS257
5	2A	13	4B	
6	2B	14	4A	
7	2Y	15	G BAR	
8	Gnd	16	Vcc	

```
*****************************
*          74LS283          *
* 4-BIT BINARY FULL-ADDERS  *
*****************************
```

POWER PER BIT: 24mw CARRY TIME: 10ns ADD TIME: 15ns

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	SUM2	9	CARRY4	FAIR 74LS283
2	BIT2	10	SUM4	HIT HD74LS283
3	A2	11	BIT4	MOT DM74LS283
4	SUM1	12	A4	NAT SN74LS283
5	A1	13	SUM3	
6	BIT1	14	A3	
7	CARRY0	15	BIT3	
8	Gnd	16	Vcc	

****
* 74LS366 *
* HEX BUS DRIVERS *
****

DELAY: 9.5ns MAX SOURCE CURRENT: -2.6ma MAX SINK CURRENT: 24ma

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	G BAR1	9	Y4	FAIR 74LS366
2	A1	10	A4	HIT HD74LS366
3	Y1	11	Y5	MOT DM74LS366
4	A2	12	A5	NAT SN74LS366
5	Y2	13	Y6	
6	A3	14	A6	
7	Y3	15	G BAR2	
8	Gnd	16	Vcc	

****
* 74LS367 *
* HEX BUS DRIVERS *
****

DELAY: 9.5ns MAX SOURCE CURRENT: -2.6ma MAX SINK CURRENT: 24ma

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	1G BAR	9	1Y4	FAIR 74LS367
2	1A1	10	1A4	HIT HD74LS367
3	1Y1	11	2Y1	MOT DM74LS367
4	1A2	12	2A1	NAT SN74LS367
5	1Y2	13	2Y2	
6	1A3	14	2A2	
7	1Y3	15	2G BAR	
8	Gnd	16	Vcc	

****
* 74LS374 *
* OCTAL D-TYPE FLIP-FLOPS *
****

FREQUENCY: 50mhz POWER PER F-F: 17mw DATA SETUP: 20ns
HOLD: 0ns

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	OC BAR	11	CLOCK	FAIR 74LS374
2	1Q	12	5Q	HIT HD74LS374
3	1D	13	5D	MOT DM74LS374
4	2D	14	6D	NAT SN74LS374
5	2Q	15	6Q	
6	3Q	16	7Q	
7	3D	17	7D	
8	4D	18	8D	
9	4Q	19	SQ	
10	Gnd	20	Vcc	

****
* LM386 *
* LOW POWER AUDIO AMPLIFIER *
****

PIN	SIGNAL	REPLACEMENTS
1	GAIN	NAT LM386
2	INPUT-	
3	INPUT+	
4	Gnd	
5	Vout	
6	Vs	
7	BYPASS	
8	GAIN	

****
* NE555 *
* TIMER *
****

PIN	SIGNAL	REPLACEMENTS
1	Gnd	RCA SK3564
2	TRIGGER	FAIR UA555C
3	OUTPUT	NAT LM555C
4	RESET	SIG SE555C
5	Vref	
6	THRESHOLD	
7	DISCHARGE	
8	Vcc	

****
* 556 *
* DUAL TIMER *
****

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	DISCHARGE	8	TRIGGER	NAT LM556
2	THRESHOLD	9	OUTPUT	FAIR UA556C
3	CONTROL VOLT	10	RESET	SIG SE556
4	RESET	11	CONTROL VOLT	INT NE556
5	OUTPUT	12	THRESHOLD	
6	TRIGGER	13	DISCHARGE	
7	Gnd	14	Vcc	

****
* 558 *
* QUAD TIMER *
****

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	OUTPUTA	9	OUTPUTC	SIG NE558
2	TIMINGA	10	TIMINGC	SA558
3	TRIGGERA	11	TRIGGERC	SE558
4	CONTROL VOLT	12	Gnd	
5	Vcc	13	RESET	
6	TRIGGERB	14	TRIGGERD	
7	TIMINGB	15	TIMINGD	
8	OUTPUTB	16	OUTPUTD	

****
* LM1886 *
* VIDEO PROCESSOR *
****

PINOUT UNAVAILABLE AT THIS TIME

REPLACEMENTS

NAT LM1886N

****
* LM3900 *
* OPERATIONAL AMPLIFIER *
****

PIN	SIGNAL	PIN	SIGNAL
1	INPUT1+	8	Vcc
2	INPUT2+	9	INPUT3+
3	INPUT2-	10	INPUT4+
4	OUTPUT2	11	INPUT4-
5	OUTPUT1	12	OUTPUT4
6	INPUT1-	13	OUTPUT3
7	Gnd	14	INPUT3-

REPLACEMENTS

RCA SK3688
NAT LM3900N

****
* DP8304 *
* DUAL FULL ADDER *
****

PINOUT UNAVAILABLE AT THIS TIME

REPLACEMENTS

AMD DP8304B
MOT MC8304
NAT DP8304

****
* 9334 *
* 8-BIT ADDRESSABLE LATCH *
****

PINOUT UNAVAILABLE AT THIS TIME

REPLACEMENTS

FAIR 9334C
NAT DM9334
SIG N9334

SPECIALIZED INTEGRATED CIRCUITS

****
* MC1488 *
* RS-232 TRANSMITTER *
****

OUTPUT AVG: +/- 7.5v

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	Vee	8	OUTPUTC	AMD MC1488
2	INPUTA	9	INPUTC2	EXAR XR1488
3	OUTPUTA	10	INPUTC1	FAIR UA1488
4	INPUTB1	11	OUTPUTD	NATL DS1488
5	INPUTB2	12	INPUTD2	RAYT RM1488
6	OUTPUTB	13	INPUTD1	TI MC1488
7	Gnd	14	Vcc	SILG SG1488

****
* 26LS31 *
* RS-422 TRANSMITTER *
****

OUTPUT: 20ma. COOMMON MODE: +/- 7v

PINOUT UNAVAILABLE AT THIS TIME	REPLACEMENTS
	AMD AM26LS31C
	MOT AM26LS31
	AM26LS31C
	NAT DS26LS31C
	SIG AM26LS31
	TI AM26LS31

****
* MC3486 *
* RS-422 RECEIVER *
****

INPUT THRESHOLD: +/- .02v COMMON MODE: +/- 3v

PINOUT UNAVAILABLE AT THIS TIME

REPLACEMENTS

NAT DS3486
TI MC3486
MOT MC3486

****
* SY6551 *
* ASYNCRONOOUS COMMUNICATION INTERFACE ADAPTER *
****

PIN	SIGNAL	PIN	SIGNAL
1	Gnd	15	Vcc
2	CS0	16	DCD BAR
3	CS1 BAR	17	DSR BAR
4	RES BAR	18	DATA0
5	RxC	19	DATA1
6	XTAL 1	20	DATA2
7	XTAL 2	21	DATA3
8	RTS BAR	22	DATA4
9	CTS BAR	23	DATA5
10	TxD	24	DATA6
11	DTR BAR	25	DATA7
12	RxD	26	IRQ BAR
13	RS0	27	CLOCK2
14	RS1	28	R/W

REPLACEMENTS

AMI S6551
ROC R6551
SYN SY6551

****
* 6502 MICROPROCESSOR *
****

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	Gnd	21	Gnd	ROC R6502
2	RDY	22	A12	SYN SY6502
3	PHASE 1	23	A13	
4	IRQ	24	A14	
5	-	25	A15	
6	NMI	26	D7	
7	SYNC	27	D6	
8	Vcc	28	D5	
9	A0	29	D4	
10	A1	30	D3	
11	A2	31	D2	
12	A3	32	D1	
13	A4	33	D0	
14	A5	34	R/W	
15	A6	35	-	
16	A7	36	-	
17	A8	37	PHASE 0	
18	A9	38	Gnd	
19	A10	39	PHASE 2	
20	A11	40	RES	

****
* ADC0809CCN ADC *
****

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	CS BAR	11	DB7/MSB	MOS MK50809
2	RD BAR	12	DB6	TI ADC0809
3	WR BAR	13	DB5	NAT ADC0809CCN
4	CLK/IN	14	DB4	
5	INT	15	DB3	
6	Vin+	16	DB2	
7	Vin-	17	DB1	
8	Gnd	18	DB0	
9	Vref/2	19	CLK/R	
10	Gnd	20	Vcc/Vref	

1879 CLOCK

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	INT	13	DB0	RCA SK1879
2	RES	14	DB1	RCA CDP1879
3	PDOWN	15	DB2	NAT LM1879
4	RD	16	DB3	
5	MEM	17	DB4	
6	TPB	18	DB5	
7	TPA	19	DB6	
8	CS	20	DB7	
9	A2	21	CLOCK OUT	
10	A1	22	XTAL BAR	
11	A0	23	XTAL	
12	Vss	24	Vdd	

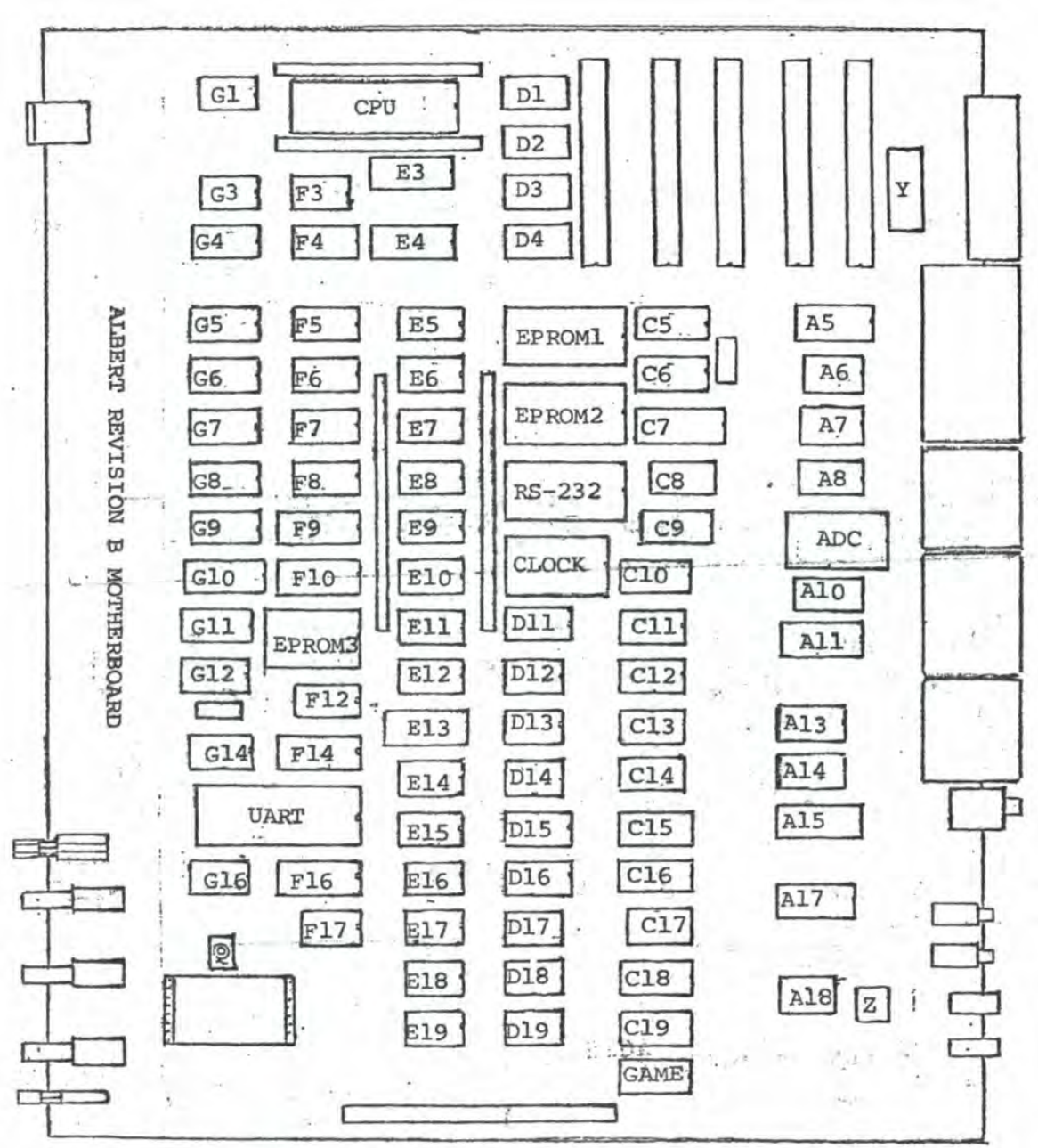
74LS189
64 BIT HIGH-PERFORMANCE RAM

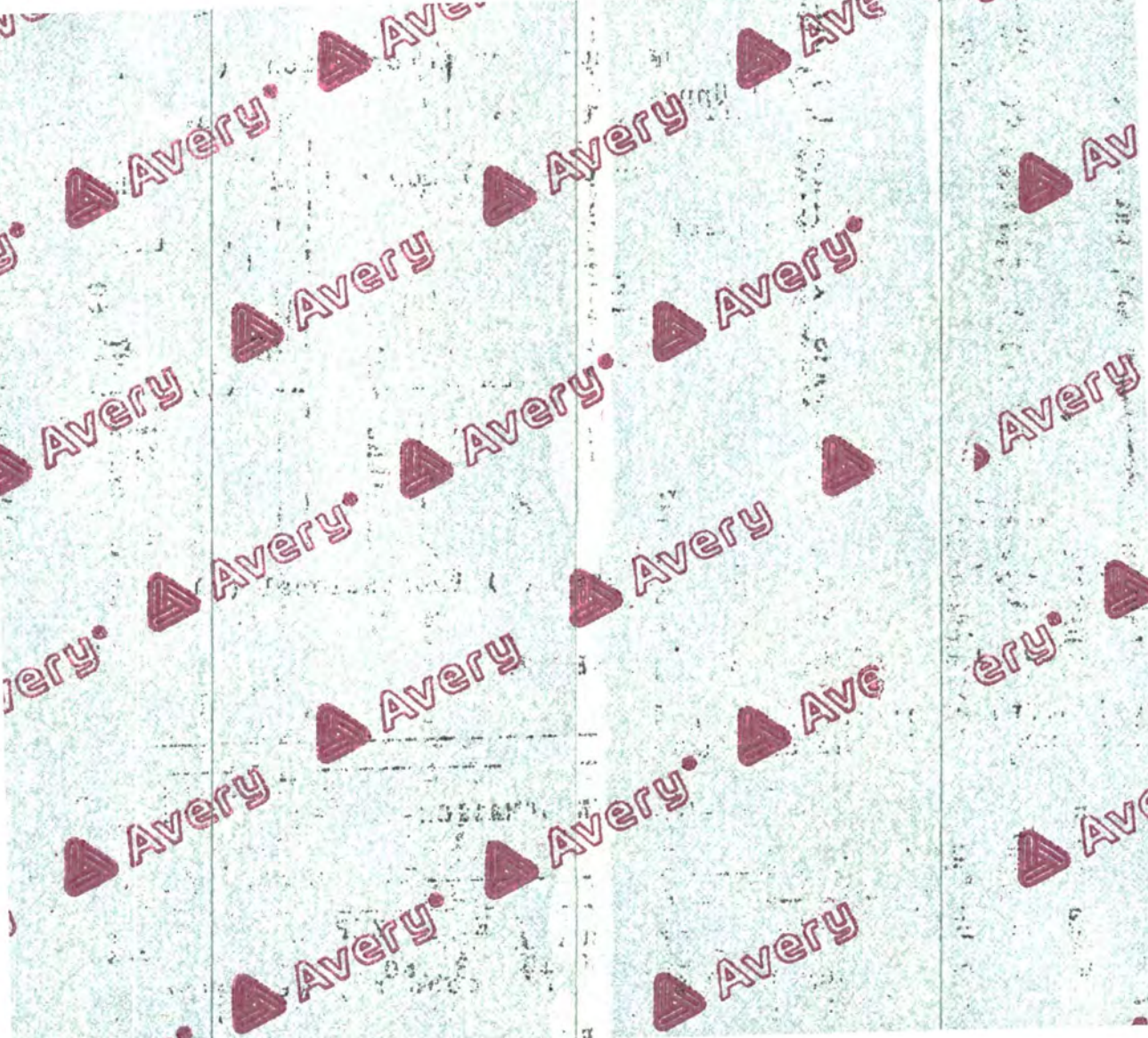
ADDRESS TIME: 50ns ENABLE TIME: 35ns POWER PER BIT: 2.7mw

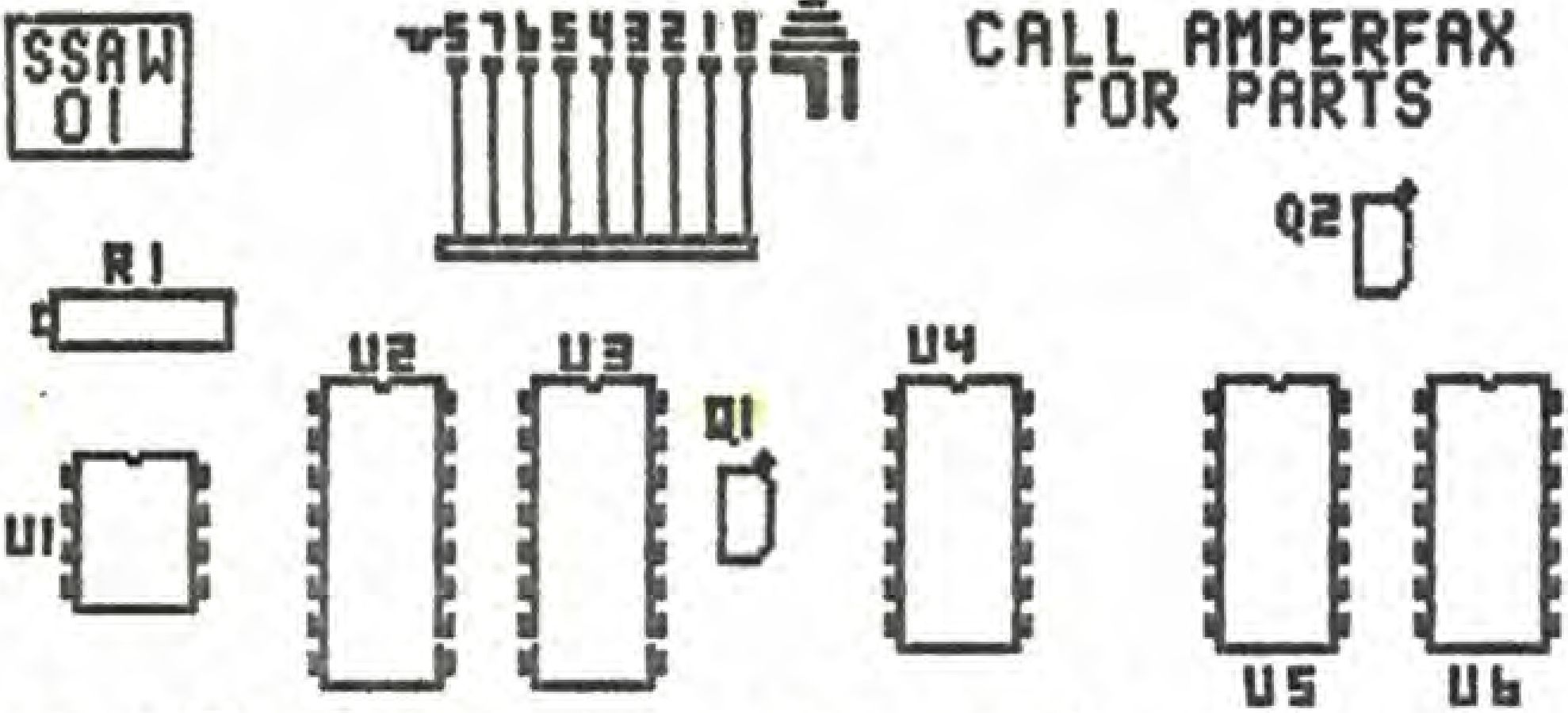
PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS
1	A0	9	Q BAR3	FAIR 74LS189
2	S BAR	10	D3	TI SN74LS189A
3	R/W	11	Q BAR4	
4	D1	12	D4	
5	Q BAR1	13	A3	
6	D2	14	A2	
7	Q BAR2	15	A1	
8	Gnd	16	Vcc	

****
* AY3-1015 UART *
****

PIN	SIGNAL	PIN	SIGNAL	REPLACEMENTS	
1	Vcc	21	XR	FAIR	F6850
2	-	22	TBMT	AMI	S1602
3	Gnd	23	DS	MOT	MC6850
4	RDE	24	EOC	RCA	CDP6402
5	RD8	25	SO	HIT	HD6850
6	RD7	26	DB1	FUJ	MB8868A
7	RD6	27	DB2		
8	RD5	28	DB3		
9	RD4	29	DB4		
10	RD3	30	DB5		
11	RD2	31	DB6		
12	RD1	32	DB7		
13	PE	33	DB8		
14	FE	34	CS		
15	OR	35	NP		
16	SWE	36	TSB		
17	RCP	37	NB2		
18	RDAV	38	NB1		
19	DAV	39	EPS		
20	SI	40	TCP		


ALBERT REVISION B MOTHERBOARD
G1
CPU
D1
D2
E3
G3
F3
D3
Y
G4
F4
E4
D4
G5
F5
E5
EPROM1
C5
A5
G6
F6
E6
C6
A6
G7
F7
E7
EPROM2
C7
A7
G8
F8
E8
RS-232
C8
A8
G9
F9
E9
C9
ADC
G10
F10
E10
CLOCK
C10
A10
G11
EPROM3
E11
D11
C11
A11
G12
F12
E12
D12
C12
E13
D13
C13
A13
G14
F14
E14
D14
C14
A14
UART
E15
D15
C15
A15
G16
F16
E16
D16
C16
F17
E17
D17
C17
A17
E18
D18
C18
A18
Z
E19
D19
C19
GAME


Avery



CALL [305] 667-5555 FOR SERVICE

U1 = 555
U2 = 74LS165
U3 = 74LS165
U4 = NE556
U5 = 7400N
U6 = 74LS074

MAJOR ENCODER PARTS

Q1 = 2N3906
Q2 = 2N3906
R1 = BAUD ADJUST
ALL CAPS = .1 UF

ASP

READING REFERENCES

The following is a source list of material related to the Albert.

APPLESOFT][BASIC PROGRAMMING REFERENCE MANUAL. APPLE COMPUTER INC., CUPERTINO CA., 1978.

THE DOS MANUAL. APPLE COMPUTER INC., CUPERTINO CA., 1980.

APPLE][REFERENCE MANUAL. APPLE COMPUTER INC., CUPERTINO CA., 1979.

MCS6500 PROGRAMMING MANUAL. MOS TECHNOLOGY INC., NORRISTOWN PA., JANUARY 1976.

LUEBBERT WF: WHAT'S WHERE IN THE APPLE?: AN ATLAS TO THE APPLE COMPUTER. MICRO INK, INC., CHELMSFORD, MA. 1981.

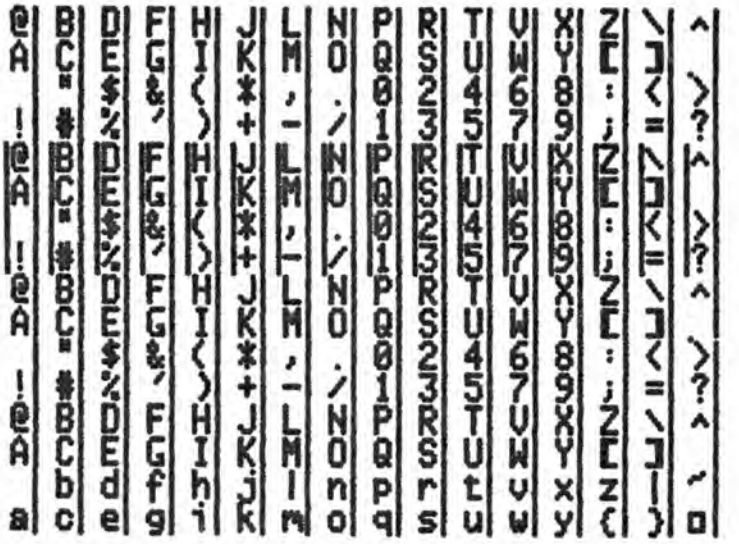
WAGNER R: ASSEMBLY LINES: THE BOOK. SOFTALK PUBLISHING, INC., NORTH HOLLYWOOD, CA. 1982.

PARKER AJ, STEWART JF: APPLE BASIC FOR BUSINESS FOR THE APPLE][. RESTON PUBLISHING COMPANY, INC., RESTON, VA. 1981

LEWIS TG: PASCAL PROGRAMMING FOR THE APPLE. RESTON PUBLISHING COMPANY, INC., RESTON, VA. 1981

WILLIAMS K, KERNAGHAN B, KERNAGHAN L: APPLE][COMPUTER GRAPHICS. ROBERT J. BRADY COMPANY, BOWIE, MD. 1983

CHARACTER SET COMPARISON



ORIGINAL

AMPERFAX