T I 99/4
PERSONAL COMPUTER
SYSTEM SOFTWARE
Design Specification

29FEB 80

T I 99/4
PERSONAL COMPUTER
SYSTEM SOFTWARE
Design Specification

Personal Computer Division

January 21, 1980

Version 1.0

Table of Contents

1.0 Introduction

This document contains an overview of the TI 99/4 Personal Computer system software. It discusses details pertaining to the design and implementation of software for the product.

1.1 Purpose

The information provided herein is for use in maintenance of the system software for the 99/4 computer. It will serve as reference for the development of peripherals and for the design and development of future Personal Computer products. This document refers heavily to information in the other dosuments listed in section 2.0 and thus serves as a reference document.

1.2 Scope

This document is not intended to provide a detailed presentation of the hardware characteristics of the product although detail will be present where it is not available from other sources. More detail on the hardware characteristics can be found in the hardware specifications referred in 2.0. This document discusses the design and features of the 99/4 console software and the software interface provided for software which may be plugged in on the Solid State Software and peripheral ports.

2.0 Applicable Documents

Home Computer File Management Specification
(Vers. 2.4, 16 November 1979)

TMS 9918 VDP Video Display Processor Data Manual
(Revised 25 June 1979)

TMS 9919 Sound Generator Controller Specification
(Released 16 October 1979)

9900 Family Systems Design and Data Book (Learning Center Manual LCC-4400)

Graphics Programming Language Programmer's Guide
(Revised 3 December, 1979)

Home Computer System Memory, CRU, and Interrupt Mapping Specification (Revised 3 December, 1979)

Home Computer Software Development System Programmer's Guide
(Revised 6 November 1979)

TI 99/4 Home Computer BASIC Interpreter Design Specification
(Revision 1.0 7 December 1979)

Home Computer BASIC Language Specification
(Revision 4.1 12 April 1979)

Texas Instruments Home Computer User's Reference Guide
(Learning Center Manual LCB-4491)

I/O Bus Specification; document number 1037185

3.0 General Description

This section provides an overview of the system features provided in the hardware and software. The software exploitation of the hardware (e.g. ROM, RAM) is also discussed.

3.1 Hardware Description

This product uses a plastic case with a number of plug-in ports for software modules and hardware extensions. Specifically two major ports are provided: one for an application cartridge and one for peripheral units. The application cartridge port allows a user to plug in a solid state command module with up to 40K bytes of software. A peripheral unit (such as a disk or RS-232 interface) will contain it's own software Device Service Routine (DSR) within the unit but will not, in general, contain it's own microprocessor. Other ports allow connection of the Handheld Controllers (joysticks) and up to two audio cassette recorders for data or BASIC program storage.

Speech capability is provided as a peripheral device. The software required to access the speech peripheral from the BASIC language is provided in a application plug-in module. Solid-state applications for the 99/4 which use speech must include software to access the speech hardware in the application software module.

3.1.1 Keyboard

The 99/4 keyboard consists of 40 keys in a staggered array. The stagger of the keys is achieved by placing the keytops off center on the actual keys. The 99/4 keyboard does not include lower case letters or enough special characters to support international character set requirements. A detailed keyboard layout is given on page 165 of the Home Computer User's Reference Manual.

The keyboard does not include any hardware to scan the keys and provide interrupt signals to the 9900 microprocessor. Keystrokes are not buffered in any way and the keyboard is not scanned by the software except on request of an application program. An exception to this is that the keyboard is scanned for the shift-Q key on every VDP vertical retrace interrupt which occurs 60 times a second.

3.1.2 Application Software Port

The application module port provides plug-in capability for software provided by TI or third party authors (Milton Bradley is one) in Solid State Command Modules. A Command Module will contain 1 through 5 TMS0430 "GROM" chips and up to 8K bytes of

ROM in the form of 4764, 4732, 4716, etc. As mentioned in Appendix A the ROM space may be expanded through the use of a paging scheme.

The port is designed such that when a command module is plugged in the machine is reset. This reset appears identical to the reset that occurs when the computer is powered up. This is done because the chips in the command module may cause spurious signals on the data and address busses when the module is plugged in. In particular the GROM chips in the module will not be synchronized to the same internal address as the GROMs built into the console. If the system software would read the GROM address at this time (the GROM address is read often) garbage would be obtained.

3.1.3 I/O Port

The peripheral port provides all of the signal lines required to access the memory and CRU (Communication Register Unit as described in the 9900 Family Systems Design and Data Book) ports. The I/O port is fully described in the I/O Bus Specification.

3.1.4 CPU Memory Map

This section essentially duplicates information contained in the Home Computer System Memory, CRU and Interrupt Mapping Specification.

The CPU memory map is designed to provide a great deal of expansion for future accessories and for compatibility with future products. The memory map is as follows:

```
0000 +------------+
     !            !
     !            !  ROM contained in the console
     !            !
     +------------+
2000 !            !
     !            !  Unused
     !            !
     +------------+
4000 !            !
     !            !  ROM in peripheral (mapped)
     !            !
     +------------+
6000 !            !
     !            !  ROM in application module (optional)
     !            !
     +------------+
8000 !            !
     !            !  CPU RAM and memory mapped devices
     !            !
     +------------+
A000 !            !
     !            !  Unused
     ~~~~~~~~~~~~~~
     ~~~~~~~~~~~~~~
FFFF !            !
     +------------+
```

The peripheral ROM is mapped in to memory by selection through the CRU. The peripheral devices each have a unique CRU address. An access to that address maps the ROM for that device into the memory map. This addressing is further described in "Home Computer System Memory, CRU, and Interrupt Mapping Specification."

The block from 8000 to A000 contains the memory mapped devices and the 256 byte block of RAM on the CPU bus. Memory mapped devices (i.e. VDP, GROM, and SOUND chips) do not have fully decoded addresses. This causes certain locations in this block to be "unavailable". Actually the addresses of the chip memory mapped locations repeat in these lost areas. Only the recommended (primary) locations are explicitly identified below. Other locations are either unused or unavailable for use because of the partial decoding. All locations are one-byte data transfers except for CPU RAM. This area is subdivided as follows;

8300-83FF	256 byte CPU RAM
8400	Sound chip write data
8800	VDP read data
8802	VDP read status
8C00	VDP write data
8C02	VDP write address
9000	Speech read data
9400	Speech write data
9800	GROM page 0 read data
9820	GROM page 1 read data
9C00	GROM (GRAM) page 0 write data
9C02	GROM page 0 write address

The explanation of GROM pages and the expansion capability for GROM is given in section 3.1.6.

3.1.5 Video Display Processor

The Video Display Processor (VDP) is accessable as a memory mapped device. Access to the VDP chip is through certain memory mapped locations. The uses of these locations are read status, write address and read data. The memory addresses are listed in 3.1.4. Details on the use of the VDP chip from 9900 assembly language is given in "TMS 9918 VDP Video Display Processor Data Manual". The Graphics Programming Language Programmer's Guide describes the methods of accessing the VDP from that language.

3.1.6 GROM Memory Map

The GROM memory map comprehends 3 GROM chips within the TI 99/4 console and up to 5 chips in a plug-in command module. Each GROM chip contains 6K bytes of data but resides on a 8K byte boundary. This leaves 2K "holes" in the address space which cannot be used. The mask program of each GROM chip contains the base address of the data in that chip. This corresponds to a chip number as illustrated in the figure below.

```
             +-- +-------------------------+
             !   ! chip 7     base E000    !
             !   +-------------------------+
 application!   ! chip 6     base C000    !
     module  !   ~~~~~~~~~~~~~~~~~~~~~~~~~~~
             !   ~~~~~~~~~~~~~~~~~~~~~~~~~~~
             !   ! chip 3     base 6000    !
             +-- +-------------------------+ --+
                 ! chip 2     base 4000    !   !
                 +--------+-------+--------+   !
                 ! chip 1     base 2000    !   ! built into
                 +-------------------------+   !   console
                 ! chip 0     base 0000    !   !
                 +-------------------------+ --+
```

The 5 chip limitation in an application module can be overcome if additional hardware can be provided in the application. The additional hardware would probably require a larger than standard application plastics or a separate box on the end of a cable. It may also require a power supply since the amount of power that can be drawn from the GROM port is very limited. The additional circuitry can decode the the address lines for the ROM to provide several "pages" of GROM in the cartridge. These pages are than accessed by using different memory map locations for the device registers. The GROM register addresses listed in section 3.1.4 correspond to page 0 in this expanded scheme. The system software is designed to access 15 other pages as listed in the table below.

PAGE	READ DATA	READ ADDRESS	WRITE DATA	WRITE ADDRESS
0	9800	9802	9C00	9C02
1	9804	9806	9C04	9C06
2	9808	980A	9C08	9C0A
3	980C	980E	9C0C	9C0E
4	9810	9812	9C10	9C12
5	9814	9816	9C14	9C16
6	9818	981A	9C18	9C1A
7	981C	981E	9C1C	9C1E
8	9820	9822	9C20	9C22
9	9824	9826	9C24	9C26
10	9828	982A	9C28	9C2A
11	982C	982E	9C2C	9C2E
12	9830	9832	9C30	9C32
13	9834	9836	9C34	9C36
14	9838	983A	9C38	9C3A
15	983C	983E	9C3C	9C3E

The three GROM chips built into the console are accessed at any of the application pages. This scheme is illutrated in the figure below.

```
                     page 0              page 1           . . . . . . . .
               +-   +---------+        +---------+
               !    ! chip 7  !        ! chip 7  !
               !    +---------+        +---------+
    application!    ! chip 6  !        ! chip 6  !
       module  !    ~~~~~~~~~~~        ~~~~~~~~~~~
               !    ~~~~~~~~~~~        ~~~~~~~~~~~
               !    ! chip 3  !        ! chip 3  !
               +-   +---------+--------+---------+---      -+
                    !           chip 2              /       !
                    +-------------------------------/       !
                    !           chip 1              /       !  built into
                    +-------------------------------/       !    console
                    !           chip 0              /       !
                    +-------------------------------       -+
```

The initial program menu seletion searches all 16 pages for valid programs and enters the correct page according to the user selection. Programs may branch between pages with the CALL subprogram feature describes on page H-6 of the GPL Programmer's Guide.

3.2 Software Description

The 99/4 software includes support for applications modules containing interpretive Graphics Programming Language (GPL) object code. GPL is a powerful assembly type language designed especially to provide easy access to the special graphics and sound features of the 99/4 hardware. Included within the 99/4 are a BASIC programming language and an Equation Calculator.

3.2.1 Features

The console will support application modules containing programs written in GPL or BASIC (or a combination). The interpreter for GPL is contained in the system ROM. To reduce the size of application modules certain subroutines needed in the computer console have linkage provided for the use of applications modules. These subroutines are contained in the console GROM chips and include such features as trig functions etc.

A power-up program is contained in the console GROMs. This program initializes the system hardware and prompts the user for a menu selection of the desired application.

3.2.2 Supported Options

Linkage to peripheral devices is provided in the computer console. The 99/4 peripherals contain software to service the device. The linkage routines allow applications to call a Device Service Routine (DSR) for a device by name and in a device independent manner. A detailed description of the requirements for an applicaton to request service from a peripheral is described in the GPL Programmer's Guide and the Home Computer File Management Specification.

The 99/4 console contains 16K bytes of RAM attached to the VDP chip. This RAM is not expandable beyond 16K bytes. The software is self-configuring with respect to the amount of VDP RAM so that a 99/4 derivitive could be sold with less than 16K bytes of RAM. The amount of VDP RAM is placed in the GPL status block as part of the power-up sequence and must be tested by applications programs to ensure that enough RAM is available for the program to run. Peripheral devices may "steal" some of the VDP RAM at power-up time for use as buffers etc. The peripherals will modify the location in the status block which specifies the memory size to reflect the amount of memory pre-empted.

3.2.3 ROM Usage

The console ROMs contain the following software functions;

GPL interpreter
Radix 100 floating point package (+, -, * and /)
Keyboard scan routine
Subprogram/DSR search routine
Low level audio cassette Device Service Routine (DSR)
Interrupt processing including;
- Auto-sound
- Sprite motion
- Interval timer
- DSR interrupt

3.2.4 GROM Usage

The console GROMs contain the following software functions;

GPL support routines including;
- Subprogram and DSR linkage
- Arithmetic and trigonometric functions

System power up and program selection
High level Audio Cassette DSR
Keyboard character (translation) tables
User BASIC language editor and interpreter
Equation Calculator interface to the BASIC interpreter

3.2.5 CPU RAM Usage

The CPU RAM has several uses as follows;

Free space for use by GPL applications
9900 workspace area (one workspace)
Partial workspace for interrupt handling
Work area for the GPL interpreter
GPL status block
Device Service Routine work area
CALL routine work area

These areas are described in detail in the GPL Programmer's Guide. The 9900 code uses only one workspace for all processing. Interrupts are only allowed when most workspace registers can be destroyed. The interrupt is taken into a second workspace where only registers 13 through 15 are used to save the interrupt status. Immidiately after the interrupt is taken a LWPI instruction is executed to restore the context back to the one workspace. Another LWPI instruction is executed before the RTWP to return from the workspace context.

The unused space in the interrupt workspace is used by the GPL interpreter to save various information such the last key pressed on the keyboard in order to debounce the keyboard. The two workspaces (one is partial) and this work area occupy 32 bytes of the 256 byte CPU RAM from address >83C0 to >83FF.

The GPL status block occupies location >836E through >837F. The use of these locations is documented in section 3.3.1 of the GPL Programmer's Guide. Locations >836E and >836F contain the Floating Point Stack pointer as described in Appendix K of the same manual.

3.2.6 VDP RAM Usage

The system utilization of VDP RAM can be closely controlled by a GPL application program. Many of the data structures for display on the screen are programmable through VDP registers and are described in the GPL Programmer's Guide. Certain data structures for sprites and the floating point roll out area are at fixed locations but may not be used by a particular application program.

The use of the VDP RAM by the BASIC language interpreter is described in the Home COmputer BASIC Interpreter Design Specification.

4.0 Console Software

This section provides an overview of the system features provided by the software contained in the 99/4 computer console. These features are provided to support application program modules and access to peripheral units from application modules.

4.1 System Power-up Sequence

Most of the system power-up initialization is written in the interpretive GPL language. When the system is powered up the level 0 interrupt is taken. This interrupt vactor is at address 0 and loads the workspace to >83E0. After loading R13 with the GROM read address the GPL interpreter starts interpreting GPL codes at GROM location >20. The GPL code performs the rest of the initialization as follows:

- Load R15 with the VDP write address address (>8C02)
- Load R14 with status flags
- Clears the sound list indicator used for auto-sound (location >83CE)
- Turns off the speech chip (when attached) and the sound generators
- Initializes the two GPL stacks (subroutine and data) in the status block
- Initializes the VDP registers to default values
- Zeros much of CPU RAM >8300 - >8371, >8382 - >83BF, >83C2 ->83C9
- Clears the Handheld Units (not needed)
- Enable the audio gate so that the cassette data will be heard on the monitor speaker
- Enable the Handheld Unit interrupt (not needed)
- Enable the VDP 60 Hertz interrupt
- Enable the external interrupt
- Enable audio cassette motors (to on state)
- Issue a beep to signal powered up
- Determine the VDP memory size and set the VDP register bit accordingly
- Clear the first 4K bytes of VDP RAM
- Load the default color and character tables
- Initialize all keyboards by scanning them
- Display the power-up screen (screen turned off)
- Call possible Power-up screen modification routine in cartridge. This is done for foreign languages.
- Call power-up routines in ROM and GROM (see a description of the GROM/ROM header). Power up routines may modify the VDP RAM size placed in location >8370.
- Turn on VDP screen (it is turned off during initialization)
- Wait for a key on the console keyboard then beep
- Build a list of the available programs including looking for a library. Only GROM is searched.
- Display the program menu screen (screen turned off)
- Call possible menu screen modification routine in cartridge. This is done for foreign languages.

Turn the screen on
Wait for user menu selection
Branch to starting address of program

If the console only contains GROM 0 (as a future product) and no cartridge is inserted then the menu screen will display "INSERT CARTRIDGE" instead of the menu. This cannot happen in the 99/4 because BASIC and the Equation Calculator are always present.

A user selected program must always be a GPL program or must at least be started in GPL. The GPL program may initiate assembly language by the XML intruction or may initiate BASIC from GROM as described in the Home COmputer Software Development System Programmer's Guide.

4.2 GPL Application Support

The GPL application support consists of the GPL object code interpreter and the callable console GPL subroutines. In addition the perpheral support described in section 4.5.3 allows access to peripherals from an application program. These applications are developed on a 990/10 or 990/4 minicomputer using the development aids described in Appendix C. Development of GPL application programs by the end user of the computer is not supported and there is no plan to do so for the general user.

4.2.1 GPL Interpreter

The GPL object interpreter consists of 9900 assembly routines to interpret the object code generated by the GPL assembler. A floating point arithmetic package is also included. This package consists of assembly language routines accessable by the GPL XML instruction and GPL routines accessable by the CALL statement. The floating point routines use an 8-byte radix 100 floating point representation which provides at least 13 digits of significance.

4.2.2 Support Subroutines

Certain GPL subroutines in the 99/4 console which could be useful to application programs are made accessable by a branch table at a fixed location in the console GROMs. The use of this branch table provides a fixed address to enter the routines even if the console code changes. The routines are discussed in Appendix K of the GPL Programmer's Guide. The complete list of routines is given below;

address	name	use	ref
10	LINK	Link to subprograms and DSRs	GPL App H
12	RETN	Return from subprogram or DSR	GPL App H
14	CNS	Convert floating point to ASCII	GPL App K
16	CHR1	Load 8 dot high character set	GPL App H
18	CHR2	Load 6 dot high character set	GPL App H
1A	BWARN	Warning message from BASIC subprogram in GPL	
1C	BERR	Error message from BASIC subpro-gram in GPL	
1E	BEXEC	Begin execution of GROM BASIC program	
20	PWRUP	Restart system (used at power up)	
22	INT	Greatest integer of a floating point number	GPL App K
24	PWR	Exponentiation	GPL App K
26	SQR	Square root	GPL App K
28	EXP	Inverse natural logarithm	GPL App K
2A	LOG	Natural logarithm	GPL App K
2C	COS	Cosine	GPL App K
2E	SIN	Sine	GPL App K
30	TAN	Tangent	GPL App K
32	ATN	Arctangent	GPL App K
34	TON1	Good prompt tone	GPL App H
36	TON2	Bad prompt tone	GPL App H

Certain 9900 routines are accessable from GPL through the XML instruction. Many of these routines are also accessable as subroutines (with R11 as the link) to the 9900 code in the console. However they are not available to external 9900 code (in peripherals or Command Modules) because their addresses are not fixed. The XML routines are as follows;

XML number	name	use	ref
00	unused	unused	
01			GPL App K
02			GPL App K
03			GPL App K
04			GPL App K
05			GPL App K
06	FADD		GPL App K
07	FSUB		GPL App K
08	FMUL		GPL App K
09	FDIV		GPL App K
0A	SCOMP		GPL App K
0B	SADD		GPL App K
0C	SSUB		GPL App K
0D	SMUL		GPL App K
0E	SDIV		GPL App K
0F	SCOMP		GPL App K
10	CSN		GPL App K

11			GPL App K
12	CFI	Rounded convert f.p. to integer	GPL App K
13			
14			

More information on the general use of XML instructions is given in the H/C System Memory, CRU, and Interrupt Mapping specification.

4.2.3 Application Configuration

Application programs may only be contained in GROM. The user selects an application program from the system menu. A program is placed in the system menu by its reference in a GROM header. The GROM header is defined in Appendix H of the GPL Programmer's Manual. An example of a GROM header for an application program is given below.

```
        GROM 3
        ORG  0
        DATA >AA              HEADER IDENTIFIER
        DATA 0                VERSION NUMBER (NOT REALLY USED)
        DATA 1                NUMBER OF PROGRAMS (NOT REALLY USED)
        DATA 0                NOT USED (RESERVED)
        DATA #0               ADDRESS OF POWER-UP HEADER (NONE HERE)
        DATA #PROG1           ADDRESS OF APPLICATION PROGRAM HEADER
        DATA #0               ADDRESS OF DSR HEADER (NONE HERE)
        DATA #0               ADDRESS OF SUBPROGRAM HEADER (NONE HERE)
        DATA #0               ADDRESS OF INTERRUPT LINK (NONE IN GROM)
        DATA #0               UNUSED
*
PROG1   DATA #0               LINK TO NEXT HEADER (NONE)
        DATA #START           ENTRY POINT
        DATA 19               LENGTH OF PROGRAM NAME
        DATA :APPLICATION PROGRAM:   PROGRAM NAME
```

The GROM header may be placed at the beginning (address 0) of any GROM chip. When the system starts an application program the system memory is initialized to mostly zeros as described in Appendix H of the GPL Programmer's Guide.

A link editor is not provides for GPL to resolve references between separately assembled modules. A technique that has proven useful is to place a branch table a the beginning of a module for those routines which are referenced by other assemblies. In this way the addresses of external routines remain fixed although the actual routine address may move within the separate assembly.

4.3 BASIC Interpreter

The BASIC interpreter is a GPL applicatiom program which is built into the 99/4 console. To provide sufficient speed many of the core execution routines are written in 9900 assembly language. Linkage to these routines is through system defined XML instructions. All of the edit and symbol table generation portions of the BASIC interpreter are written in GPL. Much more detail of the design of the BASIC interpreter can be found in the TI 99/4 Home Computer BASIC Interpreter Design Specification.

4.4 GROM BASIC Program Support

The 99/4 computer allows BASIC programs to be placed in GROM cartridges and executed from there. This is provided through extensions to the BASIC interpret which allow a program to be interpreted either from GROM or VDP RAM. The program contained in the GROM is in the same memory image as a program would be if contained in VDP RAM. A processing program on the 990/10 computer converts the BASIC source program into the memory image form. A GROM BASIC program is limited to 1500 lines since one of the GROM tables cannot cross a GROM boundary and contains 4 bytes for each line in the program.

4.4.1 Program Execution

A GROM BASIC program contains a GPL program header since all applications must at least be initiated in GPL. For a GROM BASIC program this GPL header places a mesage on the screen and then calls the BASIC interpreter. The BASIC builds the symbol table and then starts interpreting the BASIC program. Program execution is identical to that of a program contained in VDP RAM with the exception that more memory is available and a program can be much larger than if placed in the VDP RAM.

Because of a bug in BASIC the CALL CHAR statement will not work in a GROM BASIC program. The space is allocated properly in the character table but the character definition is not properly placed in the character table. A solution to this problem is to allocate the space with the CALL CHAR statement (using the highest numbered character) and use a custom GPL subprogram to set up the character defintions.

4.4.2 Hybrid GPL/BASIC Programs

A GROM BASIC program may call GPL subprograms which are also contained in the GROM cartridge. The GPL subprograms are defined through the GROM header and are called by name with the CALL statement. Typical GPL subprograms would be to avoid the CALL CHAR bug described above or to do screen formatting when more speed is needed.

Another special purpose for hybrid programs is to speed the sysbol table generation process. The symbol table generation can be very slow for a long program since it is written in GPL. A trick that can be done is to include a second BASIC program in the GROM cartridge which defines the same BASIC variables. The small program is the one selected when BASIC is called. The first executable statement in the small program is a CALL to a subprogram which changes all of the BASIC pointers to start execution of the large program. The programmer must be very careful to include all BASIC variables and user defined functions in the small program including the same dimensions etc.

The construction and use of hybrid GPL/BASIC programs is not documented at this time. Programming experiments have shown that pure GPL programs can be developed much easier with only slightly more time in most cases.

4.5 Peripheral Support

The 99/4 system provides for peripheral Device Service Routines to be contained in peripheral ROM or in system or application cartridge GROM. ROM DSRs are written in 9900 assembly language while GROM DSRs are written in GPL. A DSR may contain a power up routine if that is required for the device. An interrupt entry is only provided for ROM (9900 assembly language) DSRs.

4.5.1 General Concepts

The three entry points (power up, I/O call, interrupt) are defined in the ROM/GROM header. The same header structure is used in either ROM or GROM although some fields are not used in one or the other. For example the interrupt field in a GROM header and the application program field in a ROM header are not used. The ROM header must be placed at the beginning (address >6000) of a ROM DSR.

```
         DATA >AA00          HEADER IDENTIFIER, VERSION NUMBER
         DATA 0              NOT USED
         DATA PWR            ADDRESS OF POWER-UP HEADER
         DATA 0              NOT USED
         DATA DSR            ADDRESS OF DSR HEADER
         DATA 0              ADDRESS OF SUBPROGRAM HEADER (NONE HERE)
         DATA INT            ADDRESS OF INTERRUPT LINK
         DATA 0              UNUSED
*
PWR      DATA 0              LINK TO NEXT HEADER (NONE)
         DATA PWRUP          POWER UP ENTRY
*
DSR      DATA 0              LINK TO NEXT HEADER (NONE)
         DATA START          ENTRY POINT
         BYTE 19             LENGTH OF PROGRAM NAME
         TEXT 'APPLICATION PROGRAM'  PROGRAM NAME
*
INT      DATA 0              LINK TO NEXT HEADER (NONE)
         DATA INTENT         INTERRUPT ENTRY
```

The power up and interrupt headers do not have name entries in them.

4.5.2 Power-up

The power up entry of every DSR is executed before the power up "press any key" screen is placed on the monitor. ROM power up routines are executed before GROM power up routines. Power up routines may use CPU RAM locations >8304 through >83BF. Note that these locations cannot all be used in the other DSR entries. ROM power up routines may use registers R0 through R10 at will. Other registers contain the following;

```
R11  return address
R12  CRU base address
R13  GROM read date address
R14  system flags
R15  VDP write address address
```

R12 must contain the same value on the return to monitor as it contained on entry to the DSR. DSRs will often change the address in R12 but since peripherals reside on >100 boundaries R12 can be restored with an ANDI R12,>FF00 instruction.

4.5.3 I/O Calls

During an I/O call the DSR may use CPU RAM locations >834A through >836D. A ROM DSR may also use >83EA through >83EF if it does not enable interrupts. A ROM device service routine may use registers R0 through R10 at will. Other registers contain the following;

```
R11  return address
R12  CRU base address
R13  GROM read date address
R14  system flags
R15  VDP write address address
```

R12 must contain the same value on the return to monitor as it contained on entry to the DSR. DSRs will often change the address in R12 but since peripherals reside on >100 boundaries R12 can be restored with an ANDI R12,>FF00 instruction.

A GPL I/O routine returns to the monitor with a CALL RETN (see section 4.2.2). Note that a CALL must be used and not a branch. A ROM (9900) I/O routine must return with the following code.

```
INCT R11
RT
```

4.5.4 Interrupts

When an external interrupt occurs the system interrupt handler enters the interrupt entry of each peripheral device. Each device service routine must determine if its device requires service and handle it. An interrupt routine returns to the monitor with a 9900 "RT" instruction.

ROM interrupt routines may use R0 through R7 and R10 at will. R8 may be used but must be cleared (set to zero) before exiting the interrupt routine. R9 cannot be destroyed. The use of R11 through R15 is as described above. Other areas of CPU RAM are not available for use in an interrupt routine.

Appendix A *Future Expansion*

Appendix B Multi-lingual Support

Appendix C <u>Software Development Methods</u>

Appendix D Compatibility

In general there will be no designated compatibility between the TI99/4 and any previous computers. There is no compatibility with products of Texas Instruments calculator line. The BASIC language is very similar to other microcomputer BASICs, however, most programs will need some changes to run on the 99/4. Any programs which use the graphics capibilities of other computers will need to be totally rewritten to run on the 99/4. The memory format of a BASIC program is unique as are most personal computers. The image which is recorded to mass-storage in a SAVE command is this memory image which limits the capability of transporting BASIC programs to other computers even if they could read our mass storage media.

Application cartridges from the 99/4 computer will run on the 99/3 when a GPL interpreter personality module is inserted. This personality module will be bundled with the user programmable BASIC language since the BASIC interpreter is partly written in GPL.

Peripheral devices for the 99/4 including the Thermal Printer, Mini-floppy Disk and RS-232 Interface will not work on any other personal computer. Peripherals from other computers will not work on the 99/4 except for those with RS-232 interfaces which can be attached to our RS-232 peripheral. The media of other mass storage peipherals (audio tape or disk) will not be transportable to the 99/4.

Very little groudwork has been done to determine the requirements to make future personal computer products compatible with the 99/4 although this is a necessary goal.

Appendix E <u>Hardware Diagnostic Methods</u>