DEVICE SERVICE ROUTINE SPECIFICATION FOR THE TI-99/4(A) PERSONAL COMPUTER

Consumer Group
Mail Station 5890
2301 N. University
Lubbock, Texas 79414

TEXAS INSTRUMENTS
INCORPORATED

Date: March 28, 1983
Version 2.0

TABLE of CONTENTS

Paragraph Title

SECTION 1 INTRODUCTION

1.1 Interface in General

SECTION 2 I/O BUS

2.1 I/O Bus Pin Assignments and Descriptions

SECTION 3 HARDWARE STRUCTURE OF DSR

3.1 DSR ROM
3.2 CRU Mapping

SECTION 4 SOFTWARE STRUCTURE OF DSR

4.1 Symbol Definition Block
4.2 Header and Linkage Block
4.2.1 A Sample Program for Header and Linkage Block
4.3 Power-up Routine
4.4 Interrupt Routine
4.5 Main Device Service Routine

SECTION 1

INTRODUCTION

This document describes the hardware and software interfaces between the console and peripherals of the TI-99/4 and TI-99/4A Home Computers. The purpose of this document is to provide a reference for third parties who want to design new peripherals for the TI-99/4A family.

1.1 Interface in General

The 99/4x console has a 44-pin male card edge-connector, called the I/O Bus, on its right edge. The I/O Bus contains all the information needed for the console to communicate with the peripherals. It is suggested that each peripheral have its own power supply instead of sharing the voltages with the console through the I/O Bus.

Each peripheral should include a nonvolatile Device Service Routine (DSR) software package to drive the peripheral. The DSR communicates with the console software through certain memory locations. The console software sets up information in these memory locations and passes it to the user-selected peripheral. From there on, the selected peripheral's DSR should have the capability to interpret the information set up by the console software, physically drive the peripheral, and pass the peripheral's data or status to the console software if necessary.

SECTION 2

I/O BUS

The purpose of this section is to describe the pin assignments of the I/O Bus, functions of I/O signals, sources of Output Signals, and destinations.of Input Signals.

2.1 I/O Bus Pin Assignments and Descriptions

Pin #	Notation	I/O	Description
1	+5V	O	5V Power Supply
2	SBE	O	Low when MPU read from >90xx or write to addreses >94xx memory
3	$\overline{\text{RESET}}$	O	Master Reset, low active
4	$\overline{\text{EXT}}$ $\overline{\text{INT}}$	I	External Interrupt, low active
5	A5	O	Address Bit 5
6	A10	O	Address Bit 10
7	A4	O	Address Bit 4
8	A11	O	Address Bit 11
9	DBIN	O	Derived from MPU's DBIN pin, same parity
10	A3	O	Address Bit 3
11	A12	O	Address Bit 12
12	$\overline{\text{READY}}$	I	If device or memory is ready after being addressed by MPU in memory R/W cycle, device or memory should cause this input to go high
13	$\overline{\text{LOAD}}$	I	To 9900's $\overline{\text{LOAD}}$ pin
14	A8	O	Address Bit 8
15	A13	O	Address Bit 13
16	A14	O	Address Bit 14
17	A7	O	Address Bit 7
18	A9	O	Address Bit 9
19	CRUOUT/A15	O	CRU Output/Address Bit 15, LSB
20	A2	O	Address Bit 2
21	GND		Signal Ground

Pin #	Notation	I/O	Description
22	$\overline{\text{CRUCLK}}$	O	Inversion of MPU's $\overline{\text{CRUCLK}}$ pin
23	GND		Signal Ground
24	$\overline{\text{PHI3}}$	O	Inversion of Phase 3 Clock
25	GND		Signal Ground
26	$\overline{\text{WE}}$	O	Derived from MPU's $\overline{\text{WE}}$ pin, same parity
27	GND		Signal Ground
28	$\overline{\text{MBE}}$	O	Low when MPU addressing >4000 - >5FFF Memory
29	A6	O	Address Bit 6
30	A1	O	Address Bit 1
31	A0	O	Address Bit 0, MSB
32	$\overline{\text{MEMEN}}$	O	Derived from MPU's $\overline{\text{MEMEN}}$ pin, same parity
33	CRUIN	I	CRU Input to MPU
34	D7	I/O	Data Bus Bit 7, LSB
35	D4	I/O	Data Bus Bit 4
36	D6	I/O	Data Bus Bit 6
37	D0	I/O	Data Bus Bit 0, MSB
38	D5	I/O	Data Bus Bit 5
39	D2	I/O	Data Bus Bit 2
40	D1	I/O	Data Bus Bit 1
41	IAQ	O	MPU's IAQ pin
42	D3	I/O	Data Bus Bit 3
43	-5V	O	-5V Power Supply
44	AUDIO IN	I	To Sound Generator Controller's AUDIO IN pin

SECTION 3

HARDWARE STRUCTURE OF DSR

3.1 DSR ROM

Normally, a DSR is written in 9900 Assembly Language and is installed in a ROM, which is itself part of the TI-99/4A family's peripheral. All DSRs must begin at address >4000 and should not exceed >5FFF. All of the eight data pins of the DSR ROM must be buffered before being connected to the data bus of the I/O Bus. This buffer is enabled and disabled by a preassigned CRU output bit which is controlled by the console software, so that not more than one DSR is accessed at any time.

3.2 CRU Mapping

The CRU I/O is used by the system to access the peripherals, if the speed of data transfer is not too crucial. The decoding format for the CRU addressing is indicated as:

```
A0 A1 A2    A3 A4 A5    A6 A7    A8 A9 A10 A11 A12 A13 A14    A15/CRUOUT
0  0  0     User ID     Device        CRU I/O Bit Decode           0
```

The CRU address space, ranging from >0 to >1FFE with A0, A1, A3, and A15 unused, is broken into eight blocks of 512 bits each. The User ID, indicated through the Address Decoding of A3, A4, and A5, represents each of the 512 bit blocks. They are assigned as follows:

MPU Address	A3	A4	A5	Assignment
0000-03FE	0	0	0	Console Use
1000-13FE	1	0	0	TI Peripheral Space #1
1400-17FE	1	0	1	TI Peripheral Space #2
1800-1BFE	1	1	0	TI Peripheral Space #3
1C00-1FFE	1	1	1	TI Peripheral Space #4

The standard device decoding through A6 and A7 gives a total of four device blocks within each User ID Block. The CRU I/O bit decoding through A8-A14 allows 128 addressable bits each of input and output. In the 128 CRU Output bits, the first bit, with A8-A14 all 0's, is reserved for enabling the Data Buffer of the DSR ROM in each peripheral as mentioned in the last subsection. Setting this bit to logic one enables the DSR ROM, while setting it to logic zero disables the DSR ROM.

SECTION 4

SOFTWARE STRUCTURE OF DSR

A DSR must follow a specific format in order to communicate with the console software properly. The purpose of this predefined format is to let the console software have the least overhead and the DSR have the maximum flexibility in device servicing.

A DSR, in general, contains the following elements:

* Symbol Definition Block

* Header and Linkage Block

* Main Device Service Routine

* Power-up Routine(optional)

* Interrupt Routine(optional)

Each of the above elements will be discussed in detail in the following sections.

4.1 Symbol Definition Block

The Symbol Definition Block serves two purposes. First, it equates frequently used data or addresses with symbols for ease of recognition. Secondly, it specifies the CPU RAM location in a certain way so that each DSR can be used around future and existing TI-99/4A family members, such as the TI-99/4, with the least modification. The first purpose is common in every program, but the second one needs more explanation.

Each model in this family has a different memory structure within the console. Therefore, care must be taken in handling memory addressing in a DSR so that the DSR can support future Home Computer models. The console software always enters the DSR through the instruction:

```
BL *R9
```

where R9 contains the DSR entry address. Upon entry of the DSR, the Workspace Pointer contains the beginning address of the Register File, which the console software uses. If all the CPU RAM which the DSR may access is predefined with respect to this Workspace Pointer, the DSR does not have to know the memory map of each console. The DSR can address the CPU RAM through Indexed Addressing via the Workspace Pointer. For example, if one wants to move the contents of a CPU RAM (having a displacement [DISP] with respect to the Workspace Pointer) to R0, do the following:

```
ENTRY   STWP  R4               Entry of the DSR
        .
        .
        MOV   @DISP(R4),R0
        .
        .
```

Some of the frequently used CPU RAM's symbols and locations are listed below.

```
PAD     EQU   ->E0             Start of CPU RAM in 99/4 console
FAC     EQU   PAD+>4A          Start of 36 bytes available to DSR
R0LB    EQU   PAD+>E1          Lower byte of R0
R1LB    EQU   PAD+>E3          Lower byte of R1
 .
 .
R15LB   EQU   FAC+>FF          Lower byte of R15
OPCODE  EQU   FAC+>0           Beginning of PAB, I/O operation code
FLGSTS  EQU   FAC+>1           PAB - Flag/Status
BUFADR  EQU   FAC+>2           PAB - Data Buffer Address
LRECLN  EQU   FAC+>4           PAB - Logical Record Length
CHRCNT  EQU   FAC+>5           PAB - Character Count
RECNUM  EQU   FAC+>6           PAB - Record Number
SCNOFF  EQU   FAC+>8           PAB - Screen offset
OPTLEN  EQU   FAC+>9           PAB - Option Length
DEVLEN  EQU   FAC+>10          PAB - Device Length
PABVDP  EQU   FAC+>12          PAB - Pointer to PAB in VDP RAM
VWA     EQU   >8C02            Address for VDP Write-Adddress operation
GRD     EQU   >9800            Address for GROM Read-Data operation
 .
 .
```

4.2 Header and Linkage Block

The DSR must contain a certain header starting at >4000 so that the linkage to the console software can be established properly. This header contains the following information:

* Validation flag(>AA)

* Name(s) of device(s) being serviced by this DSR (the device name should be 7 characters or less)

* Entry point(s) of the device(s) being serviced by this DSR

* Entry point of Power-up Routine, if necessary

* Entry point of Interrupt Routine, if necessary

4.2.1 A Sample Program for Header and Linkage Block.

This section gives an example of the Header and Linkage Block in a typical DSR. The syntax of DX10 Assembler is obeyed in this program. DX10 Assembler Language is similar to, but not identical with, the TI Editor/Assembler. Information on the language can be found in the 9900 Family Systems Design and Data Book.

```
*****************************************************
*                                                   *
*       Sample Program Showing DSR Header Format    *
*                                                   *
*****************************************************
        RORG >4000            Start of DSR
        BYTE >AA              Telling console this is a valid DSR
        BYTE 1                Version #
        DATA 0                Not used in DSR calls, so leave it
                                '0'
        DATA PWRLNK           Power-up Routine's Link. 'PWRLNK'
                                is replaced by '0' if power-up
                                set up is not necessary
        DATA 0                Not used in DSR calls, leave it '0'
        DATA DSRLNK           DSR Link, CAN'T be '0' here
        DATA 0                Not used in DSR calls, leave it '0'
        DATA INTLNK           Interrupt Routine's Link.  'INTLNK'
                                is replaced by '0' if interrupt
                                is not used
        DATA 0                Not used in DSR calls, leave it '0'
*
*    Note: This Power-up Routine can be omitted if power-up
*          initialization is not necessary for this peripheral
*
PWRLNK  DATA 0                Linkage, set to 0
        DATA PWRUP            Entry point of power-up routine
        BYTE 0                Name length, set to 0
        EVEN
        .
        .
PWRUP   .                       Entry of power-up routine
        .
        .
        B    *R11               Return to console software through
        .                         *R11  (It must not be destroyed)
        .
*
*    Note: This Interrupt Routine can be omitted if Interrupt
*          Request is never issued by this peripheral
*
INTLNK  DATA 0                  Linkage, set to 0
        DATA INTDSR             Entry point of the Interrupt Routine
        BYTE 0                  Name length, set to 0
        EVEN
        .
        .
INTDSR  .                  Entry of interrupt routine:
        .                    if interrupt_request_from_this_peripheral
        .                            = true
        .                    then
        .                       begin interrupt_service ;
        .                             reset Interrupt_request;
        .                             goto INTEND
        .                       end
        .                    else goto INTEND;
        .
INTEND  B    *R11               Return to console software through
        .                         *R11  (It must not be destroyed)
        .
*
*       *** Main Device Service Routine. Assuming three devices,
*       *** with names 'DEVICE', 'DEVIC/1', 'DEVIC/2', are
*       *** supported in this peripheral.
*
DSRLNK  DATA DSRLK2             Linkage to next device field
        DATA ENTRY1             Entry point of 1st device
        BYTE 6                  Name length of 1st device
        TEXT 'DEVICE'           Name of 1st device
        EVEN
DSRLK2  DATA DSRLK3             Linkage to next device field
        DATA ENTRY2             Entry point of 2nd device
        BYTE 7                  Name length of 2nd device
        TEXT 'DEVIC/1'          Name of 2nd device
        EVEN
DSRLK3  DATA 0                  Linkage to next device field(none)
        DATA ENTRY3             Entry point of 3rd device
        BYTE 7                  Name length of 3rd device
        TEXT 'DEVIC/2'         Name of 3rd device
        EVEN
        .
        .
ENTRY1  .                      Entry of 1st device servicing
        .
        .
ENTRY2  .                      Entry of 2nd device servicing
        .
        .
ENTRY3  .                      Entry of 3rd device servicing
        .
        .
EXIT    INCT R11               Return to console software through
        B    *R11              *R11 (It must not be destroyed).
        .
        .
```

4.3 Power-up Routine

For some peripherals, it is necessary to initialize the hardware at power-up time. It is suggested that a power-up routine be included to do the initialization through software for these pheripherals. Power-up Link should be set in the Header Field as described above.

Power-up routines are executed whenever the system is reset by either hardware or software. The console software searches all peripheral DSRs for power-up routine addresses and executes them, if they are found. Each power-up routine can use R0-R10. Upon entry, R12 is set to the beginning address of the CRU space for that peripheral DSR (note that this address is used to enable the DSR ROM). R11 contains the return address. R13 and R15 contain the memory-mapped addresses of GROM Read Data and VDP Write Address, respectively. All VDP and GROM operatons can be indexed from these 2 registers. The power-up routine may use VDP RAM from 0 to the location pointed to by CPU RAM >70, offset from (Workspace Pointer - >E0). It can use all CPU RAM except location >55, >6D and >C0 through >DF offset from (Workspace Pointer - >E0).

All power-up routines must return with:

```
B *R11
```

4.4 Interrupt Routine

If the peripheral issues Interrupt Request to the 9900 CPU, the DSR should also include an interrupt routine and Interrupt Link in the Header as described above. Every interrupt that is not recognized as a console interrupt (VDP or 9901 Timer) causes the console software to execute every interrupt routine it can find. The interrupt routine must check to see whether the Interrupt Request is raised by this peripheral. If it is not, exit by:

```
B *R11
```

Upon entering the DSR interrupt service routine, the Workspace Pointer is >83E0. The interrupt routine may use R1-R8, and R10. R8 must be cleared before exiting the DSR. All other registers must be kept as they are. The contents of R11-R15 are the same as those of the Power-up Routine Section. The interrupt routine and Main DSR can split the allocation of CPU RAM from >4A to >6D offset from (Workspace Pointer - >E0).

All interrupt routines end with B *R11. Interrupt Request raised by the peripheral must be cleared before exit.

4.5 Main Device Service Routine

When the DSR processing is completed, exit by performing the following:

```
INCT R11
B *R11
```