Technical Information on Toshiba Laptops

Jonathan Buzzard jonathan@buzzzard.org.uk

April 2001 Version 1.3

DISCLAIMER: The information in this document has been obtained by reverse engineering the software supplied by Toshiba for their portable computers in strict accordance with the European Council Directive 92/250/EEC on the legal protection of computer programs, and it's implementation into English Law by the Copyright (Computer Programs) Regulations 1992 (S.I. 1992 No.3233). As such it is likely to be incomplete and comes with no warranties as to it's accuracy.

Overview

On a Toshiba laptop the pair of ports E4h and E5h can be used to get and set many of the features of the laptop. For the most part thought it is better to use the System Configuration Interface and Hardware Configuration Interface where possible. These methods are often identical although the underlying hardware may be very different. As a result any software using the SCI or HCI is likely to work across a wider range of laptops than if it manipulated the laptops chipset directly. Unfortunately a number of features can only be manipulated by accessing the laptops chipset directly though the E4h and E5h ports.
The two ports work in a similar manner to the CMOS RAM ports 70h and 71h. First we write to port E4h which of the chipsets registers we wish to manipulate. Then we read or write as appropriate to port E5h to get or set the register. All the Toshiba code that manipulates these ports slows down the I/O with short jumps, however this does not appear to be necessary. It is however essential that interrupts are disabled while reading and writing to ports 0xE4 and 0xE5. Failure to do so could lead to a system crash if two or more programs attempt access to these ports at the same time.

Machine Identification Number

All Toshiba laptops have a unique identification number held in the BIOS that can be used to identify the model on which a program is running. There are two different methods by which this number can be retrieved. The method needed depends upon the model in question. The machine identificastion number is always an unsigned 16 bit number.

Original Method

The first method is the simplest and is the method reqired on all older models. For this method the identification number is stored in two bytes of the BIOS memory area.
The following pseudo code show how to retrieve the identification number

```
READ low FROM MEMORY F000:FFFA
READ high FROM MEMORY F000:FFFE
id EQUALS (0x100*high)+low
```

SCT Table method

On newer models (including all current ones) a different method is required. This method is called the SCT Table method by Toshiba. This method of retrieving the machine identification number needs to be user when the original method returns a value of 0xFC2F.

```
LOAD REG ax WITH 0xC000
LOAD REG bx WITH 0x0000
LOAD REG cx WITH 0x0000
READ BYTE FROM 0xB2 INTO REG al
READ WORD temp FROM MEMORY 0xF0000+bx
READ WORD temp FROM MEMORY 0xF0009+bx+temp
READ WORD temp FROM MEMORY 0xF000a+temp
id EQUALS ((temp%0xFF)*0xFF)+(temp/0xFF)
```

After reading the byte from OxB2 into the al register under Microsoft Windows the bx register contains the value 0xe6f5. This has been verified on a Satellite Pro 430CDT, Tecra 750CDT, Tecra 780DVD and Satellite 310CDT. However I have been unable to dublicate this behaviour under Linux and consequently have always had to fudge the value to be 0xe6f5.

Fan

There are two methods of controlling the fan. However it would appear that only the fan on the Portage 610CT and Tecra 700x cannot be controlled using the Hardware Configuration Interface. The method needed to control the fan differs between the Portage 610CT and Tecra 700x. The GetMachineID function from my Hardware Configuration interface library provides a method to determine which laptop model you are running on and to select the appropriate method. It is however preferable to use the Hardware Configuration Interface wherever possible.

Portage

While I call this the Portage method, this method does in fact work on a wider range of models, and was the method used in all version of my fan program prior to version 2.0. This method works on the Satellite Pro 400x, 410x, 420x, 430x, the Satellite 100x, 110x, 200x, 210x as well as the Portage 610CT. It would appear that the distinguishing feature of these models being that they do not have a PCI bus. The Portage 610CT has a machine ID of 0xFCCB.
The following pseudo code will turn the fan on:

```
DISABLE INTERRUPTS
WRITE 0xBE TO PORT 0xE4
READ byte FROM PORT 0xE5
CLEAR BIT 0 OF byte                    (i.e. least significant bit)
```

```
WRITE 0xBE TO PORT 0xE4
WRITE byte TO PORT 0xE5
ENABLE INTERRUPTS
```

The following pseudo code will turn the fan off:

```
DISABLE INTERRUPTS
WRITE 0xBE TO PORT 0xE4
READ byte FROM PORT 0xE5
SET BIT 0 OF byte
WRITE 0xBE TO PORT 0xE4
WRITE byte TO PORT 0xE5
ENABLE INTERRUPTS
```

The following pseudo code will get the current status of the fan:

```
DISABLE INTERRUPTS
WRITE 0xBE TO PORT 0xE4
READ byte FROM PORT 0xE5
ENABLE INTERRUPTS
IF BIT 0 OF byte SET THEN ON ELSE OFF
```

Tecra

This methods is aimed at the Tecra 700CDS and Tecrs 700CDT only although it may work on other models. Although the System Management Mode method will turn the fan on/off on these models, it causes the processor to hang about two seconds later. Therefore it is necessary to use the following method on these models. The Tecra 700x models have a MachineID of 0xFCCC.
The following pseudo code will turn the fan on:

```
DISABLE INTERRUPTS
WRITE 0xE0 TO PORT 0xE4
READ byte FROM PORT 0xE5
SET BIT 0 OF byte                    (i.e. least significant bit)
SET HIGH BYTE OF WORD TO byte
SET LOW BYTE OF WORD TO 0xE0
WRITE word TO PORT 0xE4
ENABLE INTERRUPTS
```

The following pseudo code will turn the fan off:

```
DISABLE INTERRUPTS
WRITE 0xE0 TO PORT 0xE4
READ byte FROM PORT 0xE5
CLEAR BIT 0 OF byte
SET HIGH BYTE OF WORD TO byte
SET LOW BYTE OF WORD TO 0xE0
WRITE word TO PORT 0xE4
ENABLE INTERRUPTS
```

The following pseudo code will get the current status of the fan:

```
DISABLE INTERRUPTS
WRITE 0xE0 TO PORT 0xE4
READ byte FROM PORT 0xE5
ENABLE INTERRUPTS
IF BIT 0 OF byte SET THEN ON ELSE OFF
```

Hotkey

The Fn key on Toshiba laptops in addition to it's usual function on a laptop of providing for the numeric keypad, also provides a number of other services. In particular the combinations Fn+F1 through to Fn+F5 do useful things such as change boot mode, set the alarm volume, switch between the LCD and external monitor etc. While all these work under Linux, under MS-DOS or Microsoft Windows the Fn+F2 and Fn+F3 combinations provide visual feedback to what the current setting is and what you have changed it to. Toshiba have also provided a program called Fnesse which enables you to create short cuts to running a program with the Fn key.
The following method works on a wide range of Toshiba laptops from a T1900 all the way through to a Tecra 740. The pseudo code shows how to get back the status of the Fn key.

```
DISABLE INTERRUPTS
WRITE 0x8E TO PORT 0xE4
READ byte FROM PORT 0xE5
ENABLE INTERRUPTS
```

The byte held in the variable byte indicates the status of the Fn key. If the Fn key has not been depressed in combination with another key (or any of the keys making up the numeric keypad) then byte will hold 0x00. If the Fn key has been depressed in combination with another key and the other key is *still* held down then byte will hold the BIOS scan code of that key. If the key has been released but the Fn key is still held down then byte will hold the BIOS scan code of the key with bit 7 *set*.
On newer models the status of the Fn Key can be obtained by directly reading from a specific port. The port changes from model to model, but so far only two ports have been identified as holding the information. These ports are 0x62 and 0x68. The format of the byte returned is identical to the previous method. The following table show which port to use for which machine ID.

Port	Machine ID
0x62	0xfc02,0xfc04,0xfc09,0xfc0a,0xfc10,0xfc11,0xfc13,0xfc15,0xfc1a
0x68	0xfc08,0xfc17,0xfc1d,0xfcd1,0xfce0,0xfce2

It should be noted that the Fn key combinations that are used to emulate keys on a standard keyboard cannot be detected as Fn combinations.