XLR8er™

OPERATOR MANUAL

Revision 2.0 04/30/88

Copyright © 1988 MISOSYS, Inc., All rights reserved

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of MISOSYS, Inc.

MISOSYS, Inc
PO Box 239
Sterling, VA 22170-0239
703-450-4181

SOFTWARE LICENSE AGREEMENT

MISOSYS, Inc., authorizes you to use this software on only one computer. You are authorized to make archived copies of the software for the sole purpose of backing up your software.

MISOSYS, Inc., warrants the physical diskette and physical documentation to be free of defects in materials and workmanship for a period of 30 days from the date of purchase. Upon notification of defects in material or workmanship within the warranty period, MISOSYS, Inc., will replace the defective documentation or diskette.

MISOSYS, Inc disclaims all other warranties, expressed or implied, including but not limited to any implied warranty of merchantability and/or fitness for particular purpose. Under no circumstances shall MISOSYS, Inc be liable for any loss of profit or any other damage, including but not limited to special, incidental, exemplary, consequential or other damages.

CP/M is a trademark of Digital Research, Inc.
LDOS is a trademark of MISOSYS, Inc.
TRSDOS is a trademark of the Tandy Corporation.

Table of Contents

Chapter 1 - General Description	1
Chapter 2 - Installation	3
Chapter 3 - Software	10
Chapter 4 - I/O	22
Chapter 5 - Programmers Reference	24
Appendix - Application Notes	35
Appendix - Use with Graphics Boards	37
Appendix - Figures	39
Warranty	46

Chapter 1 General Description

Introduction

This chapter provides an overview and functional description of the XLR8er board for the Model 4 series of computers. The purpose of this chapter is to provide a basic understanding of the XLR8er board and how it greatly enhances the performance of your Model 4.

Features

The XLR8er circuit board is designed around Hitachi's new state- of-the-art high integration microprocessor, the HD64180. The HD64180 provides the benefits of high performance, reduced system costs and low power operation while maintaining compatibility with the existing Z80 instruction set. Performance is improved by virtue of high operating frequency, pipelining of instructions, an enhanced instruction set and an integrated Memory Management Unit with 512K bytes of physical memory address space. Combined with the additional hardware capabilities of the HD64180 and the circuitry on the XLR8er board, an exceptional level of performance improvement can be realized as compared to the standard Model 4 using a 4MHZ Z80. The main features of the XLR8er board are:

* HD64180 microprocessor @ 6.144 MHz (equivalent to Z80 @ 8 MHz)

* 256K of additional high speed RAM

* On chip memory management unit

* Two channel DMA controller (transfers of 1 MB/sec)

* Internal Wait State Generator

* Two channel 16-bit programmable reload timer

* Programmable Dynamic RAM refresh addressing and timing

* Interrupt controller with 12 levels of control

* Twelve new instructions including multiply

* OPTIONAL: Two channels of RS232 communications

❋ OPTIONAL: Expansion port (compatible with Ciarcia's SB180 as featured in Byte magazine)

System Overview

The XLR8er board installs in the Model 4 by removing the existing Z80 from its socket and plugging the XLR8er connector and board into the empty socket. No soldering or trace cuts are involved in installing the board. Once the board is in place, the existing memory and I/O's that were on your Model 4 motherboard still operates as it always has. You still have access to the 128K of RAM on the motherboard as well as all the I/O ports.

Since the HD64180 is capable of directly addressing 512K, the additional 256K of RAM on the XLR8er board starts at the physical address of 262,144 (40000 hex). The information contained in chapter five explains in greater detail how to access the additional RAM. Any physical address below 262,144 (40000 hex) accesses any RAM that is present on the motherboard.

The additional I/O ports provided by the XLR8er board are internal to the HD64180. The port addresses start at 00 hex and go to 3F hex. These port addresses may overlap port addresses of any additional devices that you may have installed in your computer and will operate in conjunction with them. (this technique is explained in chapter five).

The additional RS232 ports are brought out on a 20 pin ribbon cable header on the XLR8er board and the expansion bus is brought out on a 40 pin ribbon cable header. These ports are mapped into the Model 4's I/O map in such a manner as they do not conflict with any of the I/O ports on a standard Model 4. As stated above, if port address conflicts do occur due to additional hardware you may have installed on your Model 4, the HD64180 instruction set will allow you to access both I/O ports due to the fact that one set is internal to the chip.

Model differences

There are three types of Model 4 motherboards that the XLR8er Board may be installed on with the types listed as below.

> Type 1 - The original motherboard with the Z80 socket in the lower left of the circuit board, and is considered NON-GATE ARRAY. The model number 26-1069 is located on the bottom of the machine

> Type 2 - The new GATE ARRAY motherboard with the Z80 socket in the upper left of the circuit board. (This includes some Model 4s and all Model 4Ds) Model numbers 26-1069A and 26-1070.

> Type 3 - The motherboard in the Model 4P. (Gate array and non-gate array)

Special Instructions for installing with Graphics Boards are located in the APPENDIX.

Determine your type of motherboard and then proceed to that section of this chapter to get the installation procedure.

Before installing the XLR8er board make backup copies of the distribution software and store the original(s) in a safe place. NEVER use the original distribution disks as working disks in your system.

> ***WARNING: Before attempting to install the XLR8er board make sure your computer is unplugged from the wall socket to avoid electrical shock. Also, during the installation process, avoid any contact with the high voltage portion of your CRT as this section could still produce a potential shock. If you feel unsure about being able to install the XLR8er ask a technically oriented friend to help you out.***

In all of the installation instruction sections you are instructed to examine the speed of the RAMS that are currently in your system. The dynamic RAM chips in your Model 4 must have an access time of at least 150ns or

you cannot run the XLR8er board at full speed. If you need some 150ns RAM, MISOSYS can provide them at a reasonable cost.

It may also be necessary to replace one other chip associated with keyboard functions, labeled 74LS245P. Its location is given in the installation instruction. The replacement chip is 74HCT245 and is a higher speed chip than the 74LS245. This 74HCT245 chip is provided with the XLR8er kit.

On both the Model 4 and the Model 4D systems, it is necessary to install a new RF shield in order to complete the installation of the XLR8er board, unless your installation includes a graphics board.

Type 1 Motherboards (26-1069)

Note: All instructions are given as viewed from the rear of the Model 4. Refer to Figure 1.1 for this installation.

1. Remove the screws on the underneath side of your computer holding the top molding on the base.

2. Make sure the doors to your floppy disk drives are in position to clear the top cover when it is removed. (Some types are best left open, others left closed.)

3. Carefully lift up the top portion of the housing (***Be careful as the CRT is contained in this part of the enclosure. Be especially careful to not let the yoke of the CRT bump against anything***). Once it has cleared the top of the disk drives, lay it on it's side being careful not to stretch the wiring harness going to the CRT.

4. Remove the screws holding the RF shield in place, and then remove the RF shield.

5. Carefully pry the Z80 from its socket and save it in case you need to use it again.

6. Check the dynamic ram chips and assure that they are at least 150ns parts. They will have a "-150" or "-15" on the chip after the RAM part number. If they are 200ns RAMs, they will probably have a "-2" number suffix.

Check the 74LS245P chip and replace with 74HCT245P if needed. Note that the suffix letter may be different. It will be located to the left of the PAL chip and if two are there, it will be the furthermost one left of the PAL chip. (The Pal chip is in socket U-72)

7. Now check the revision number of your motherboard. If it a revision "C" or "D" the following step is required. find the jumper wire on the motherboard that is labeled as "W3" on each end. (This wire runs from U31 on the left portion of the motherboard to U25 on the right portion). This wire needs to be grounded for proper operation of the XLR8er. The easiest way to ground W3 is to remove a small portion of the insulation and attach a wire to insert under one of the mounting screws holding the mother board in place. If "W3" does not run from U25 to U31, then do not ground it!

8. Plug the 40 pin ribbon cable connector into the Z80 socket being careful not to bend under any pins. Be sure that the ribbon is **bent per sketch (see figure 1.1).**

9. Install the six #4 screws provided, in the holes surrounding the opening in the back of the RF shield. Secure these screws with #4 nuts provided. The screws should be inserted from the inside of the RF shield, facing outward toward the back of the unit. The #4 nuts will act as stand-off spacers for the cover plate which will mount over the screws after the RF shield is reinstalled on the system.

10. Install the new **RF shield** in your system, allowing the 40 pin cable to exit through the large port in the new RF shield Check to see that the motherboard housing is as far forward as possible; the mounting brackets have some adjusment in them. Ensure that they are pushed as far forward as possible.

11. Mount the XLR8er board onto the RF shield cover plate using four #4 screws provided; insert the four nylon spacers to allow for separation between the XLR8er board and the cover plate. ***The XLR8er board must be mounted on the fiberglass side (not the copper side) of the cover plate.*** It should be installed using the 4 holes marked for the Model 4. The connectors (headers) on the XLR8er should be aligned so that they are facing toward the right side of the Model 4 ***as viewed from the back of the board.*** The diagonals are marked for the different models.

12. Once the XLR8er board is in position, plug the free end of the forty pin ribbon cable connector into the empty Z80 socket on the XLR8er board, being careful not to bend under or break any pins on the connector.

13. Install the RF shield cover plate with the XLR8er board facing the inside of shield, and the connectors on the XLR8er facing towards the right as you view the system from the back. Attach the cover plate to the shield by aligning the six holes over the six protruding screws; tighten with six additional #4 nuts.

14. Check all connections and be sure to plug in the power supply plug on top of the mother board. Tighten all screws and nuts that might have been loosened during disassembly.

15. CAREFULLY set the top back on the computer (again be careful of the CRT yoke).

16. Put the screws back in the base of the computer and you are through.

17. Refer to the appropriate software section now to boot up using your new XLR8er board.

Type 2 Motherboards

Note: All instructions are given as viewed from the rear of the Model 4 Gate Array and Model 4D (26-1069A and 26-1070). Refer to Figure 1.2 for this installation.

1. Remove the screws on the underneath side of your computer holding the top molding on the base.

2. Make sure the doors to your floppy disk drives are in position to clear the top when removed. (Some are best left open, some left closed.)

3. Carefully lift up the top portion of the housing (***Be careful as the CRT is contained in this part of the enclosure. Be especially careful to not let the yoke of the CRT bump against anything***). Once it has cleared the top of the disk drives lay it over to the side being careful not to stretch the wiring harness going to the CRT.

4. Remove the screws holding the RF shield in place, and then remove the RF shield. Unplug the disk drive power supply plug on the top of the motherboard.

5. Carefully pry the Z80 from its socket and save it in case you need to use it again; it is located in the upper left corner of the motherboard.

6. Check the dynamic RAM chips and ensure that they are at least 150ns parts. They will have a "-150" or "-15" after the RAM part number if they are. If they are 200ns RAMs, they will probably have a "-2" number suffix.

   Check the 74LS245P chip (socket U-77) and replace with a 74HCT245P if necessary. Note that the suffix letter may be different.

7. Plug the 40 pin ribbon cable connector into the Z80 socket being careful not to bend under any pins. Be sure that the ribbon is **bent per sketch (see figure 1.2).**

8. Install the six #4 screws provided, in the holes surrounding the opening in the back of the RF shield. Secure these screws with #4 nuts provided. The screws should be inserted from the inside of the RF shield, facing outward toward the back of the unit. The #4 nuts will act as stand-off spacers for the cover plate which will mount over the screws after the RF shield is reinstalled on the system.

9. Install the new **RF shield** in your system, allowing the 40 pin cable to exit through the large port in the back of the shield. Check to see that the motherboard housing is as far forward as possible; the mounting brackets have some adjustment in them. Ensure that they are pushed as far forward as possible.

10. Install the XLR8er board on the RF shield cover plate using the four nylon spacers to allow for separation. ***The XLR8er board must be mounted on the fiberglass side (not the copper side) of the cover plate.*** It should be installed using the four holes marked for the Model 4D. The connectors (headers) on the XLR8er should be aligned so that they are facing toward the right side of the Model 4D ***as viewed from the back of the board.*** The diagonals are marked for the different models.

11. Once the XLR8er board is in position, plug the free end of the forty pin ribbon cable connector into the empty Z80 socket, being careful not to bend under any pins on the connector.

12. Install the RF shield cover plate with the XLR8er board facing the inside of shield, and the connectors on the XLR8er facing towards the right as you view the system from the back. Attach the cover plate to the shield by aligning the six holes over the six protruding screws; tighten with six additional #4 nuts. Reconnect the disk drive power supply plug on top of the RF shield.

13. CAREFULLY set the top back on the computer (again be careful of the CRT yoke).

14. Put the screws back in the base of the computer and you are through.

15. Refer to the appropriate software section now to boot up using your new XLR8er board.

Type 3 Motherboards

Note: The XLR8er board fits in to the modem slot on your Model 4P computer. Refer to Figure 1.3 for this installation.

1. Remove the case on the 4P by removing the six (6) Phillips head screws holding it on the chassis. The first four screws are the beige colored screws around the front bezel of the computer. The last two screws are under the carrying handle. Additional screws may be located near the I/O ports. Set the screws aside in a safe place for now.

2. Turn the 4P up on its screen now and pull straight up on the case. Be careful of the keyboard as you pull off the case as it might fall as the case is being removed.

3. Once the case has been removed, the pan assembly in which the main PC board is encased must be removed. There are four (4) screws on each side of the pan assembly. Remove these eight screws and set them aside for now.

4. Gently pry the two halves of the pan assembly apart. The disk drive ribbon cable which is connected to the computer board will have to be removed before you can fully open up the pan assembly to gain access to the circuit board.

5. With the pan assembly opened up far enough to gain access to the chips on the circuit board, gently pry the Z80 chip from its socket (U-47) and set it aside for safe keeping.

6. Plug one end of the 40 pin ribbon cable supplied with the XLR8er board into the socket you just removed the Z80 from being careful to note the position of pin 1 on the socket and on the ribbon cable connector. Also be careful not to bend under any pins on the connector as you plug it in.

7. Now slide the XLR8er board into the internal modem slot with the I/O connectors towards the inside of the slot (near the internal modem plug; refer to Figure 1.3).

8. While you have access to the motherboard, check the RAM installed in your 4P and make sure it is 150ns RAM (it will be located in sockets U-133 to U-139 and U-153 to U-159). If they are 200ns RAMs, they will probably have a "-2" number suffix. If they are not 150ns RAMs, the XLR8er board will not be able to run at full speed.

   Check for the chip labeled 74LS245P and replace it if necessary. Note that the suffix letter may be different. It will be adjacent to the RAM chips and may be in socket U-117.

9. Close the pan assembly slightly until the other end of the 40 pin cable that you plugged into the Z80 socket reaches the empty 40 pin socket on the XLR8er board. Carefully plug the 40 pin ribbon cable connector into the socket.

10. Now re-connect the ***disk drive ribbon cable connector*** (this is the part that everyone forgets) and close the pan assembly.

11. Follow the above instructions in reverse order; put the case back on the chassis and your done.

Introduction

The sections in this chapter describe the support software that you received with your XLR8er board. Since there are several operating systems currently running on the Model 4 series of computers, each one requires a different set of support utilities. Even though the utilities may differ from operating system to operating system in their syntax and implementation, they all perform the same function, that is to provide a user friendly interface to some of the internal functions of the XLR8er board.

In a power up or reset condition the XLR8er board is operating at its slowest speed. This allows the Model 4 to read its relatively slow boot ROMs and boot the operating system from the floppy disk. Once the system has booted, the supplied software utilities allow you to select the speed at which you want your Model 4 to run. Some of the operating system utilities also allow you to tell the operating system about the extra RAM that is available, so programs that use the bank switching of RAM can utilize the extra 256K bytes of RAM provided on the XLR8er board.

Currently, RAMDISK software utilities are available for those users using TRSDOS 6.x and Montezuma's CP/M 2.3x. These utilities are further described under their appropriate sections below.

TRSDOS 6.x utilities

This software section is divided into two parts: "Getting Started" for the novice; and "The Nitty-Gritty" for the user who wants to know more about the inner workings of the software. The *Getting Started* section will give only a high-level description of the relevant software with little computereze, and lots of examples. The *Nitty-Gritty* section, however, assumes that the user has some experience in assembly language programming, and access to a Model 4 Technical Reference Manual.

Before you run any software, copy the following files to your boot disk: FIXALL/FLT, FIXBANK/CMD, SET180/CMD, CFGRD/JCL, SETUP/JCL and RAMDISK/DCT. If you do not know how to copy files, refer to your TRSDOS owner's manual for instructions. Also, if you have use for it, copy DISAS/CMD to an applications disk. You should have backup copies of this software at ALL times; you should NEVER operate directly from the distribution diskette.

Next, create a working system disk with the above software on it to play with. You will want to run the example programs to get a feel for the new hardware and software. If during this break-in period you make a catastrophic error, you will not lose much. You will also be more free to experiment with new system configurations without the overhead of your usual system.

Getting Started

For the novice user, there are only four pieces of software to keep track of. These four programs are described below.

FIXALL

With no additional software, the XLR8er can run 70% faster than a normal Model 4. Getting that last 30% requires some extra software. Due to slow keyboard hardware, the XLR8er must run slower when accessing the keyboard; it can then be sped back up for all other tasks. FIXALL must be present when the XLR8er is run at maximum speed.

FIXALL also acts as a filter to the Model 4 interrupts so that normal RAM can be imaged for the TRSDOS interrupt task processor. Thus, **FIXALL must be present when the FIXBANK software module is installed to utilize the extra eight banks of XLR8er RAM**.

SET180

Above, we talked about changing the speed of the XLR8er. How is this done? There aren't any little switches to flip. The XLR8er speed is controlled by running a program called SET180 with certain parameters. The meaning of these parameters doesn't matter now, just so we know what they are.

FIXBANK

As you know, there is 256K of memory on the XLR8er in addition to the 64k or 128k you already have installed in your Model 4. If you have a 128k machine, its extra 64K is partitioned into two 32K "banks" which may be used one at a time. FIXBANK similarly partitions the XLR8er's 256K into eight "banks" of 32K which may be used one at a time. In this way, com-

patibility is maintained with all software which may use these extra banks of memory. FIXBANK must be resident if you plan to use RAMDISK to build a memory disk drive which uses any of the XLR8er's extra memory.

RAMDISK

Like Tandy's MemDISK, RamDisk enables the user to set aside a section of memory and make it act like a super-fast disk drive. Unlike MemDISK, RamDisk is capable of 319K memory disk drives. If the RamDisk is used as the system drive, performance is drastically increased again. For RamDisk installation, see the "Installation and Operation" section below.

Some installation examples

To install FIXALL and set the XLR8er fo full speed, issue the following command:

```
DO SETUP
```

Let's take a look at what the SETUP JCL file does. Here is a listing of SETUP/JCL:

```
SET180 (M=1,I=1,R=80)
FIXBANK
SET *WS FIXALL/FLT
FILTER *KI *WS
SET180 (M=0)
```

The first line of this JCL file sets the speed of the XLR8er as fast as it will go without the FIXALL software. Next, FIXBANK is installed to provide access to the extra 256K of memory. Now comes the tricky part; the FIXALL filter is assigned to the device *WS, and then the keyboard is filtered with *WS. These two lines accomplish the installation of FIXALL. Finally, the last line moves the XLR8er up to full speed.

Even though the above JCL file does not take an inordinate amount of time to execute, it would be annoying to go through it every time you booted. It is for this reason that FIXALL and FIXBANK have been given the ability to be SYSGENed. That is to say, once they have been installed, you can issue the SYSGEN command and from then on, they will be automatically loaded and set up when you boot. Then, all that you need do is issue a

"SET180 (M=0,I=1,R=80)" to set the speed to maximum. This can also be done automatically by issuing the command;

```
AUTO SET180 (M=0,I=1,R=80)
```

Now let's take a look at how to set up some useful RamDisk configurations. Type the following command at TRSDOS Ready;

```
DO CFGRD
```

Wow, that sure took a while! Good thing we have the XLR8er or it would have taken all day! Type CAT or FREE and notice the new arrangement of the drives. RamDisk has been installed as the system drive, and the old drive :0 is now drive :2. Also notice that there is STILL well over 150K free on RamDisk; quite a step up from a diminutive 63K! Now that RamDisk is set up, other useful files can be copied to it. Remember, a file on RamDisk can be accessed or executed 5-10 times faster than the same file on a floppy disk. Keep in mind that any file contained in the RamDisk will go away without power, so any file that has been created or modified on the RamDisk must be copied to a floppy BEFORE you power down your system. Here is a listing of the CFGRD/JCL file.

```
. Configure RamDisk for use as system disk.
SYSTEM (DRIVE=2,DRIVER="RAMDISK")
E
3
8
Y
BACKUP SYS/SYS:0 :2 (I,S)
COPY FORMAT/CMD.UTILITY:0 :2
COPY BACKUP/CMD.UTILITY:0 :2
SYSTEM (SYSTEM=2)
```

The first line tells TRSDOS that drive :2 is to be configured for RamDisk. The next four lines answer questions that RamDisk ask. They state that the RamDisk is to be <E>nabled, start at bank number 3, use 8 contiguous banks, and finally, that the memory should be formatted by RamDisk. This file will work for any two drive Model 4; 64K or 128K. If you have 128K, you might want to change the 3 to 1, and the 8 to 10; this would produce a RamDisk that starts at bank 1 and extends through bank 10. In this configuration, RamDisk will hold 319K! If you have more than two disk drives, change all the 2's to the number of your next available drive slot.

The nitty-gritty

Several pieces of software have been included with your XLR8er board. Some of them are essential to its efficient operation, the rest are merely nice to have around. The "essential" software is as follows:

1. **FIXALL** - A filter to control keyboard and interrupt wait states.

2. **SET180** - A utility to set the HD64180's wait states and refresh cycle time.

The "nice" software consists of the following programs:

1. **FIXBANK** - A new module that adds 10-bank capability to @BANK (also needed for the operation of RamDisk).

2. **RAMDISK** - Similar to MemDisk. Capable of 319K memory disk.

3. **DISAS** - A simple 64180 disassembler.

4. **BANKSTAT** - Lets you determine which banks are in use.

What is the purpose of this software?

FIXALL

FIXALL is a two-pronged attack against problems caused by the keyboard hardware at the higher speeds of the XLR8er. The problem present is a slow PAL chip which cannot strobe the keyboard matrix in the shorter memory read cycle of the 64180. It is therefore necessary to insert a number of memory wait states before every access to the keyboard hardware. Generally, these accesses occur in two places: the resident *KI keyboard driver, and the KFLAGS$ scanner.

The *KI accesses can be trapped by a filter which inserts wait states before it passes control to *KI, and removes them again after *KI has returned. Controlling accesses by the KFLAGS$ scanner is a much more difficult procedure. Since the location of the KFLAGS$ scanner is not documented, it is necessary to trap all interrupts, insert wait states, call the interrupt processor, and then remove the wait states. Admittedly, this is not a

particularly elegant software solution, but it is one of the only ways to make sure the software is portable between different versions of TRSDOS.

SET180

Since the HD64180 can have zero to three memory wait states; one to four I/O wait states; and a refresh period every 10, 20, 40, or 80 clock cycles, provisions must be made for changing these operating parameters. SET180 is provided for this purpose. The above 64180 parameters are altered simply by invoking SET180 with different command line parameters enclosed in parentheses. The exact syntax of this utility is described below in the installation and operation section.

FIXBANK

Since an extra 256K of memory is included on the XLR8er board, some way must be provided to access it. The only logical way of doing this is to add eight more banks of capability to the @BANK supervisor call. This provides a standard, already documented way of accessing the additional eight 32K banks created by the 256K of RAM. These eight banks act exactly like the three already present in a 128K Model 4 with only one exception: the keyboard and video shadow RAM cannot be switched in on top of them. This should never be a problem since all access to the shadow RAM is done by the system with bank zero switched in.

The eight new banks of RAM are "switched in" by changing two registers in the 64180's Memory Management Unit (MMU). The 64180 directly addresses 64K of logical memory space from a physical memory space of 512K. The 64K of logical memory space is composed of three smaller address spaces called Common Area 0 (CA0), Bank Area (BA), and Common Area 1 (CA1). CA0 starts at logical address 0000H and extends to some 4K byte boundary (including zero) less than 64K. The BA starts where the CA0 ends and extends for a number of 4K byte boundaries (including zero) that places its end less than 64K. CA1 starts where the BA ends and extends to the end of the logical address space (FFFFH or 64K).

When banks 0, 1, or 2 are switched in, FIXBANK sets up the MMU to have CA1 occupy the whole logical address space. (No CA0 or BA) If a bank from the 256K of on-board RAM is selected, (banks 3-10) the MMU is reconfigured with CA0 starting at 0000H and extending to 7FFFH and CA1 extending from 8000H to 0FFFFH. In this way, the lower 32K (0000H -

7FFFH) is left undisturbed, and the upper 32K can be moved over a 32K section of the on-board 256K. It is for this reason that shadow RAMing cannot occur in banks 3-10; all the Model 4 bank switching and shadow RAMing circuitry is logically bypassed.

RAMDISK

Since Tandy's MemDisk driver will only handle a two-bank memory disk, a new driver had to be written to use the other eight banks of RAM available. The banked (319K) memory disk is now possible using RamDisk, not a paltry 63K! A detailed discussion of the internal workings of RamDisk is beyond the scope of the manual, and will therefore not be addressed.

Installation and operation of software

FIXALL and SET180

Since FIXALL and SET180 are so closely tied together, and have similar functions, their installation and operation will be described together.

When booted, your XLR8er will initialize itself with three memory wait states, four I/O wait states, and a refresh period every 10 clock cycles (About 23% slower than a normal 4Mhz Z80 system). This can be immediately improved to 69% faster than normal speed by setting the memory wait states to one, and the refresh period to every 80 clock cycles. Slow keyboard hardware prevents the last memory wait state from being removed without the addition of certain software. The software that fixes this problem is as mentioned above, FIXALL. FIXALL is installed in the following way:

```
SET *WS FIXALL/FLT
FILTER *KI *WS
```

The above commands set the FIXALL filter to the device *WS, and specify that there should be two memory wait states during all keyboard hardware accesses. Once this filter is installed, the last memory wait state may be eliminated by issuing the following command:

SET180 (*MWAIT*=0,*IOWAIT*=1,*REFRESH*=80)	
SET180 may have any or all of the following parameters:	
MWAIT=N	Sets memory wait states to N. N may be 0-3
IOWAIT=N	Sets I/O wait states to N. N may be 1-4
REFRESH=N	Sets refresh period to N. N may be 10,20,40,80.
STATUS	Causes SET180 to display the current status of memory waits, I/O waits, and refresh period.
Abbrev: M=MWAIT, I=IOWAIT, R=REFRESH, S=STATUS	

Since FIXALL must be installed every time your machine is booted, it has been written so that it may be SYSGENed like COM/DVR, KSM/FLT, or FORMS/FLT.

FIXBANK

FIXBANK is simple to install. Simply type "FIXBANK" at the TRSDOS prompt, and the patch will be installed and located in low memory. FIXBANK must be resident if you plan to use RamDisk with banks 3-10, and it may be SYSGENed.

RAMDISK

RAMDISK is installed in a manner very similar to the installation of Tandy's MemDisk. The appropriate incantation is:

```
SYSTEM (DRIVE=2,DRIVER="RAMDISK/DCT")
```

This will set up drive "dd" as the RamDisk. Once the above command is executed, RamDisk will respond with several questions:

1. "*Do you wish to <E>nable or <D>isable the RamDisk?*"

The user should enter an "E" when installing the RamDisk, and a "D" when removing it.

2. "*Enter the starting bank number. (1-10)*"

This is the first bank of memory that RamDisk will use.

3. "*How many contiguous banks?*"

The only constraint here is that the highest numbered bank used must be less than or equal to 10.

4. Either "*RamDisk contains data. Format over it?*" or "*RamDisk is unformatted. Do you wish to format it?*"

The user should input a "Y" to format the RamDisk, or an "N" to abort. If "N" is entered, RamDisk will exit the command prompt having done nothing.

When RamDisk is through formatting and verifying memory, it will print out how many cylinders there are in the RamDisk and exit to the command prompt. If any errors occur during the formatting process, RamDisk will print out an error message and abort. If a "disable" request was given to RamDisk, it will print out a message saying whether or not it was able to reclaim the low memory it previously occupied.

Some nice-to-have-around utilities

DISAS

DISAS is a simple Z80/64180 in-memory disassembler that will output to the display, or to both the display and printer. DISAS is invoked by simply typing its name at the command prompt with no arguments. DISAS will then ask where to start disassembling. Type in the hex address of the code you wish disassembled, and press <ENTER>. Twenty instructions will be disassembled from that location and displayed on the screen. A list of options will be displayed at the bottom of the screen. You may then continue disassembling, set a new disassembly address, or exit the program.

BANKSTAT

If you ever want to know which memory banks are free or in use, simply execute "BANKSTAT" and it will display a chart telling which banks are available.

Montezuma Micro CP/M 2.2 Utilities

Most of the CP/M utilities are designed to work ONLY with Montezuma's BIOS version 2.3x, and if the extended RAM disk is wanted, the only way to get it is to be running that version. Montezuma Micro has a very reasonable update policy, and there are many advantages to running the required version.

If you don't have version 2.3x, you can still run with the XLR8er installed, and you will be able to initialize the hardware to run at as low as 1 wait state, but you will be losing a good deal of the functionality of the XLR8er.

INIT-180

Works with any BIOS version, even CP/M+

This program will perform the initialization of the XLR8er's wait states and RAM refresh rates. If you do not know what these are, don't worry, as you can find the optimum parameter values without knowing what they do.

To invoke, enter the command, followed by the three required parameters. For example;

```
INIT-180 M1 I1 R3
```

would set the memory wait states at 1, the I/O wait states at 1, and the RAM refresh rate at once every 80 states. These are the values that would give you the maximum performance available from the board, without running 0-WAIT.

If the system doesn't work properly with those values, then re-boot, and try higher values for M and I, and lower values for R until the system runs correctly. If INIT-180 is invoked with no parameters, a help screen will be displayed. The allowed ranges for the parameters will also be displayed.

Even though 0 is allowed for memory wait states, it won't work if set from this program (see 0-WAIT).

0-WAIT

Only works with BIOS version 2.3x

If you were able to use the 1 wait state parameter in the INIT-180 command line, then you may want to try running this program to let you get down to 0 wait states for memory access. That will increase your performance to the maximum possible.

To invoke the program, just enter 0-WAIT at the command prompt. There are no parameters needed. There is a danger involved in using this program, in that if you use the SYSGEN option of CONFIG to save any changes to disk after 0-WAIT was invoked, then the system image on the disk will be damaged, and you will not be able to re-boot from that disk. The program will remind you of that every time it is invoked.

If you run with 0-WAIT, then you do not need to run INIT-180 except under certain circumstances involving the RAM disk.

MDSK-180

Only works with BIOS version 2.3x

This program will set up a 256 RAM disk (drive M:) using all of the memory on the XLR8er. To invoke it, just enter MDSK-180 at the command prompt. It actually will figure out what size image you are using, and relocates itself over the old RAM disk driver. If you use the SYSGEN option of CONFIG to save the image to disk after it was invoked, then the new RAM disk driver is saved permanently to disk, and you will never need to run this program again.

If you save the image to your boot disk in this fashion, MAKE SURE that 0-WAIT had not been previously invoked. Due to the fact that a patch installed in the BIOS has to be of the same size or smaller than the code you are writing over, it was not possible to add the logic necessary to allow utilizing the extra 64k on 128k machines. All is not lost however, as you can still run Monte's windows in that 64k. (See the notes which follow).

M-FORMAT I

Only works with BIOS version 2.3x

Once the RAM disk drive has been installed (either by invoking MDSK-180, or booting with a disk that had the code saved in the system image), it is necessary to format it. Just enter M-FORMAT I from the command prompt. Any parameter other than I will cause the program to abort the initialization.

There is an advantage to having a separate format utility for the RAM disk instead of having the BIOS automatically format it every time you boot (like Montezuma does it). It allows you to perform a cold boot without losing what was on the RAM disk, as long as power was not lost from the system. To restore the RAM disk after a cold boot, just run INIT-180 with whatever parameters you normally use, and if the new RAM disk driver was not saved to the system image, also run MDSK-180. Do NOT run M-FORMAT again. If you log onto the RAM disk, you will see that all the files that were there previously are still there.

Notes

It was mentioned previously that you can still run Monte's Windows in the extra 64k on 128k systems. Due to the fact that Monte's windows still thinks that it is running on a 128k standard Model 4, it goes in and alters the Disk Parameter Block for drive M:, and removes 32k from the available space. Since the DPB for the new RAM disk is located in the same place as the old one, you'll notice that after installing Monte's Windows your RAM disk shrunk 32k.

In actuality, ALL of the space is still available, and all that is necessary to reclaim the lost space is to run MDSK-180 after you install Monte's Windows. You will need to do this even if you have saved the system image to disk with the new RAM disk driver, and if you do not run MDSK-180, AND save changes to disk after having installed Monte's Windows, the lost 32k is lost on all subsequent boots.

Since a lot of the CP/M specific software performs the same functions as the TRSDOS software, be sure to read the TRSDOS section of this manual for further enlightenment as to the reason for setting wait states, refresh rates, etc.

Introduction

This section explains how to install the optional I/O kits for adding two RS232 ports and an expansion port to your XLR8er board. The I/O kits are very easy to install; they require no soldering for installation. If you installed your XLR8er board, you will be able to install the I/O kits.

Installation

Refer to Figure 1.4 for these installations.

RS-232 Option

1. Remove the XLR8er board from your system if it is already installed.

2. Insert the chip labeled "1488" into the socket labeled U22. Orient the chip so that the "dot" on the chip faces the side of the socket with the notch.

3. Insert the chip labeled "1489" into the socket labeled U23. Orient the chip so that the "dot" on the chip faces the side of the socket with the notch.

4. Plug in the RS232 power connector supplying the board with +12v/-12v power onto the 4 pin header labeled "P3". (Refer to the Figure P3 connector diagrams for connector pin assignments.)

5. Plug the special RS232 breakout cable onto the P2 ribbon cable header. For the pinout of the two RS232 connectors on the end of this cable refer to Figures RS1 and RS2.

6. Feed the appropriate cable out of the computer enclosure through the feed-through holes in the bottom of the enclosure.

7. Re-install your XLR8er board.

Using the RS-232 ports

The operation of the RS-232 ports will be described in documentation which accompanies that option. Pinouts of the appropriate headers and cables which support this option are located in the APPENDIX.

Expansion port option

This option assumes the use of your own expansion port connecting cable.

1. Remove the XLR8er board from your system if it is already installed.

2. Insert the 74HC138 chip into the 16 pin socket labeled U19. Orient the chip so that the "dot" on the chip faces the side of the socket with the notch.

7. If you plan to use the I/O expansion port, plug one end of your 40 pin ribbon cable onto the "P1" header.

8. Feed the appropriate cable out of the computer enclosure through the feed-through holes in the bottom of the enclosure.

7. Re-install your XLR8er board.

Using the expansion port

The expansion port provided at the P1 header connector is of the same pinout as Ciarcia's SB180™ as featured in Byte™ magazine. All the signals needed for experimental circuits or add on circuit boards that Ciarcia has designed can be plugged into this header. The only difference in the expansion port as compared to the expansion port on the SB180 is that the EXP SEL0^ and EXP SEL1^ mapping has been changed to avoid any conflicts with the I/O map of the Model 4. EXP SEL0^ goes true (low) any time an I/O address of 40H to 5FH is accessed. EXP SEL1^ goes true (low) any time an I/O address of 60H to 7FH is accessed. The pinout of the P1 header which supports this option is located in the APPENDIX.

Introduction

This chapter describes some of the aspects of programming the XLR8er board in a machine language environment. For most users this section will be only for reference as the software supplied with the board takes care of setting up the 64180 chip in the right mode for the proper operation of standard application software. If you would like to delve further into the world of the 64180 it is suggested that you obtain a copy of Hitachi's 64180 Technical Data Book.

New instruction set

The HD64180 is compatible with the Z80 instruction set, but also has several new instructions including a hardware 8x8 multiply. The new instructions are summarized below:

MLT - Multiply

The MLT performs unsigned multiplication on two 8 bit number yielding a 16 bit result. MLT may specify BC, DE, HL or SP registers. In all cases, the 8-bit operands are loaded into each half of the 16-bit registers and the 16-bit result is returned in that register.

IN0 g,(m) - Input, Immediate I/O address

(Use this instruction to access the 64180 internal registers.)

The contents of the immediately specified 8-bit I/O address are input into the specified register. When I/O is accessed, 00H is output in the high-order address bits automatically.

OUT0 (m),g - Output, immediate I/O address

(Use this instruction to load the 64180 internal registers.)

The contents of the specified register are output to the immediately specified 8-bit I/O address. When I/O is accessed, 00H is output in the high-order address bits automatically.

OTIM, OTIMR, OTDM, OTDMR - Block I/O

(Use these instructions for block moves to the 64180 internal Registers.)

The contents of memory pointed to by HL is output to the I/O address in (C). The memory address (HL) and I/O address (C) are incremented in OTIM and OTIMR and decremented in OTDM and OTDMR respectively. The B register is decremented. The OTIMR and OTDMR variants repeat the above sequence until register B is decremented to 0. Since the I/O address (C) is automatically incremented or decremented, these instructions are useful for block I/O initialization. When I/O is accessed, 00H is output in the high-order address bits automatically.

STIO m - Test I/O Port

The contents of the I/O port addressed by register C are ANDed with the immediately specified 8-bit data and the status flags are updated. The I/O port contents are not written (non-destructive AND). When I/O is accessed, 00H is output in the high-order address bits automatically.

TST g - Test Register

The contents of the specified register are ANDed with the accumulator (A) and the status flags are updated. The accumulator and specified register are not changed (non-destructive AND).

TST m - Test Immediate

The contents of the immediately specified 8-bit data are ANDed with the accumulator (A) and the status flags are updated. The accumulator is not changed (non-destructive AND).

TST (HL) - Test Memory

The contents of memory pointed to by HL are ANDed with the accumulator (A) and the status flags are updated. The memory contents and accumulator are not changed (non-destructive AND).

SLP - Sleep

(Probably never used on the Model 4)

The SLP instruction causes the 64180 to enter SLEEP low power consumption mode.

Some 64180 internal registers

There are 47 internal registers or I/O ports on the 64180, mapped at address 00H to 37H. For a detailed explanation of all of these registers refer to Hitachi's HD64180 Microprocessor User's Manual. The discussion of internal registers here will be limited to the ones that have a direct effect of the performance in the operation of the Model 4. For most all operations of the XLR8er board on the Model 4 the following registers have a direct bearing on the speed at which the XLR8er board runs:

DCNTL register - port address 32H

This register controls how many wait states are inserted into memory or I/O reads/writes. By changing the values loaded into this register, the speed of the Model 4 can be 29% slower to 100% faster as compared to the Z80 version. One note of caution should be mentioned here; if all the memory wait states are removed the keyboard operation may become unreliable. Thus, make sure that any program that accesses the keyboard puts in one wait state before accessing the keyboard port and then removes it upon completion. The bit assignments and mode tables of the DCNTL register are listed in the following tables.

```
            7    6    5    4    3    2    1    0
bit ----> MWI1 MWI0 IWI1 IWI0 DMS1 DMS0 DIM1 DIM0
reset --> 1    1    1    1    0    0    0    0
          |_____|    |_____|    |_____|    |_____|
                |          |          |          |___ DMA channel 1
                |          |          |               I/O memory
                |          |          |               mode select
                |          |          |___ DREQ select
                |          |____ I/O wait state insertion
                |____ Memory wait state insertion

        |  The number of                        |  The number of
MWI1,0  |   wait states                  IWI1,0 |   wait states
--------|---------------                 -------|----------------
  0,0   |       0                          0,0  |       0
  0,1   |       1                          0,1  |       2
  1,0   |       2                          1,0  |       3
  1,1   |       3                          1,1  |       4
```

By examining the above tables it can be seen that after a reset the 64180 is running with the maximum number of memory and I/O wait states. By writing values other than 1's to the DCNTL register you can dynamically change the speed at which the 64180 does memory and I/O operations. Section 5.4 will provide information that you can use to use your Z80 assembler to write the machine code to modify the values of this register.

RCR - Refresh Control Register - Internal port address 36H

One of the reasons that the 64180 is so much faster than the Z80 is that it has programmable dynamic RAM refresh control. With the RCR one can program the 64180 to wait 10, 20, 40, or 80 machine cycles until it performs a dynamic RAM refresh. In contrast the Z80 does RAM refresh every M1 cycle. In the Model 4 computer the RCR register can be programmed to provide dynamic RAM refresh every 80 machine cycles without causing any RAM integrity problems. The bit mapping of the RCR register is shown below.

```
                7    6    5    4    3    2    1    0
  bit --->   REFE REFW ---- ---- ---- ---- CYC1 CYC0
reset --->    1    1                        0    0
              |    |                        |_____|
              |    |          Cycle select ____|
              |    |____ Refresh wait states
              |____ Refresh enable


     CYC1,0  |  Interval of refresh cycle
     --------|--------------------------
       0,0   |       10 states
       0,1   |       20
       1,0   |       40
       1,1   |       80
```

Leave bits 6 and 7 in the RCR control register set to a one. Bits 0 and 1 can be changed to any value as seen in the table, but the best performance of the Model 4 will be realized when they are both set to ones.

Using the 64180 instruction set with a Z80 assembler

Although it is beyond the scope of this manual to provide all the op codes of the new 64180 instructions and how they may be used, an explanation of how to use some of the new instructions with your existing Z80 assembler is in order. The instruction that will be used in the following example is the "OUT0 (m),g" so that you can experiment with loading the various internal registers with desired values. In this example we will be loading the internal register, DCNTL, with a value that is stored in the accumulator (A). Using a 64180 assembler the syntax would be as follows;

```
DCNTL   EQU     32h             ;Define DCNTL
                                ; control regs address
        .
        .
        LD      A,01010000B     ;Set 1 memory and
                                ; 1 I/O wait state
        OUT0    (DCNTL),A       ;Load the DCNTL regs. with A
```

The above code would load the value in A (01010000b or 50H) into the DCNTL register (port address 32H). The same operation can be accomplished by using a Z80 assembler by substituting the 64180 specific codes with define bytes, or MACRO's. The following example will use define bytes to do the same thing as the above code. Before attempting the coding the actual hex op codes must be found in the 64180 manual. For a OUT0 (m),A instruction the hex bytes that the assembler would produce

are EDH, 39H, xxH where xx represents the hex address of the port. Now let's try the actual coding.

```
DCNTL   EQU     32H             ;Define the port address
        .
        .
        LD      A,50H           ;Bit mask for 1 memory
                                ; and 1 I/O wait
        DB      0EDh,39h,DCNTL  ;Same as - OUT0 (DCNTL),A
```

A few more notes on port I/O

As mentioned in several of the sections and chapters above, with the XLR8er board port addresses can overlap between the ports internal to the 64180 (addresses 00H - 3FH) and ports external to the 64180 occupying the same port addresses. The "trick" as it were is to make sure that when port I/O is done to external ports in this address range, that something other that 00h appears on the upper address lines. When doing port I/O with the standard Z80 instructions set, most I/O instructions place the value from an internal register on the upper 8 address lines. For example the instruction;

```
        OUT     (C),A           ;output value from A to port (C)
```

places whatever value is in the B register on the upper 8 address lines. As long as this value is not 00h the external port will be accessed instead of the 64180 internal port. For all internal port I/O use the new 64180 OUT0, and IN0 instructions and the upper 8 address lines will always be forced to 00H.

Memory Management Unit (MMU)

The MMU of the 64180 is one of its most unique, powerful, and confusing features. This section will explain the techniques involved in using the MMU, and will provide some programming examples.

The 64180 divides the memory that is accesses into three (3) separate areas called **Common Area 0**, **Common Area 1**, and **Bank Area**. How these areas of memory are physically mapped into a logical 64K of RAM is determined by how the internal registers are set up.

The first thing to clarify is what is meant by "physical" memory map versus "logical" memory map. The 64180 can address a physical memory map of 512K but only in 64K blocks (since the 64180 is compatible with the Z80 instruction set we can't load HL indirect from address 4FFFFH.) The

following memory map might represent a typical configuration of how memory could be mapped.

```
Physical Address            MEMORY            Logical Address

  4FFFFH -------->                                  FFFFH
                  | Common Area 1 |   4K
  4F000H -------->                                  F000H
  0EFFFH -------->                                  EFFFH
                  |  Bank Area    |   52K
                  |               |
  02000H -------->                                  2000H
                  | Common Area 0 |   8K
  00000H -------->                                  0000H
```

In the above memory map we can see there is a physical gap between the top of Bank Area and Common Area 1, but the 64180 thinks that it is a contiguous 64K block of RAM ,thus logically we are addressing 64K of RAM but physically we have a gap between Bank Area and Common Area 1.

To understand how to move the 64180 memory around in the 512K of available space, one needs to first understand how the three (3) internal registers associated with the MMU work. The first register to examine is the Common/Bank Area Register referred to as CBAR from now on.

```
MMU Common/Bank Area Register (CBAR : I/O Address = 3AH)

     bit  7    6    5    4    3    2    1    0
          CA3  CA2  CA1  CA0  BA3  BA2  BA1  BA0
```

CA3 - CA0 (bits 7-4)

CA specifies the start address (on 4K byte boundaries) for Common Area 1. ***This also determines the last address of the bank area.*** After a hardware reset all of the CA bits are set to 1.

BA3 - BA0 (bits 3-0)

BA specifies the start address (on 4K byte boundaries) for the Bank Area. ***This also determines the last address of the common area 0.*** After a hardware reset all of the BA bits are set to 0.

Thus in the above memory map example CBAR would be loaded with a F2H. The most significant 4 bits of CBAR tell the 64180 where the Common Area 1 is LOGICALLY located. In this case we want the Common Area 1 to start at F000H. The least significant 4 bits of CBAR tell the 64180 where the Bank Area is LOGICALLY located. In this case we want the Bank Area to start at address 2000H. Also note that since the Bank Area starts at address 2000H, Common Area 0 ends at address 1FFFH, and since Common Area 1 starts at address F000H, Bank Area ends at address EFFFH. So CBAR really defines 4 distinct addresses: End of Common Area 0 - Start of Bank Area - End of Bank Area - Start of Common Area 1.

Think of CBAR as setting up the way memory is logically partitioned between the three areas of Common Area 0, Bank Area, and Common Area 1. Since, in the example, CBAR would have been loaded with an F2H, Common Area 0 starts at 0000H and extends to 1FFFH - Bank Area starts at 2000H and extends to EFFFH - Common Area 1 starts at F000H and extends to FFFFH. (Remember, these are logical addresses, not the address that would actually be put out on the address bus.)

Now that the logical memory map has been set up how do we tell the 64180 where in physical memory to map the different partitions? This is where the other two 64180 MMU registers come in to play.

MMU Common Base Register (CBR : I/O Address = 38H)

bit	7	6	5	4	3	2	1	0
	-	CB6	CB5	CB4	CB3	CB2	CB1	CB0

CBR specifies the base address (on 4K boundaries) used to generate a 19 bit physical address for Common Area 1 accesses.

After a reset all of the CBR bits are set to 0.

MMU Bank Base Register (BBR : I/O Address = 39H)

bit	7	6	5	4	3	2	1	0
	-	BB6	BB5	BB4	BB3	BB2	BB1	BB0

BBR specifies the base address (on 4K boundaries) used to generate a 19 bit physical address for BanK Area accesses. All bits of BBR are set to 0 after a hardware reset.

In the above memory map example, notice that the Bank Area does not have an offset or base added to it the generate the Bank Area address thus in out example BBR would be loaded with a 00H. The Common Area 1 however, does have an offset of 40000H. So when the 64180 generates a logical address of F000H or above we want the physical address that is generated to have 40000H added to it. To do this load the CBR register with a value of 40H.

BBR and CBR registers allow us to define the location of Bank Area and Common Area 1 anywhere in the 512K of the physical memory map. Note that Common Area 0 always starts at physical and logical address 0000H. Below are some typical mapping configurations and address generation examples. In the partition examples it can be seen that Common Area 1 and Bank Area can overlap and that Common Area 1 and Bank Area can be freely relocated (on 4K boundaries). In this manner Common Area 0 and/or Bank Area could be completely mapped out.

THE FOUR POSSIBLE PARTITIONS FOR LOGICAL MAPPING

```
1                2                3                4
 _____________    _____________    _____________    _____________
| Common      |  | Common      |  | Common      |  |             |
| Area 1      |  | Area 1      |  | Area 1      |  |             |
|-------------|  |             |  |             |  |             |
| Bank        |  |-------------|  |-------------|  | Common      |
| Area        |  |             |  |             |  | Area 1      |
|-------------|  | Bank        |  | Common      |  |             |
| Common      |  | Area        |  | Area 0      |  |             |
| Area 0      |  |             |  |             |  |             |
|_____________|  |_____________|  |_____________|  |_____________|
```

Address examples

```
LET CBAR = F2H ( as in the example above )

     IF CBR = 50H and BBR = 20H

      Partitions
      Common Area 0 ===> 00000H to 01FFFH
      Bank Area     ===> 22000H to 2EFFFH
      Common Area 1 ===> 5F000H to 5FFFFH

     Logical Address   Physical Address
       0 000           0 000 + 0 0 000 = 00000H (Common Area 0)
       2 010           2 010 + 2 0 000 = 22010H (Bank Area )
       E FFF           E FFF + 2 0 000 = 2EFFFH (Bank Area )
       F 001           F 001 + 5 0 000 = 5F001H (Common Area 1)
```

```
LET CBAR = F0H

     IF CBR = 70H and BBR = 00h

     Partitions:
     Common Area 0 ===> not mapped (Bank Area overlaps it)
     Bank Area     ===> 00000H to 0EFFFH
     Common Area 1 ===> 7F000H to 7FFFFH

     Logical Address   Physical Address
       0 000          0 000 + 0 0 000 = 00000H (Bank Area)
       4 02C          4 02C + 0 0 000 = 0402CH (Bank Area)
       E FFF          E FFF + 0 0 000 = 0EFFFH (Bank Area)
       F 000          F 000 + 7 0 000 = 7F000H (Common Area 1)
       F 877          F 877 + 7 0 000 = 7F877H (Common Area 1)

     Then change CBR = 60H and BBR = 20H

     Partitions:
     Common Area 0 ===> not mapped
     Bank Area     ===> 20000H to 2EFFFH
     Common Area 1 ===> 6F000H to 6FFFFH

     Logical Address   Physical Address
       0 000          0 000 + 2 0 000 = 20000H (Bank Area)
       E FFF          E FFF + 2 0 000 = 2EFFFH (Bank Area)
       F 000          F 000 + 6 0 000 = 6F000H (Common Area 1)
       F FFF          F FFF + 6 0 000 = 6FFFFH (Common Area 1)
```

Application

An interesting note as to the advantages of coding to utilize the 64180's extended instruction set was brought to our attention by our CP/M programmer. He is running a non-standard version of Monte's BIOS, modified to directly support the XLR8er and the Z operating system, and in one simple change in the video routines, was able to speed up a rewrite of the screen from 1.3 seconds to 1.0 seconds. This is a change that shows up every time anything is written to the screen. The code is included here as an example of optimizing code for the XLR8er.

Original code

```
PUSH    BC
PUSH    DE
LD      BC,0F800H       ;Video RAM base
LD      D,C             ;Row # to DE
LD      E,L
LD      C,H             ;Col # to C
LD      H,D             ;row also to HL
ADD     HL,HL           ;HL=row # * 4
ADD     HL,HL
ADD     HL,DE           ;HL=row # * 5
ADD     HL,HL           ;HL=row # * 80
ADD     HL,HL
ADD     HL,HL
ADD     HL,BC           ;Add video base, column #
POP     DE
POP     BC
RET
```

New code

```
PUSH    BC
LD      BC,0F800H       ;Ram base
LD      C,H             ;Col to C
LD      H,80            ;* 80
MLT     HL
ADD     HL,BC           ;Add base
POP     BC              ; & col #
RET
```

As demonstrated in this section the 64180 is a very powerful and versatile chip. We only touched on the high points of the internal registers of the 64180 leaving out the RS-232 registers and the two 16-bit internal timers. If you wish to learn more about all of the 64180 internal registers we strongly suggest that you obtain a copy of the 64180 Technical Reference Manual from Hitachi.

Advanced system setup

The utilities supplied with the XLR8er board combined with the advanced JCL features of TRSDOS, provide a powerful environment in which to run your MODEL 4. The system setup JCL file provided on the distribution diskette provides a system setup adequate for most users. However, some special applications may require additional hardware and/or software system setup. This document discusses advanced system setup configurations and provides additional insight to the use and function of some of the XLR8er utilities.

TRSDOS memory usage

The TRSDOS operating system uses two areas of memory to hold system device drivers, filters, and other memory modules. These areas are (1) low memory, inside the operating system, and (2) in high memory above the application memory area. The space reserved in low memory is relatively small and is used for things which are small enough, or cannot reside in high memory for some reason.

Objects which are commonly placed in low memory are: the communications line driver, and hard disk drivers. These modules are capable of loading themselves into high memory, as well as low memory. However, this relocation will only happen when there is insufficient space available in the low memory area.

If a driver module uses any of the extra memory in the Model 4, it cannot be loaded into high memory; otherwise, when the module switched memory banks, it would switch the memory bank in which it was executing out of the memory map. All support software provided with the XLR8er (FIXBANK, FIXALL, and RAMDISK) exhibits this characteristic. Therefore, these programs must occupy low memory only. Once these programs are loaded, however, there is little if any low memory left over for other programs. To remedy this, you must force the other modules, (communications line driver and the hard disk drivers) to load into high memory.

Forcing these other modules into high memory is not an easy task, and it involves re-creating your system configuration. If you are running a hard disk system, this can be a time consuming and sometimes confusing

process. For this reason, we are providing some examples of typical system configurations with the communications line driver, and hard disk drivers.

When you boot an original TRSDOS disk, it is configured for two floppy disk drives, no communications line, and no hard disk drivers. In addition, all of the low memory area and all of the high memory area is available for use. At this point, we will begin creating your new system configuration. The first step is to install the XLR8er software (FIXALL, FIXBANK, and RAMDISK). NOTE: A system cannot boot up with RAMDISK in place, however, space must be reserved for it in low memory for it to load at a later time.

The following is a typical JCL file that configures a three drive system with the communications line driver installed, and space reserved in low memory for RAMDISK.

```
set180 (m=1,i=1,r=80) ;speed up a bit
FIXBANK ;install new @bank driver
set *ws FIXALL ;install FIXALL filter
filter *ki *ws
set180 (m=0) ;set speed to maximum
system (drive=2,driver="floppy") ;force drive :2 as a floppy
3
system (drive=3,driver="RAMDISK") ;install dummy RAMDISK
e
1
1
Y
set *cl com ;force com/dvr to high mem
system (drive=3,driver="RAMDISK") ;remove dummy RAMDISK
d
```

Upon completion of this JCL file, a SYSGEN command should be issued from the command line, which will save the system configuration. This enables the machine to return to this configuration automatically the next time it is booted.

Note: RAMDISK was installed in the system and then disabled a short time later for two reasons; (1) SYSGEN cannot be executed with RAMDISK resident in memory, and (2) this forces the COM/DVR into high memory in order to reserve space for RAMDISK in low memory. The very same strategy is used for installing hard disk drivers. The commands to set up the hard disk drivers are simply inserted after, or in place or, the "set *cl com" line in the JCL file.

Installation of XLR8er board

When installing the XLR8er Board in conjunction with MICRO-LABS' GRAPYX SOLUTION Hi-resolution graphics board in the Mod 4/4D/4P, a few changes were required to be made to the normal installation described in the manual.

Type 1 Motherboards (26-1069)

Refer to the full-size 11"x17" supplemental drawing provided for these installation instructions.

1. Read the original Instructions included in Chapter 2 before proceeding. Notice the reference about the need to replace some of the chips on your motherboard.

2. To mount the XLR8er Board with the graphics board in place, it will be necessary to cut an opening in the original RF shield on your Model 4 per the supplemental drawing. A jig saw or coping saw will do an excellent job, but be sure to thoroughly de-burr and clean any metal particles left by this operation.

   Shorten the copper clad backing plate by 1/2" with the same tool used to modify the RF shield; again reference the drawing.

   When this operation is complete, thoroughly de-burr, clean and bolt the XLR8er Board to it using the original instructions for Type 1 motherboards.

3. Carefully remove the graphics board and identify the Z80 CPU chip in the socket on the lower central area of the mother board. Remove the Z80 chip and save in case you need to use it again.

   Bend the 7-1/2" connector cable per the supplemental drawing and plug one end into the Z80 socket. Replace the graphics board in its original position. Plug the other end of connector cable in to the XLR8er and test its location on the RF shield.

   At this time, mark the location of the four bolt holes that will secure the XLR8er Board/backing plate. Drill these four holes using an 1/8" drill bit, de-burr and clean. Assemble the RF shield

to the XLR8er Board's backing plate following the directions in the original instructions for Type 1 motherboards.

4. Re-assemble following the instructions for Type 1 motherboards, being careful in bending the connector cable.

Type 2 Motherboards (26-1069A and 26-1070)

1. Read the original Instructions included in Chapter 2 before proceeding. Notice the reference about the need to replace some of the chips on your motherboard.

2. To mount the XLR8er Board with the graphics board in place, we had to use a little imagination. The simplest solution is to locate the XLR8er Board behind the motherboard (closer to the Disk Drives but inside the inner RF shield). This is accomplished with the use of a 7-1/2" connector cable bent 90 degrees to go over the top of the mother board and behind it. The mounting of the XLR8er Board in that location is accomplished using four pieces of double sided tape. ***Extreme care should be taken to see that none of the soldered connections on the back of the XLR8er Board come in contact with the aluminum backing of the motherboard after installation as this could cause a short and possibly a disaster.***

3. Follow the original instructions type 2 motherboards to complete installation and usage.

Type 3 Motherboards (4P Gate and non-gate array)

1. Read the original Instructions included in Chapter 2 before proceeding. Notice the reference about the need to replace some of the chips on your motherboard.

2. The installation of the XLR8er Board in the Model 4P with the graphics board is the same as the Model 4P given in the original instructions for Type 3 motherboards. Extra care is needed in routing the connector cable and plugging it into the XLR8er Board and should be within the ability of persons able to disassemble the 4P to begin with.

I/O connector pinouts

P1 - EXPANSION PORT

Signal	Pin #		Signal
+5V	1	2	+5v
GND	3	4	GND
RD^	5	6	SYSTEM CLOCK
WR^	7	8	RESET^
E	9	10	LIR^
NMI^	11	12	EXP SEL0^
WAITEX^	13	14	EXP SEL1^
INT0^	15	16	HALT
ST	17	18	N.C.
A0	19	20	A1
TEND0^	21	22	A2
A3	23	24	A4
DREQ0^	25	26	IOE^
N.C.	27	28	RESET OUT
D7	29	30	D6
D5	31	32	D3
D4	33	34	D2
D1	35	36	D0
N.C.	37	38	N.C.
N.C.	39	40	N.C.

```
   P2 - RS232 CONNECTOR

  Signal          Pin #          Signal
----------     ----------     ----------
                 1 | 2    ------ TX0
   RX0 -------   3 | 4    ------ RTS0
  CTS0 -------   5 | 6    ------
   GND -------   7 | 8    ------ DCD0
                 9 | 10
                11 | 12   ------ RX1
   TX1 -------  13 | 14
                15 | 16
                17 | 18   ------ GND
                19 | 20
               ---------
```

```
   P3 - RS232 POWER

  Signal         Pin #
--------      ---------
   +5v -------   1 <---- **
  +12v -------   2
  -12v -------   3
   GND -------   4

** Note: Not used for most applications )
```

RS1 - Port 1 RS232 DB25 pinout

```
 Signal          Pin #
--------        -------
 TX0 ---------> 2
 RX0 <--------- 3
 RTS0 --------> 4 ----┐
 CTS0 <-------- 5     │
 GND <--------> 7     │
 DCD0 <-------- 8     │
 DTR            20 ---┘
```

RS2 - Port 2 RS232 DB25 pinout

```
 Signal          Pin#
--------        ------
 RX1 <--------  2
 TX1 --------> 3
 GND <-------> 7
```

Figure 1.1

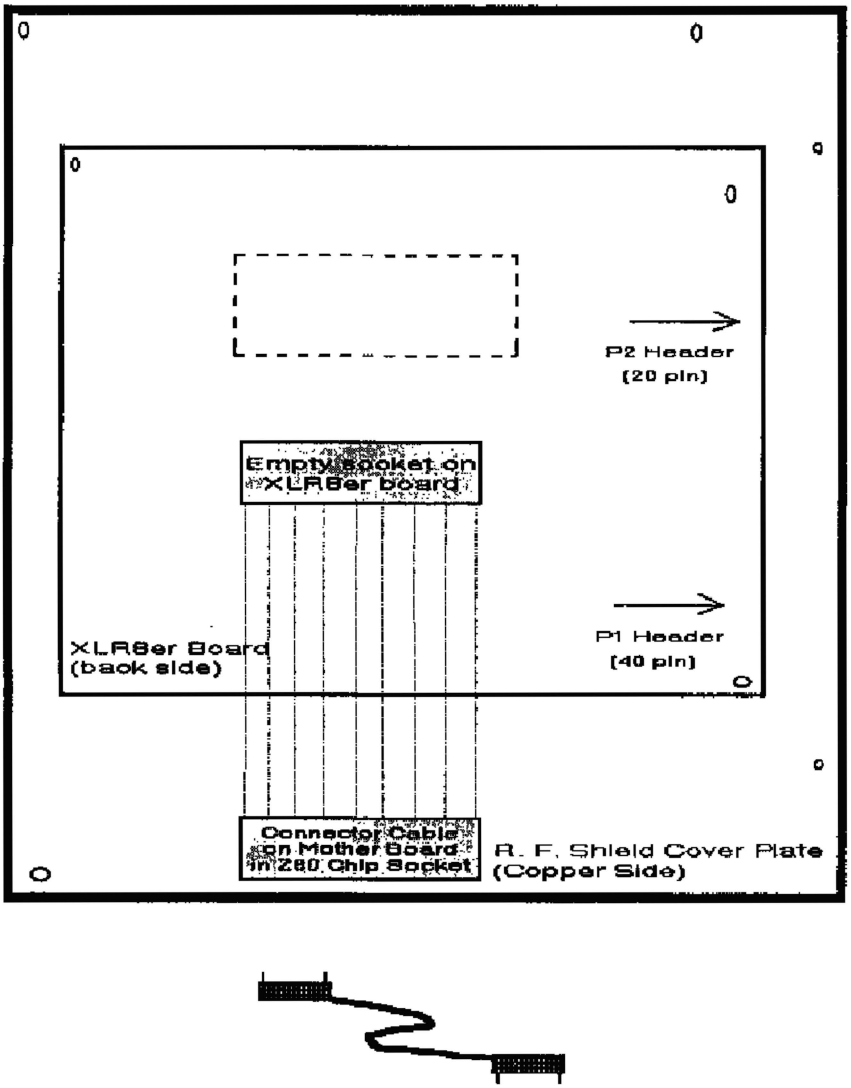

P2 Header
(20 pin)
Empty socket on
XLR8er board
P1 Header
(40 pin)
XLR8er Board
(back side)
Connector Cable
on Mother Board
in Z80 Chip Socket
R. F. Shield Cover Plate
(Copper Side)

Figure 1.2

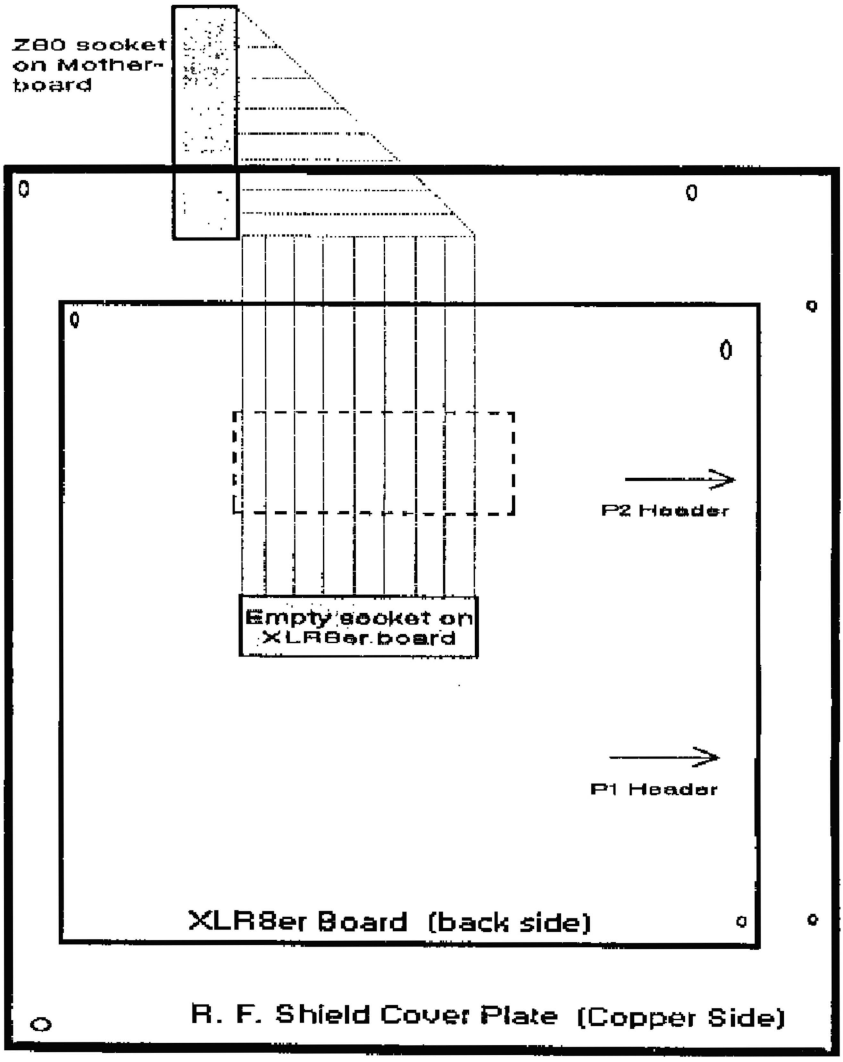

Z80 socket on Mother-board
P2 Header
Empty socket on XLR8er board
P1 Header
XLR8er Board (back side)
R. F. Shield Cover Plate (Copper Side)

Figure 1.3

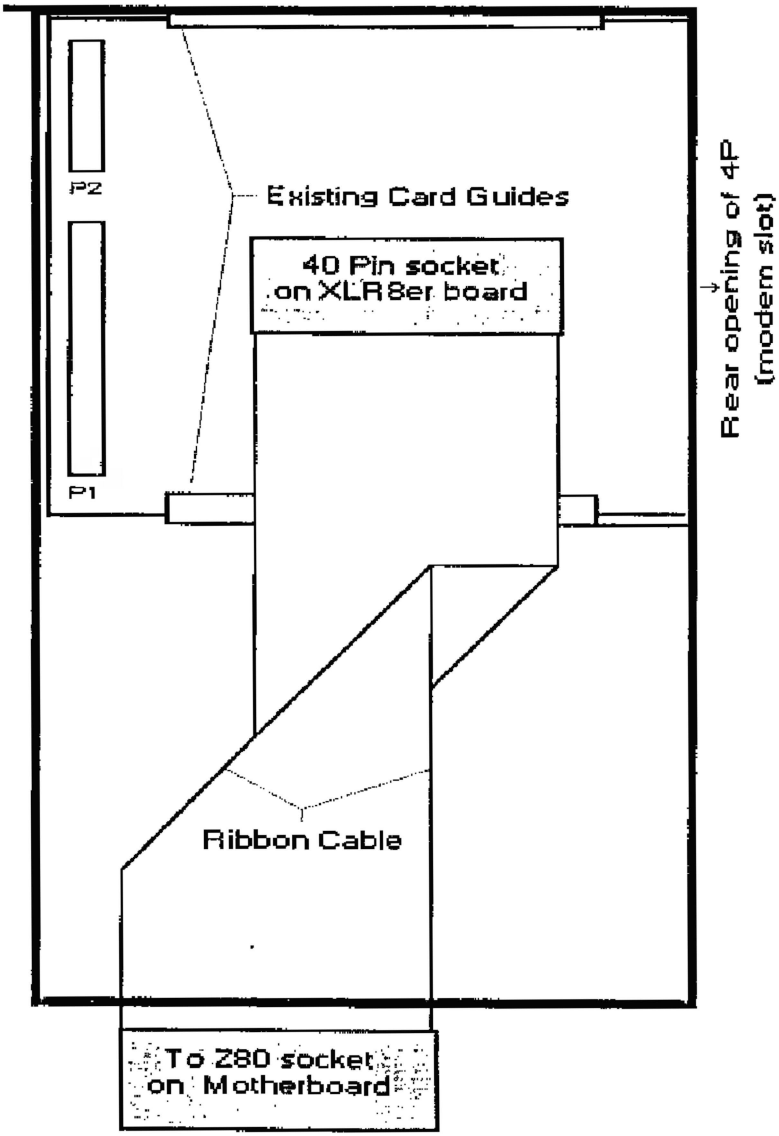
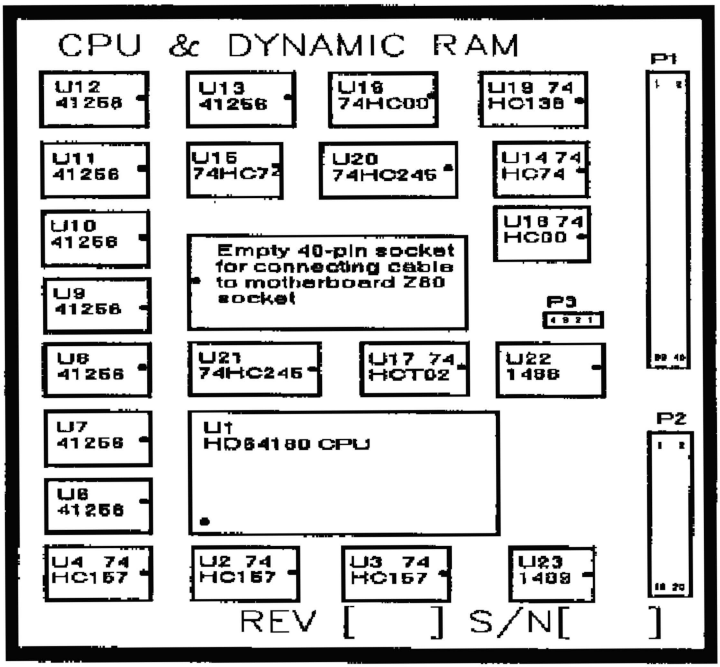

P2
P1
Existing Card Guides
40 Pin socket
on XLR8er board
Rear opening of 4P
(modem slot)
Ribbon Cable
To Z80 socket
on Motherboard

Figure 1.4


CPU & DYNAMIC RAM
U12 41256
U13 41256
U16 74HC00
U19 74 HC138
P1
U11 41256
U15 74HC72
U20 74HC245
U14 74 HC74
U10 41256
U18 74 HC00
Empty 40-pin socket for connecting cable to motherboard Z80 socket
U9 41256
P3
U6 41256
U21 74HC245
U17 74 HCT02
U22 1488
U7 41256
U1 HD64180 CPU
P2
U8 41256
U4 74 HC157
U2 74 HC157
U3 74 HC157
U23 1489
REV [ ] S/N[ ]

MISOSYS, Inc.
Post Office Box 239
Sterling, VA 22170-0239
U.S.A.

703-450-4181

WARRANTY

This product is warranted against defects for one year from date of purchase by MISOSYS, Inc.

Within this period MISOSYS, Inc. will repair or replace it at its option without charge for parts or labor. The warranty does not cover transaction costs; nor does it cover a product subject to misuse or accidental damage.

DISCLAIMER

EXCEPT AS PROVIDED HEREIN, MISOSYS, Inc. MAKES NO WARRANTIES EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR APPLICATION PURPOSES.

Some states do not permit limitations or exclusion of implied warranties therefore the afore said limitations or exclusions may not apply to the purchaser.

This warranty gives you specific legal rights and you may also have other rights which vary form state to state.