CHALLENGER III

Multiple Processor Operations Manual

(Preliminary Version)

Introduction

This is a preliminary manual covering the fundamental operations of the Challenger III triple-processor CPU Board. The Model 510 CPU Board is a new concept in microcomputing allowing the user to utilize any of three processors. The operation of the triple-processor and understanding of it is definitely not trivial, but can be mastered if one has a good basic understanding of microcomputers. This manual covers fundamental routines allowing the user to switch back and forth between processors, and to run Z-80, 6800, and 6502 programs under the benefits of 6502 disk operating systems for program storage. It is a preliminary manual in the sense that we are planning to create a large library of utilities and programs making use of the 510 Board. This will greatly expand its usability, particularly for persons not versed in software or programming. We are very concerned about your interests in software for the 6800 and Z-80 and would like you to fill out the questionnaire at the back of this manual to guide us in developing and marketing software packages that can be of use to you.

TABLE OF CONTENTS

Challenger III CPU

The Model 510, or Challenger III, CPU Board features the 6502A, Z-80, and 6800 microprocessors. This CPU Board allows you to run programs for the 6502, Z-80, 6800, or 8080 on your Challenger III computer system. Operation of Z-80, 6800, and 8080 programs is fairly straighftforward once you become completely familiar with the general operation of your computer system. Therefore it is very important that you become fluent in the operation of the 6502 portion of the system before you explore these other processors.

We will first discuss in general how the Challenger III CPU works before stating specific examples and presenting demonstration programs. The Challenger III is available in two configurations. One is the stripped-down or standard version with a processor-select switch. The other is a deluxe version, which has a software switch, a swappable RAM memory (i.e., RAM scratchpad), and a PIA port.

Let us first consider the operation of the minimal CPU, that is, the CPU with the selector switch. The basic Challenger III, or 510 CPU Board, is equipped with a single pole three position switch. The position of this switch specifies which processor is activated. The other processors are off and in the tristate condition. The 6502 position indicates that when a reset operation takes place by the reset switch, the processor will vector to the floppy disk bootstrap program, and the user can then elect to boot in the disk or go to 65A Monitor program. When the selector switch is in the Z-80 position, the restart occurs from location 0, so that when the user depresses the reset switch and releases it, the processor will start to execute code at location 00 (hex). The user must place his program at 00 (hex), or preferably place a JMP, or C3 (hex) command, to a location and page, in that order, to start up his program. Since users will most likely want to bring in their Z-80 and 6800 programs via the 6502 Disk Operating System, it is generally desirable to place their programs in high memory so that they can be co-resident with the 6502 DOS. For this reason, the user will generally want to put a JMP vector at location 00 for Z-80 programs. All standard 510 CPU Boards with or without software switching option are equipped with 68A Mod 4 PROM Monitor, such that when a user selects a 6800 and resets the computer, the processor will vector to the 68A4 Monitor program. The 68A Monitor provides the minimum requirements for manipulating and writing programs in the 6800, including the LOAD, PRINT, and GO commands.

The 68A documentation includes a discussion of how to convert MIKBUG-compatible programs to run under its system. MIKBUG is a Monitor ROM that has been marketed for a long time by Motorola. The MIKBUG Monitor was used in some of Motorola's early evaluation kits, and has been used in the SWTP 6800 computer.

Several independent software companies such as TSC have written extensive 6800 software packages utilizing the MIKBUG Monitor. The problem with MIKBUG compatibility is that MIKBUG was written for a small computer system. It is a 512-word Monitor residing at E000 to E200. It also makes use of a subroutine stack and scratchpad area at A000 up for 128 bytes. MIKBUG-dependent programs usually start at 0100 (hex) and have branches or subroutines in the MIKBUG Monitor at E000. They may also make use of the small amount of scratchpad memory at A000. The OSI system user has two alternatives when dealing with MIKBUG-dependent programs, that is, programs written to run the 6800 systems with the MIKBUG Monitor. He can follow the instructions provided with the 68A Monitor, which show how to change MIKBUG

subroutine calls from MIKBUG to the 68A. This basically means changing them from an address at E000 to one on page FF, and changing the A0 addresses to 1E addresses. This is a straightforward operation in fully documented programs such as those written by TSC, but is not so easy on undocumented or lightly documented programs. The other alternative is to utilize our MIKBUG simulator routines, which actually reside at E000 (hex). It will be necessary for the user to provide memory residing at these locations to operate these programs. It will minimally require that memory reside from E000 to E200 and A000 to A100, in addition to its normal memory. If he purchases a system with 44K of RAM or more, he will automatically have the memory at A000 (hex). In this case it will be necessary for him to add only a fully or partially populated 4K memory board at E000 up. With these boards and the MIKBUG simulator program in place, it will be possible to run MIKBUG-dependent 6800 programs without modification to the programs.

In place of the selector switch, the 510 Board with the software switch option utilizes a PIA output port to control certain parameters of the board. The processor select switch is documented on page 52 of the Comprehensive Information Package II. Two signal lines, or outputs, from the PIA at F700 are fed into a 74155 at IC-E10. This 74155 does a two-to-four-line decoding on output 0 to 3, which are fed into a 7475 latch. The three possible processsor select codes are: 1-1, 6502; 1-0, 6800; 0-1, Z-80; and 0-0, all three processors tristated. Another PIA line, processor select, is used to actually change processors. This fires a series of one-shots causing all microprocessors to be reset, and then loads the new processor-select code into the 7475, bringing the processors out of reset, thus selecting the one and only one processor which is to be activated. The timing diagram is shown in the upper right corner of page 52 (Comprehensive Information Package II). An external reset switch performs a master reset, which always returns command to the 6502A when actuated. The master reset switch on the front panel always returns control to the 6502A processor when the software switch option is installed on the 510 CPU Board. To change processors, the program must first set up the new processor-select code on the PIA, which will then present this code to the input side of the 7475 latch. After this is done, the software must exercise the processor-select line on the PIA. This fires the one-shots, which reset the processors and load in the new processor-select code into the 7475, thus selecting the new processor and bringing it out of reset. Since both the 6502 and 6800 reset to their Monitor programs, the software change from a 6502 to a 6800 would normally bring a 6800 up in its Monitor program, which would then require additional manual intervention to run the user program. The Z-80, however, vectors to location 0, which can simply contain a JMP to the desired program. With this hardware alone, we can call up any specific Z-80 program from either the 6502 or the 6800.

To have the 6502 or 6800 call up and execute a specific program on the other processors, it is necessary to have additional hardware. This hardware is a swappable, or relocatable, RAM normally located at F200 up. It can be moved to FF00 up. This is accomplished by a single line on the PIA. When this line is actuated, this RAM becomes write-protected and is relocated in the address space of the computer to FF00, disabling the normal Monitor EPROM.

The obvious use for this RAM is to store new restart and interrupt vectors for either the 6502 or the 6800, such that when the

user performs a processor select, this RAM has been swapped up to the restart vector location and contains the restart vector to the program of interest. By invoking the swappable RAM, the user can (completely under program control) switch from a 6502 execution to that of a specific program under the 6800, and vice versa. A secondary purpose for this RAM is to allow the user to change his breakpoint or interrupt vectors at will, or from program to program, permitting practical utilization of interrupts on a 6502 or 6800 system.

Now that we have covered the operation of the Model 510 CPU Board in general terms, let us consider some specific utility programs and examples of switching from one processor to another.

CHALLENGER III UTILITIES DISKETTE

Included with this manual is a diskette containing utilities for handling the 6800 and Z-80 processors on the 510 CPU Board. The diskette contains a modified version of OS-65D in the sense that it does not boot in BASIC, the Assembler or Editor, but simply the Operating System, I/O Distributor and File Handlers. The operation is simply to put the diskette in the drive, reset the computer and type D It will come up with the Disk Operating System prompter A*. The operating system will make use of locations $0000 through $01FF and $2180 through $3180. Other memory locations will be undisturbed by the Disk Operating System.

The DOS has four additional commands. The VU automatically brings in the Challenger III Utilites Package (documented on the following pages). This occupies from $3D70 TO $3FFF. Thus it will run on a 16K Challenger III system. The Challenger III Utilities Package provides a necessary Z-80 Monitor, 6800 and Z-80 Memory Moves, and automatic GO routines between 6800, Z-80 and back for units with a software switch option. The automatic routines which switch between processors are not usable on systems without software switches.

Two commands which can be used only on units with a software switch and after Challenger III utilities have been brought in by the VU command are R6 and RZ. R6 returns to the 6800 operating under the 68A2 Monitor, and RZ switches the machine to the Z-80 operating under the Z-80 Serial Monitor. Remember of course that the standard OS-65D has an RM which returns to the 6502 65A PROM Monitor.

To summarize, we have a (VU,) which automatically boots in the Challenger III utilities package; (RM,) which returns command to the 65A Monitor under 6502; (R6,) which returns command to the 6800 under the 68A2 Monitor (only on units with the software switch, and after the VU command has been executed); and (RZ,) which returns command to the Z-80 system under the Z-80 Monitor (only on units with the software switch option and after the VU command has been executed).

Three additional tracks outside the operating system have useful utilities for the Challenger III system. On track 8 is another copy of the Challenger III Utilities Package, which is offered for user modification if desired. Track 9 contains a MIKBUG simulator program for use on the 6800 portion of the system. It is identical in operation to the very popular Motorala 6800 MIKBUG Monitor Program, except it uses our standard ACIA I/O instead of the PIA I/O. With this Monitor up and running, the Challenger system will be functionally equivalent to a 6800 computer which runs under MIKBUG.

Requirements to run the MIKBUG simulator are a memory board with at least 512 bytes of memory at location E000 (recommended: a partially or fully populated 420 board addressed at E000) and at least 128 bytes at A000. A 48K computer will automatically have this; if not, another 420 Board will have to be strapped here.

Track 10 has a 6800 load, dump, and print data string utilites package. When this is used in conjunction with the 65A2 Monitor, it provides load and dump capabilities in the popular Motorola S-1 checkum format, and a P-Data subroutine similar to the one found in MIKBUG.

The various modules of the Challenger III utilities diskette are covered in detail in the following sections, and at the end of the manual with specific examples of use.

510 Utilities Package

The 510 Utilities Package is a collection of routines which allow convenient operation in switching between 6502, 6800, and Z-80. This utilities resides from $3D70 to $3FFF. It is booted via the VU with the modified OS-65D operating system on your Z-80 utilities disk and is also provided as a separate track for use and modification under other circumstances.

The utilities package has nine modules. The first six modules are Processor Select routines for software switch units only. By simply jumping to the go address specified, the user will switch from the specified processor to the specified processor. For example, entering the routine at 3DAA will switch the machine from the 6800 to the 6502. If a routine is called improperly , that is, if the machine is currently running as a Z-80 and the user calls the routine at 3D70, it will crash. So it is important to be careful in selecting these routines.

The result of these six routines is that the new processor comes up in its Monitor program. In the case of coming up as a Z-80, the Monitor program is located at 3E00 up, as documented separately. In conjunction with the Z-80 Monitor, the routines which select the Z-80 place a JMP 3E00 at location 00, which remains. Following the Z-80 Monitor is the Z-80 Memory Move routine, which can be executed only by the Z-80 processor, of course. With this Move routine, under the Z-80 Monitor, for instance, the user will place the low address and high address of the start of the move ($3F4F and $3F50), the low address and high address of the end of the block to be moved ($3F51 and $3F52), and the starting destination address low, then high ($3F53 and $3F54). He then jumps to the Move program at $3F55, which performs the Move operation, and returns command to the Z-80 Monitor. The user must be careful not to move the program in on top of the Z-80 Monitor, or the Move program, or a crash will occur.

Likewise there is a Memory Move for the 6800 located at $3F79 up. This Memory Move can be operated on by the 68A2 Monitor, for instance. Remember that the 6800, unlike the 6502 and Z-80, uses a syntax of high address and low address. So at location $3F79 put the high address of the beginning of the move, and at $3F7A put the low address. Likewise at $3F7B and $3F7C put the end of the block to be moved, and at $3F7D and $3F7E put the beginning destination address in the order high address-low address. Be very careful to remember the address difference between the 6800, Z-80, and 6502. Confusion here leads easily to computer crashes.

510 UTILITIES

Go Address	Processor (Old)	Processor (New)	
3D70	6502	6800	
3D88	6502	Z-80	
3DAA	6800	6502	
3DBF	6800	Z-80	
3DD4	Z-80	6502	
3DEA	Z-80	6800	
3E00	Z-80	Monitor	
3F55	Z-80	Mover	3F4F/50 Source Address 3F51/52 End Address 3F53/54 Dest. Addr.
3F7F	6800	Mover	3F79/7A Source Address 3F7B/7C End Address 3F7D/7E Dest. Addr.

Executing and Handling Z-80 and 8080 Programs

Call in the the 510 utility via the VU command. On units with the software switch, then type RZ. On units with the mechanical switch, return to the 65A via an RM, then place the JMP 3E00 via L0000 C3 00 3E R. Then hold in the reset switch and rotate the processor switch to Z-80; release the reset switch. You should now see the Z-80 Monitor prompter "?". . The user can then enter programs and run them via the Monitor. If large programs available on paper tape or cassette are to be entered, it would be desirable to write a loader program which could then be used to load other programs. Another approach is to write such loaders for the 6502, thus making use of the 6502 Assembler.

Storing Z-80 Programs on Disk

The 6502 Disk Operating System uses all of Page 0 and 1 (of necessity) and $2200 to $3200. 6502 modules such as BASIC or the Assembler use additional memory. To conserve memory a DOS-only operating system is present on the Challenger III utilities diskette. Programs which do not use $0-$200 and $2200 to $3200 can be directly stored and retrieved from this disk by using the C (=CALL) and S (=SAVE) commands. This is accomplished on units with the mechanical switch by holding in reset while rotating back to the 6502. Then reboot the 6502 DOS (just to be safe) and save the program of interest on disk. If the program of interest uses memory locations which are used by the 6502 DOS, use the move utility at the end of the Z-80 Monitor to move the program to high memory, for example, at $4000 up, then save it via the 6502 DOS. Likewise the move routine can be used to place a program back in position after being brought in via the 6502 DOS.

The procedure for getting back to the 6502 from the Z-80 on units with the software switch is either to enter the routine at $3DD4 or preferably just reset the machine.

Calling Z-80 Programs from Disk

Under the 6502 DOS call in the utilities package and your program. On software switch units type RZ. On mechanical switch units exit the DOS via an RM. Place a JMP to the Z-80 Monitor at location 00 via L0000 C3 00 3E R, then hold the reset switch in, switch to Z-80 and release. In either case, once you are in the Z-80 Monitor you can go to the program via the Monitor's GO command. Entering the Z-80 Monitor can be avoided by setting the JMP at location 00 directly to the start of the program. However it is generally desirable to have the Z-80 restart set for the Z-80 Monitor so that the Z-80 can be restarted in case of a crash.

Summary of Z-80 Operation

To enter programs:

1. Bring in 6502 DOS
2. Call in Z-80 Monitor
3. Call in Challenger III utilities
4. Switch to Z-80 (on manual switch units first set up C3003E)
5. Move program in place if necessary; if new program, key in via Monitor

To save programs:

1. Move program to high memory if necessary (above $4000)
2. Switch back to 6502
3. Reboot 6502 DOS and save

Z-80 Monitor

The Z-80 Monitor is a short machine code program that is provided to reside at 3E00 up for approximately one page. This program offers elementary memory examine, modify and program execute commands on a serial system using a standard ACIA port as the I/O device.

To operate this program one must first place it in memory typically by use of the 6502 Disk Operating System. The user must place a JMP 3E00 at location 00 in memory (C3 00 3E). Then the user must either manually reset the processor as a Z-80 or go through the software switch and convert the system to a Z-80. Keep in mind that the 6502 DOS and BASIC and other large programs will wipe out the JMP at location 00 if executed, so that the JMP placed at location 00 up should be done after exiting the DOS or BASIC, just before the user actually turns on the Z-80. The Z-80 Monitor comes up with a ? prompter. The user types in a four-digit hexadecimal address, for example, AAAA. The monitor then displays the contents of that address as two hex digits. The user can then optionally modify those locations by typing a two-digit hex number. As soon as the two-digit hex number is typed in, the monitor will automatically display the next address and its contents. The user can examine memory by simply typing <return> immediately after the contents of the memory location are displayed. By successively typing <return> he will display consecutive memory locations. When the user types a ^, the Monitor will show the memory location previous to the one shown.

The user can exit the display mode at a particular address by typing an R. This returns the Monitor to the command mode, where it expects to see an address again. Thus the user can look at a few locations, type an R, type in a new address, and examine locations there. To execute a program the user must type in the starting address of the program, then an RG, which will cause program execution at the specified address.

The Z-80 Monitor contains an INPUT subroutine with echo located at 3E91. This subroutine puts ASCII character in the Accumulator upon exit The Accumulator is modifed. Another useful routine in the Z-80 Monitor is the Character Output Routine. This routine is located at 3E9D and returns the ASCII value, outputting the ASCII equivalent of the Accumulator. Other useful routines are as follows :

3F10 LEGAL - prints a ? if (A)<>ASCII for 0-->9 or A-->F
also sets carry
if (A)=ASCII for 0-->9 or A-->F the sub returns the hex value with the carry=0

3F2C PROMPT- prints a ? and sets the carry flag (A lost)

3F44 CR-LF - prints carriage return-linefeed (A lost)

3F38 DIGIT - converts hex 0-->9 or A-->F to ASCII code (uses A only)

3EAB DISP DATA- prints </nn>, where nn= contents of
memory location pointed to by $FE(low) $FF(high)

Stack Pointer initialized at $100.

```
;SERIAL Z-80/8080 MONITOR

3E00 318001   START   LXI SP, $180
3E03 210000           LXI H,$0
3E06 22FE00           SHLD $FE
3E09 2100FC           LXI H,$FC00
3E0C 3E03             MVI A,3
3E0E 77               MOV M,A
3E0F 3EB1             MVI A,$B1
3E11 77               MOV M,A
3E12 CD443F           CALL CR-LF
3E15 CD2C3F           CALL PROMPT

3E18 CD443F   AD      CALL CR-LF
3E1B 21FF00           LXI H,$FF
3E1E 0602             MVI B,2
3E20 0E02     REP2    MVI C,2
3E22 CD913E   REP1    CALL INCH
3E25 FE47             CPI G
3E27 CA8D3E           JZ GO
3E2A FE52             CPI R
3E2C CA183E           JZ AD
3E2F CD113F           CALL LEGAL
3E32 DA183E           JC AD
3E35 0D               DCR C
3E36 CA443E           JZ SKIP1
3E39 17               RAL
3E3A 17               RAL
3E3B 17               RAL
3E3C 17               RAL
3E3D 77               MOV M,A
3E3E 3E3F             MVI A,0
3E40 B9               CMP C
3E41 C2223E           JNZ REP1
3E44 B6       SKIP1   ORA M
3E45 77               MOV M,A
3E46 2B               DCR B
3E48 C2243E           JNZ REP2
3E48 CDAB3E           CALL DISP DA

3E4E 0E02     DA      MVI C,2
3E50 CD913E   REP3    CALL INCH
3E53 FE52             CPI R
3E55 CA183E           JZ AD
3E58 FE5E             CPI $5E
3E5A CA833E           JZ BACK
3E5D FE0D             CPI $D
3E5F CA793E           JZ NEXT
3E62 CD113F           CALL LEGAL
3E65 DA183E           JC AD
3E68 0D               DCR C
3E69 CA773E           JZ SKIP2
3E6C 17               RAL
3E6D 17               RAL
3E6E 17               RAL
3E6F 17               RAL
3E70 47               MOV B,A
3E71 3E00             MVI A,0
3E73 B9               CMP C
3E74 C2503E           JNZ REP3
```

```
3E77 B0       SKIP2   ORA B
3E78 77               MOV M,A
3E79 23       NEXT    INX H
3E7A CDCC3E           CALL DISP AD
3E7D CDAB3E           CALL DISP DA
3E80 C34E3E           JMP DA
3E83 2B       BACK    DCX H
3E84 CDCC3E           CALL DISP AD
3E87 CDAB3E           CALL DISP DA
3E8A C34E3E           JMP DA

3E8D 2AFE00   GO      LHLD $FE
3E90 E9               PCHL

3E91 3A00FC   INCH    LDA $FC00
3E94 1F               RAR
3E95 D2913E           JNC INCH
3E98 3A01FC           LDA $FC01
3E9B E67F             ANI $7F
3E9D F5       OUTCH   PUSH PSW
3E9E 3A00FC   WAIT    LDA $FC00
3EA1 1F               RAR
3EA2 1F               RAR
3EA3 D29E3E           JNC WAIT
3EA6 F1               POP PSW
3EA7 3201FC           STA $FC01
3EAA C9               RET

3EAB 3E2F     DISP DA MVI A,'/
3EAD CD9D3E           CALL OUTCH
3EB0 2AFE00           LHLD $FE
3EB3 7E               MOV A,M
3EB4 F5               PUSH PSW
3EB5 1F               RAR
3EB6 1F               RAR
3EB7 1F               RAR
3EB8 1F               RAR
3EB9 CD383F           CALL DIGIT
3EBC CD9D3E           CALL OUTCH
3EBF F1               POP PSW
3EC0 CD383F           CALL DIGIT
3EC3 CD4D3E           CALL OUTCH
3EC6 3E20             MVI A,$20
3EC8 CD9D3E           CALL OUTCH
3ECB C9               RET

3ECC CD443F   DISP AD CALL CR-LF
3ECF 22FE00           SHLD $FE
3ED2 0602             MVI B,2
3ED4 0E02             MVI C,2
3ED6 7C       REP4    MOV A,H
3ED7 05               DCR B
3ED8 CADF3E           JZ SKIP3
3EDB 1F               RAR
3EDC 1F               RAR
3EDD 1F               RAR
3EDE 1F               RAR
3EDF CD383F   SKIP3   CALL DIGIT
3EE2 CD4D3E           CALL OUTCH
3EE5 0D               DCR C
```

```
3EE6 C2D63E           JNZ REP4
3EE9 0602             MVI B,2
3EEB 0E02             MVI C,2
3EED 7D       REP5    MOV A,L
3EEE 05               DCR B
3EEF CA063F           JZ SKIP4
3EF0 C3023F           JMP EXTRA
3EF3 00               NOP
3EF4 00               NOP
3EF5 00               NOP
3EF6 00               NOP
3EF7 00               NOP
3EF8 00               NOP
3EF9 00               NOP
3EFA 00               NOP
3EFB 00               NOP
3EFC 00               NOP
3EFD 00               NOP
3EFE 00               NOP
3EFF 00               NOP
3F01 00               NOP
3F02 1F       EXTRA   RAR
3F03 1F               RAR
3F04 1F               RAR
3F05 1F               RAR
3F06 CD303F   SKIP4   CALL DIGIT
3F09 CD9D3E           CALL OUTCH
3F0C 0D               DCR C
3F0D C2ED3E           JNZ REP5
3F10 C9               RET

3F11 FE30     LEGAL   CPI $30
3F13 DA2C3F           JC FAIL
3F16 FE3A             CPI $3A
3F18 DA273F           JC OK
3F1B FE41             CPI $41
3F1D DA2C3F           JC FAIL
3F20 FE47             CPI $47
3F22 D22C3F           JNC FAIL
3F25 D607             SUI 7
3F27 E60F     OK      ANI $F
3E29 37               STC
3E2A 3F               CMC
3F2B C9               RET
3F2C 3E20     FAIL    MVI A,$20   (PROMPT)
3F2E CD9D3E           CALL OUTCH
3F31 3E3F             MVI A,'?
3F33 CD9D3E           CALL OUTCH
3F36 37               STC
3F37 C9               RET

3F38 E60F     DIGIT   ANI $F
3F3A F630             ORI $30
3F3C FE3A             CPI $3A
3F3E DA433F           JC EXIT
3F41 C607             ADI 7
3F43 C9       EXIT    RET

3F44 BE0D     CR-LF   MVI A,$D
3F46 CD9D3E           CALL OUTCH
```

```
3F49 3E0A              MVI A,$A
3F4B CD9D3E            CALL OUTCH
3F4E C9                RET
```

Z-80 Mover

```
      Source address low @ 3F4F, high @ 3F50
      End address low @ 3F51, high @ 3F52
      Destination address low @ 3F53, high @ 3F54

3F55 2A513F       LHLD     ;END ADDRESS-->(HL)
3F58 EB           XCGH     ;END ADDRESS-->(DE)
3F59 2A4F3F LOOP  LHLD     ;SOURCE ADDRESS-->(HL)
3F5C 7E           MOV A,M  ;(M)-->(A)
3F5D 2A533F       LHLD     ;DESTINATION ADDR. -->(HL)
3F60 77           MOV M,A  ;(A)-->(M)
3F61 23           JNX H    ;(HL+1)-->(HL) (=INCREMENTS...
3F62 22533F       SHLD     ;DESTINATION ADDRESS & STORES IT)
3F65 2A4F3F      LHLD     ;SOURCE ADDRESS-->(HL)
3F68 7A           MOV A,D  ;END ADDRESS HIGH-->(A)
3F69 BC           CMP H    ;ARE WE ON UPPER LIMIT PAGE?...
3F6A C2723F       JNZ SKIP ;...IF NOT, SKIP TO INCR. SOURCE ADDR.

3F6D 7B           MOV A,E  ;END ADDR. LOW-->(A)
3F6E BD           CMP L    ;ARE WE DONE?
3F6F CA123E       JZ EXIT  ;YES-RETURN TO MONITOR

3F72 23      SKIP INX H    ;(INCREMENTS SOURCE ADDRESS...
3F73 224F3F       SHLD     ;...& STORES IT)
3F76 C3593F       JMP LOOP ;NOT FINISHED LOOP BACK
```

Executing and Handling 6800 Programs

The 6800 portion of the 510 Board has its own 256-byte PROM Monitor located at FF00. To turn on the 6800 and enter the 68A2 Monitor on mechanical switch units, simply hold the reset switch in, rotate the switch to the 6800 position and release the reset switch. The terminal should output a carriage return-linefeed and be in the 6800 Monitor. On units with the software switch call in the utilites package via the DOS and type RG, which will bring up the 6800.

Once in the 68A2 Monitor, programs can be entered manually if desired. If long programs are available on cassette or paper tape, it would be desirable to create a loader program to bring these in. A Motorola "S" format checksum tape loader and dumper is included in the 6800 utilities package. This format is almost universally used for all paper tape and most audio cassette programs for the 6800.

Storing 6800 Programs on Disk

The 6502 Disk Operating System uses all of Page 0 and 1 (of necessity) and $2200 to $3200. 6502 modules such as BASIC or the Assembler use additional memory. To conserve memory a DOS-only operating system is present on the Challenger III utilities diskette. Programs which do not use $0-$200 and $2200 to $3200 can be directly stored and retrieved from this disk by using the C (=CALL) and S (=SAVE) commands. This is accomplished on units with the mechanical switch by holding in reset while rotating back to the 6502. Then reboot the 6502 DOS (just to be safe) and save the program of interest on disk. If the program of interest uses memory locations which are used by the 6502 DOS, use the 6800 move utility to move the program to high memory, for example, at $4000 up, then save it via the 6502 DOS. Likewise the move routine can be used to place a program back in position after being brought in via the 6502 DOS.

The procedure for getting back to the 6502 from the 6800 on units with the software switch is either to enter the routine at $3DAA or preferably just reset the machine.

Calling 6800 Programs from Disk

Under the 6502 DOS call in the utilities package and your program. On software switch units type RG. On mechanical switch units then hold the reset switch in, switch to 6800 and release. In either case, once you are in the 68A Monitor you can go to the program via the Monitor's GO command. Entering the 68A Monitor can be avoided by setting up the swappable RAM (as covered in a later section) directly to the start of the program. However it is generally desirable to have the 6800 restart set for the 6800 Monitor so that the 6800 can be restarted in case of a crash.

Summary of 6800 Operation

To enter programs:

1. Enter 6502 DOS.
2. Call in Challenger III Utilities.
3. Call in 6800 Program from Disk.
4. Switch to 6800.
5. Move Program in place if necessary; if new program, key in via 68A2 Monitor.

To save programs:

1. Move Program to high memory if necessary (above $4000).
2. Switch back to the 6502.
3. Reboot 6502 DOS to save.

6800 Utilities

Track 10 of the Challenger III diskette contains a 6800 load-dump and p-data routine that is documented in the 68A manual. To utilize this program, simply type C1D00=10,1, utilizing the instructions in the 68A writeup. The 6800 Memory Mover is part of the Challenger III utilities. Its operation is documented there. Finally the 6800 MIKBUG Simulator is provided on track 9, and its use is discussed in the following section.

6800 Move File (relocatable)

```
1E80    3F7F   FE    LDX
1E81    3F80   1E
1E82    3F81   02
1E83    3F82   A6    LDA,X
1E84    3F83   00
1E85    3F84   FE    LDX
1E86    3F85   1E
1E87    3F86   06
1E88    3F87   A7    STA,X
1E89    3F88   00
1E8A    3F89   08    INX
1E8B    3F8A   FF    STX
1E8C    3F8B   1E
1E8D    3F8C   06
1E8E    3F8D   FE    LDX
1E8F    3F8E   1E
1E90    3F8F   02
1E91    3F90   BC    CPX
1E92    3F91   1E
1E93    3F92   04
1E94    3F93   27    BEQ
1E95    3F94   06
1E96    3F95   08    INX
1E97    3F96   FF    STX
1E98    3F97   1E
1E99    3F98   02
1E9A    3F99   20    BRA
1E9B    3F9A   E7
1E9C    3F9B   7E    JMP
1E9D    3F9C   FF    (mod for 510 system)
1E9E    3F9D   B2
1E9F    ---
```

	Source	High Limit	Destination
High	3F79	3F7B	3F7D
Low	3F7A	3F7C	3F7E

6800 MIKBUG Simulator

The MIKBUG Simulator is on track 9 of the Challenger III utilities diskette. It is a 512-byte program residing from $E000 to E1FF. The MIKBUG simulator is functionally equivalent to the popular Motorola MIKBUG Monitor, with the exception that the Motoral Monitor uses a PIA interface in conjunction with some discrete logic to perform a serial I/O, while in the MIKBUG Simulator the standard 6850 ACIA port at FC00 is utilized. Both MIKBUG and the MIKBUG Simulator reside at E000 and require a subroutine stack of at least 128 bytes at A000 up. The MIKBUG Simulator program is useful because most hobbyist-level software written for the 6800 has been written on MIKBUG-based 6800 machines, and generally this software makes heavy use of subroutines located within the MIKBUG Monitor, and also makes use of the 128-byte scratchpad RAM normally located at $A000. Therefore having the capability of configuring your Challenger III system to be virtually identical to a MIKBUG-based 6800 system may be of value.

To accomplish complete compatibility and run MIKBUG, it is necessary to have a total of 512 bytes of RAM memory at $E000 up. The recommended way of achieving this is to partially populate or fully populate a Model 420C Memory Board and address it for $E000 up and place it in the system. It will also be necessary to have at least 128 bytes of RAM memory at $A000. If the machine is a 48K computer, it will automatically have memory here. If it is less than 44K, it will be necessary to add a fully or partially populated 420C Memory Board at this location also. So if memory is at $A000 for at least 128 bytes, and at $E000 for at least 512 bytes, the user can call in the MIKBUG Simulator on track 9 by typing CE000=09,1. Then either manually or under software control go to the 68A2 Monitor. The MIKBUG Simulator Monitor is entered at $E0D0. So from 68A2 type L1F2E E0D0 RG. The machine will then output a *, which is the MIKBUG prompter.

The MIKBUG Simulator is completely identical in operation to the standard MIKBUG. The operation of MIKBUG is well documented in Motorola Engineering Note 100. The MIKBUG Monitor has five basic commands: L (LOAD), M (MEMORY), P (PUNCH), G (GO), and R (REGISTER DUMP). Typing an L with the prompter on the screen causes the MIKBUG Simulator to go into the checksum loader routine. This routine will accept the popular Motorola S-1 checksum format via the ACIA port. However, if one is not using a Teletype as an input device, it will be necessary to modify this routine for high-speed paper tape or audio cassette. It would be advisable to utilize the 6800 load-save routine on track 10 instead, or S-1 format checksum loads. The load is terminated by a CHECKSUM error, which will cause a ? and return to the command mode or an S9, which terminates the record.

The most useful command of the MIKBUG is the Memory Examine and Modify command. To utilize this type M followed by a four-digit address in hex. The MIKBUG then reprints the address followed by the contents of that location. To simply examine locations type <return>s. To modify a location type <space-bar> followed by two characters to be loaded into that location. To exit the Memory Load and Examine routine, type <space-bar> <return>.

MIKBUG also contains a Punch command which lists out a block of memory in the Motorola S-1 format. Before the Punch command is executed, it is necessary to use the Memory command to load the starting and ending addresses of the Punch. The starting address must be loaded into address A002 and A003 (high-low), and the ending

address into address A004 and A005 (high-low). Once these four locations have been loaded, type P and the MIKBUG Monitor will printout a checksum record of the memory.

Another command of the MIKBUG Monitor is the R command (REGISTER DISPLAY). Type an R and the Monitor puts out the current contents of the Condition Code Register, the Accumulators (A and B), Index Register (high and low), and the Go stack or interrupt stack. This Go stack is just like the stack in the OSI 68A2 Monitor, that is, these register values will be placed into the microprocessor upon execution of a Go command and will be reloaded into the monitor upon execution of a software interrupt, so that it will perform both go and breakpoint functions. The register values are stored in memory locations $A043 (Condition Code Register), $A044 (Accumulator A), $A045 (Accumulator B), $A046/47 (Index Register), $A048/49 (Program Counter), and $A04A/4B (Stack Pointer), so that the user is free to modify the status of all registers, including most importantly the program counter, before starting a program after breakpoint, or initially, by way of the Go command.

The final instruction is a Go command. By typing a G, the contents of A043 to A04B are loaded into the corresponding microprocessor registers, and program execution is then resumed. It is always necessary for the user to specify at least Program Counter at $A048/49 and a Stack Pointer at $A04A/4B before typing a G to execute a program.

We do not recommend that the user use MIKBUG under normal circumstances as a Monitor. We believe that the 65A2 Monitor is just as convenient as and certainly more accessible to the user. Also, the MIKBUG Monitor's breakpoint routine is useless unless its breakpoint vector has been loaded into the swappable RAM memory, and this has been swapped up to high memory. Memory should also be swapped to set the MIKBUG Simulator entry point E0D0, when it is used with programs making use of the MIKBUG reset and interrupt vectors. MIKBUG should be used in situations where MIKBUG must be resident in memory or the program in order to operate, and the user does not wish to modify the program to relieve its dependence on MIKBUG, and in situations where the swapping RAM must be swapped up to high memory, because under those circumstances the OSI 68A2 Monitor will not be usable because it is deactivated when the swappable RAM is in high memory (from FF00 up).

A Programmer's Guide to the Software Switch

As stated earlier, the software processor switch is controlled by the 6820 PIA at F700 up. On Challenger III CPUs equipped with the software switch, this PIA controls the selection of microprocessors and also optionally allows the user to exchange a 128-byte RAM normally located at F200 for the Monitor PROM located at FF00 up. Thus the user can pre-load this scratchpad RAM at F200 with new restart and/or interrupt vectors, then swap it up to replace the Monitor PROM. Hence it is called "swappable RAM."

Operation of the software switch to select processors and swap the RAM is possible manually by simply keying in the appropriate codes to the PIA control registers. The following table shows how this is accomplished. First the user enters $04 into address F701. He then enters the appropriate processor select code into location F700. He loads a 00 into location F701 and an FF into F700. The action of loading the FF into F700 will trigger the Processor Selector switch into operation, which will then select the new processor and/or swap up the RAM at F200. The Processor Select codes are shown directly below, with and without swapping the RAM. This table means that if we put a 9F in for the PS code, this routine will force the machine to switch over to the 6800, no matter what the current state of the machine in. Likewise, if we load an 8F, the machine will revert to a 6800, and we use locations in the 128-word scratchpad memory for restart and interrupt vectors, regardless of the current state of the machine. So by this simple procedure, the user can manually switch between processors and swap up the RAM memory. The execution of this routine while utilizing the Processor Select code for the same procedure should have no effect other than resetting the processor. Using this routine with the Processor Select code for the processor you are currently using and the RAM swap will cause a reset with the RAM being switched up to high memory. This is caused when bit 4 of the Processor Select code is low. In that case the RAM is moved up to FF00 up, and the PROM Monitors for the 6800 at FF00 up and the 6502 disk bootstrap PROM at FF00 up are disabled. The RAM is also write protected when it is in this condition, so it is not possible for the user to change locations within the RAM when the RAM is at FF00 up. It is still possible to use the 65A Monitor after the RAM has been swapped, but it is not possible to use the 68A Monitor after the RAM is swapped, because this swap disables the Monitor.

When the RAM memory has been swapped up to FF00 up by having bit 4 of the processor select code high, the vectors contained at locations F2FA to F2FF are utilized. This means that in order to put in a new restart vector for the 6502, the user first writes the restart vector into location F2FC, and then swaps this RAM up to high memory. Remember that all 6502 vectors are in the form "low-order address, page," while all 6800 vectors are in the form "page, address."

The following page lists machine executable subroutines for switching from one processor to any other processor with or without RAM swap. These are three separate routines, one each for the 6502, 6800, and Z-80. In each case, the code is completely relocatable. For instance, for the 6800 program to return control to the 6502, the user simply places the 6800 processor select subroutine at the end of his program as listed, and substitutes a DF for the processor select code. When this is executed by the 6800, it will return command to the 6502. To perform the same function, but go to a specific address

other than the Monitor of the 6502, the 6800 programmer would first have to load the desired restart address into location F2FC low and F2FD high of the swappable RAM, with this RAM FC00, and then execute the 6800 Processor Select subroutine with a CF in the Processor Select code instead of a DF.

Software Switch

Manually through Monitor

04--->F701
PS--->F700
00--->F701
FF--->F700

PS	NO RAM SWAP	WITH RAM SWAP
Z-80	5F	4F
6800	9F	8F
6502	DF	CF

Vectors	6502	6800
	65A	68A2

PROM		
Restart	FE35	FFE7
NMI	0130	1FE0
IRQ	01C0	1FD0
Software	----	FFA8

RAM		
Restart	(F2FC)	(F2FE)
NMI	(F2FA)	(F2FC)
IRQ	(F2FE)	(F2F8)
Software	----	(F2FA)

PROCESSOR SELECT SUBROUTINES

For 6502--

```
  A2 00       ENTRY  LDX 0
  8E 01 F7           STX$F701    SELECT DDRA
  CA                 DEX
  8E 00 F7           STX$F700    PORT A=OUTPUT
  A0 04              LDY 4
  8C 01 F7           STY$F701    SELECT DATA REG
  8E 00 F7           STX$F700    SET ALL DATA HIGH
 *A0 XX              LDY PS      PROCESSOR CODE
  8C 00 F7           STY$F700
  D0 FE       TRAP   BNE TRAP    WAIT FOR NEW PROCESSOR
```

For 6800--

```
  CE F7 00    ENTRY  LDX $F700   PIA BASE ADDRESS
  86 00              LDA A 0
  A7 01              STA A 1,X   SELECT DDRA
  4A                 DEC A
  A7 00              STA A 0,X   PORT A=OUTPUT
  C6 04              LDA B  4
  E7 01              STA B 1,X   SELECT DATA REG
  A7 00              STA A 0,X   SET ALL DATA HIGH
 *C6 XX              LDA B PS    PROCESSOR CODE
  E7 00              STA B 0,X
  3E                 WAI         WAIT FOR NEW PROCESSOR
```

For Z-80--

```
  21 00 F7           LXI H $F700
  EB                 XCHG
  21 01 F7           LXI H $F701
  3E 00              MVI A   0
  77                 MOV M,A     SELECT DDRA
  3D                 DCR A
  12                 STAX D      PORT A=OUTPUT
  47                 MOV B,A
  3E 04              MVI A 4
  77                 MOV M,A     SELECT DATA REG
  78                 MOV A,B
  12                 STAX D      SET ALL DATA HIGH
 *3E XX              MVI A PS    PROCESSOR CODE
  12                 STAX D
  76                 HLT         WAIT FOR NEW PROCESSOR
```

PS CODE--

PROCESSOR	NO RAM SWAP	WITH RAM SWAP
Z-80	5F	4F
6800	9F	8F
6502	DF	CF

*Shown in Immediate Mode, but may use any non-indexed addressing mode without disrupting program flow.

CHALLENGER III MULTIPLE PROCESSOR OPERATION EXAMPLES

```
D/M?D
A*VU
A*R
 ?
0000/C3
0001/00
0002/3E R

3DD4/21 R
D/M?D
A*R
P 0000
C3 00 3E 4C 0E 0A 0E 0F
04 11 00 18 20 30 20 20


L 1F2E 3DAA
(VMD
A*
```

Example 1. Challenger III utilities diskette was put in the disk drive, the machine was reset, the user typed a D. The operating system was booted in, returning a prompter A*. The user typed a VU, bringing in the Challenger III utilities. He then typed an RZ. The Z does not print, because the machine immediately switched to the Z-80, showing the Z-80 prompter "?". The user then typed in a four-character address (location 0), and examined that location. He typed <return> again, which outputed the following location. He typed <return> again, which outputed location 2, typed R to exit the memory examine routine. He then typed in address 3DD4, then RG. The G did not print, because the user went to address 3DD4, which immediately switched the machine back to the 6502. We then see the 6502 Monitor prompter D/M? The Disk Operating System was booted in again. This time the user typed R6, which immediately switched the machine over to the 6800, the prompter being cr-lf. The user then printed out memory location from 0 on, he then loaded 1F2E (the go locations), with the address of the routine to bring him back to the 6502 (i.e., 3DAA). He then typed an RG, which does not print. The G brought the unit back to the 6502 DOS. This routine reinitializes the ACIA, so that the first character output may not be correct. Hence, instead of getting D/M? we got VM. He then typed D, which rebooted the operating system, and brought him back to the beginning. So in this example the user went from the 6502 to the Z-80, back to the 6502, then to the 6800, and back to the 6502.

Example 2. The user is in the DOS and calls in track 10, which is the load and save of the 6800. He then typed an R6, which took him to the 6800. The under the 68A2 Monitor, he loaded the start and end address of an S-1 format dump. He then loaded the GO vectors at 1F2E for the S-1 format dump program, and set the stack at 1F80. He typed RG (R does not print). The program at 1D70 then outputed an S-1 format checksum dump and exited back to the 68A2 Monitor

```
  C1D00=10,1
A*R
L 1E02 0000 0020
L 1F2E 1D70 1F80
G
S1130000C3003E4C0E0A0E0F041100182030202 0AD
S113001001300100202020203432342020202020F0
S104002000DB
```

Example 3. The user is in the 6502 DOS and calls into the MIKBUG Simulator on track 9. He then typed R6, which converted the machine to the 6800 under the 68A2 Monitor. He then loaded vector E0D0 into the 68A2 GO address at 1F2E and typed RG, which put him in the MIKBUG simulator and outputed the * prompter. He then executed a memory command in the MIKBUG Simulator, followed by address 0000. The monito reported the contents of that location, and the user typed any key but a <space-bar> to cause the monitor to go through consecutive memory locations. Finally after address 3, the user typed a <space-bar> <return> and then used the REGISTER command, which outputed the current contents of the GO stack. The user then used a MEMORY command to modify location A045, A046, and A047, thus changing command then shows these changes. Finally the user changed the Program Counter contents to point back to the 68A2 monitor, and types a G (which does not print), returning the 6800 to the 68A2 monitor. He then prints out location 0, loads it with the sequence 01 through 0A, then examines location 200 and observes the data there. He then loads location 3F79 up with the start of move, end of move, and start of destination for the 6800 move portion of the Challenger III utilities. He then sets up the GO vector and subroutines stack at 1F2E and types a G. The move is nearly instantaneous and when it returns command to the 68A2 monitor, the user examines location 200 and verifies that the move did in fact occur.

```
A*CE000=09,1
A*R
L 1F2E E0D0

*M 0000
*0000 C3 .
*0001 00 .
*0002 3E .
*0003 4C
*R 24 24 24 2424 2424 A042
*M A043
*A045 24  10
*A046 24  20
*A047 24  30
*A048 24
*R 24 24 10 2030 2424 A042
*M A048
*A048 2C  FF
*A049 2C  A8
*A04A 2C
*A04B 2C
*
P 0000
80 80 80 80 80 80 80 80

L 0000 01 02 03 04 05 06 07 08
P 0200
80 80 80 80 80 80 80 80

L 3F79 0000 0010 0200
L 1F2E 3F7F 1F80
G
P 0200
01 02 03 04 05 06 07 08
```

```
  D/M?D
A*VU

A*R
 ?
0100/90 10
0101/90 20
0102/90 30
0103/62 40
0104/FF 50
0105/F0 60
0106/FF 70
0107/90 80
0108/90 90

0109/91    ?
0300/90
0301/90
0302/90
0303/90

0304/90
0305/90
0306/90    ?
3F4F/FF 00
3F50/3D 01
3F51/FF 10
3F52/3D 01
3F53/00 00
3F54/5E 03
3F55/2A
3F56/51    ?
3F55/2A R

G
 ?
0300/10
0301/20

0302/30
0303/40
0304/50
0305/60
0306/70

0307/80
0308/90 R
3DEA/21 R

P 0000
C3 00 3E 04 05 06 07 36


L 1F2E 3DBF

 ?
3DD4/21 R
D/M?
```

Example 4. The user resets the computer, types a D, then types VU, bringing in the utilities package, then types RZ, switching the computer to the Z-80, running under the Z-80 monitor. He then examines location 100 up, and loads it with the sequence 10 through 90. He types a legal character to exit the load sequence, types in address 300, and examines the contents of 300 upward. He again types in a legal character, causing it to exit that mode. He then loads location 3F4F upward with the start of move, end of move, and start of destination of the Z-80 move routine, which is a part of the Challenger III utilities package. He types in a legal character to exit this routine. He then sets up a vector, or jump, to the move routine at 3F55, and types RG. The move is nearly instantaneous and returns command to the Z-80 monitor. He examines location 300 up and verifies that the move did occur. Finally the user sets up location 3DEA and goes to this routine, which switches the machine from a Z-80 to a 6800. He then verifies that he is in the 6800 and sets up the GO vector to the Z-80, yielding the Z-80 prompter. Finally he sets up location 3DD4, which returns command to the 6502, thus showing how the user can move through to all processors.