STANDARD
MICROSYSTEMS
CORPORATION

APPLICATION NOTE 8.4

Device Bay Primer and DBC98C51 Test Program Guide

By Randy Goldberg

Introduction

The purpose of this application note is to cover the following topics:

1. Overview of the Device Bay concept
2. How SMSC's DBC98C51 fits into the Device Bay concept
3. More detailed description of the DBC98C51
4. A description of SMSC's DBC98C51 demo card
5. "User's" guide for SMSC's in house DBC98C51 test program
    (a) Overview
    (b) Description of "DEVBAY" test
6. How to observe DBC98C51 functionality using manual inputs
7. SMB bus interface of the DBC98C51
8. Source code key for "DEVBAY test"

NOTE: The software referenced in this application note is available on SMSC's FTP server at 1-516-233-4272 or via the World Wide Web at http://www.smsc.com/ftpdocs/chips.html. Download the files called devtext.exe, fra24.c, and frb24.c in the "appnote" directory.

Device Bay Concept

Device Bay is a specification that was developed by Compaq, Intel and Microsoft. The concept is to have a "bay" in which various devices can be inserted and removed while the system is still powered up. Typical devices are mass storage devices, communications devices, and security devices, although devices are not limited to these types.

Conceptually the Device Bay interface bus has three types of pins: Power, USB and 1394 pins, and Bay management pins.

The main communication link for a Device Bay device, is either Universal serial bus (USB) or IEEE 1394 (Firewire). The host system will typically contain USB/PCI and 1394/PCI bridges to effect the high-speed communication with the Device Bay device. Pins exist on the specified Device Bay connector for the USB and Firewire link.

Other pins on the Connector deal with the actual control and status of the device that is inserted into the bay (bay management pins). An incomplete list of these includes 1394 device present, removal request, and power enable. These pins get connected to the Device Bay controller (DBC). The DBC resides on the host side of the bay (although physically, the DBC may not be attached to the host). The DBC helps with device insertion, device removal requests, and other events.

80 Arkay Drive
Hauppauge, NY 11788
(516) 435-6000
FAX (516) 273-3123

The DBC in its simplest form, is a device with registers that are controlled by the host through some type of bus, and some inputs and outputs that that go to the Device Bay connector. Various states of a state machine are cycled through based on the pin inputs, and register activity. The state machine will be described in greater depth in a later section.

A typical Present day Device Bay system will have the bays in the PC chassis. Future Device Bay systems may have the bays in a non-traditional spot, such as the monitor. This is why there must be flexibility in the way the Device Bay controller is implemented. In a system that has the bays directly in the chassis it may be best to have an SMB Device Bay controller, as most chip sets already have an SMB controller. In a remote Device Bay system, such as Device Bays residing in a monitor, it would be best to interface the Device Bay controller to the host through USB, which is an external bus.

For more information on Device Bay, please see the Device Bay Interface specification, which can be found at, www.device-bay.org

SMSC's DBC98C51 in the Device Bay Concept

SMSC's DBC98C51 is an SMB Device Bay controller. It contains a slave SMB device, which interfaces to a host. It is ACPI compliant, supports two Device Bays, can run either off of 5 or 3.3 volts, and comes in a 28 pin SOIC package. It is compliant with Revision 0.85 of the Device Bay Interface Specification. Its main application would be in a Device Bay environment where the bays are in the chassis. The DBC98C51 would interface to the host through either the embedded SMB bus controller within the chip set, or a discrete controller such as the Phillips 8584 IIC host controller chip.

The State machine of the DBC98C51 is implemented in hardware.

More detailed DBC98C51 description

A conceptually simplified, and not totally complete pin out of the DBC98C51 is:

PIN	TYPE	DESCRIPTION
1394 device present	Input	From the Device Bay connector, indicates if an actual 1394 device is inserted in the Bay
USB device present	Input	From the Device Bay connector, indicates if an actual USB device is inserted in the Bay
Device Removal request	Input	From the Device Bay connector, indicates if the user, through the push of a button, is requesting that device removal be allowed
Security lock status	Input	From the Device Bay connector, an indication of the state of the mechanical interlock
Power enable output	Output	To the Device Bay connector, enables power to the device inserted in the bay
Software controlled interlock	Output	To the Device Bay connector, actually controls a mechanical interlock on the Bay so that the user can not remove the device at will
Green and Yellow LED outputs	Outputs	These pins control LEDS. Different states of the state machine will be reflected by various LED states
SCLK, SDATA	IO	SMB bus interface

Note: Many of the above pins occur twice; once for each Device Bay supported.

The state machine will determine its state based on the 1394 present input, USB present input, Device removal request input, and register settings as written by the host. Different states will flash the LEDS differently.

Here is a list of the states and their associated LED outputs:

STATE	STATE NUMBER	LEDS	
		YELLOW	GREEN
Bay empty	000	OFF	OFF
Device inserted	001	OFF	FLASH
Device enabled	010	OFF	ON
Removal requested	011	FLASH	OFF
Device removal allowed	100	OFF	OFF

Please refer to the DBC98C51 specification (available on the SMSC web site) for a complete list of states and events that transition the states.

In the specification, you will see two state machines, one is based on hardware events (input pin changes), and the other is based on software transitions (register changes made by the host). As expected, some transition can only be made in hardware or software. For example, the transition from the bay empty state, to the Device inserted state, can only be made through hardware, when an actual device is placed in the bay. On the other hand, the transition from the removal requested state to the Device removal allowed state, is only allowed through software (this is so that the OS has a chance to properly shut down the device before removal).

SMSC's DBC98C51 Demo card

SMSC needed to develop a demo card to do device evaluation of the DBC98C51. The card took the form of an ISA plug in board that has a discrete Phillips 8584 ISA to SMB host controller. The 8584 is connected to two slave devices on the board, the SMSC DBC98C51 and a 24C02 (an IIC EEPROM). The EEPROM is optional, and all boards may not have the EEPROM footprint populated.

The Device Bay inputs and outputs can be stimulated (inputs) and monitored (outputs), in two different ways. The board can be jumpered in the following two ways:

Automated mode

In this mode, DBC98C51 Device Bay inputs are driven by an on board 74LS245, which is an ISA mapped output register in the ISA prototype range. The DBC98C51 Device Bay outputs (including the interrupt output) are connected to a different 74LS245 input register. In this mode, the DBC98C51 is under complete software control (ISA). Device Bay registers are read/written, by communicating to the Phillips 8584, which will then talk to the DBC98C51 over the SMB bus. The DBC98C51 inputs and outputs are also accessible through the registers on the ISA bus. The automated mode is used for our in-house test, which will be described in detail in a later section.

To jumper this board for "Automated mode", populate the JP16 jumper strip, and remove all jumpers from the JP2 jumper strip.

Manual mode

In this mode, DBC98C51 Device Bay inputs are driven by a variety of different switches, and DBC98C51 outputs drive various LEDS. As in the automated mode, Device Bay registers are read/written, by communicating to the Phillips 8584, which will then talk to DBC98C51 over the SMB bus. In this mode, the users can transition through various states of the state machine by playing with the various switches.

To jumper this board for "Manual mode", populate the JP2 jumper strip, and remove all jumpers from the JP16 jumper strip.

There are two important addresses that need to be set through slide switches on the board. The first is the DBC98C51 SMB slave address. Each device on an SMB bus must have an address, and for the DBC98C51 on this address is determined by the state of two inputs (SMB_A0, SMB_A1). These inputs are controlled by two slide switches (S5). The second important address to set up on the board, is the ISA address of the Phillips 8584, and the discrete 74LS245 registers. One PAL does the address decoding for both of them. The address space must go on 256 byte boundaries, i.e. 0x100, 0x200, 0x300, etc. S4 controls the ISA base address. The board comes shipped with the ISA address set to 0x200. 0x200 or 0x300 seem to have the least amount of contention in most ISA systems. The board occupies (5) consecutive ISA addresses.

A DBC98C51 evaluation kit can be obtained by contacting your local regional sales office.

User's guide for SMSC's in house DBC98C51 test program

Overview

The in-house program is called "DEVBAY". It was developed exclusively for in house debug of the part, but it will be made available to our customers. Due to the fact that the program was developed for engineering use, the value of this program may not be readily apparent to the user. However there are two significant uses of this program by the user.

- Can use this program as an automated test for the board to confirm that there are no ISA conflicts, and that the board is working properly, in their particular ISA environments. Once this is done, the user can move on the "manual" mode of board operation, which may be more interesting to the user.

- Many parts of this test just involve pressing a button, and observing the screen say "pass". An interested user can review the source code of this program to observe what the test was doing and how the part works.

DEVBAY test

The test is menu driven, but before the menus can be accessed the program will run an inventory of the system Device Bay resources. Here is the menu structure, followed by a description of each item, including the initial system inventory check:

Main menu:

1-Device Bay register R/W
2-Device Bay tests
    1-Write register test
    2-Read register test
    3-Write/read register test
    4-Check Device Bay inputs
    5-Check Device Bay outputs
    6-Check state machine

Inventory

The first thing the program does when it is run is to do an "inventory" of devices it expects to see on the Device Bay demo board. It first looks for the Phillips 8584 connected to the ISA bus. After the host finds the 8584, it can begin looking for the Device Bay part, it does this by programming the 8584 to put out "start" commands with the appropriate Device Bay device code, on the SMB bus. The program cycles through all available 8 Device Bay address, until it "finds" the DBC98C51 on the board. It then repeats the same process to find the 24C02 EEPROM on the board.

At the conclusion of the “inventory”, the results should be that an 8584 was found (at the address that the board was jumpered for), a DBC98C51 was found (at the address that the board was jumpered for), and that a EEPROM was or wasn’t found (the board may not have the EEPROM populated, so it may not be found). If you don’t get the above results, try to find out what the system conflict is before proceeding.

Device Bay register R/W

This test option lets you Write/Read some of the registers that are accessible on the Device Bay demo board. The editor contains three pages. The first page are the registers associated with the DBC98C51 Device Bay controller, the second page are the registers associated with the 24C02 IIC EEPROM, and the last page are the test registers on the board that are used to control and monitor the DBC98C51 Device Bay inputs and outputs. Registers can be modified by typing new data into the register location, and then hitting “F5” to update the register value. All of the DBC98C51 Device Bay and 24C02 EEPROM register are accessed through the Phillips 8584 that is on the board. The test registers are directly ISA mapped. This register editor will also be useful in the “manual” test mode, to be described in a later section.

Device Bay tests: Write register test

This test option writes the DBC98C51 registers with a known pattern. Since this test is a write test, the only way for it to fail is if there is a problem with the SMB 8584 to DBC98C51 interface.

Device Bay tests: Read register test

This test option reads the DBC98C51 registers. It compares the read data with the data that was written in the above write test. Any mismatches will be flagged by this test. Only registers, which are writeable will be checked for proper read data.

Device Bay tests: Write/Read register test

Combination of above read and write tests.

Device Bay tests: Check Device Bay inputs

This test uses the discrete on board registers to stimulate the DBC98C51 inputs to verify that they work correctly. For example, the 1394_present pin, will be driven active and inactive and the Device Bay status register will be read to verify that it properly reflects the pin.

Device Bay tests: Check Device Bay outputs

This test uses the discrete on board registers to monitor the DBC98C51 outputs to verify that they work correctly. For example, the Power_enable pin will be monitored, when the control register is being written to enable and disable the power.

Device Bay tests: Check State machine

This test will stimulate the inputs, and monitor the outputs of the DBC98C51. The test will go through all the states of the state machine with a combination of hardware transitions and software transitions. Outputs (including the LED outputs) will be monitored to verify proper state sequencing.

(Note: For the automated test to work, the board needs to be jumpered for automated mode, as described in the section “SMSC’s DBC98C51 Demo card”)

Observing DBC98C51 functionality in manual mode

One way that the SMSC DBC98C51 demo card can show Device Bay controller functionality, is by using the board in manual mode. Manual mode means that Device Bay inputs on the DBC98C51 come from switches, and Device Bay outputs drive LED's. Manual mode is described in more detail in the previous section, "SMSC's DBC98C51 Demo card".

In the following example, the 1394_present pin and the removal request pin of BAY0, along with various DBC98C51 register writes will be used to cycle through all DBC98C51 states.

1. Make sure all dip switches of S2 are in the off position.
2. Run DEVTEST, and select option 1, which is the DBC98C51 register editor.
3. Observe that both the GREEN LED and the YELLOW LED are off. This is the Bay empty state.
4. Write the BAY0 control register (0x14), with 0x0c. This will enable device status changes to generate an interrupt.
5. Move switch 1, of SW2 to the 'on' position. This is the 1394_present pin input to the DBC98C51. A 1394 device is now "present". Observe that the GREEN LED has started to flash. The DBC98C51 is now in state 1(device inserted).
6. To enter state 2(device enabled), a software transition is needed. Write the BAY0 control register (0x14), with 0x2c. The GREEN LED should now always be on. The DBC98C51 is now in state 2(device enabled).
7. To enter state 3(removal requested), a hardware transition can be used. Press the momentary switch S3. This is the removal request input. The GREEN LED should now be off, and the YELLOW LED should be flashing. The DBC98C51 is now in state 3(removal requested).
8. To enter state 4(device removal allowed), a software transition is needed. Write the BAY0 control register (0x14), with 0x4c. Both LEDs should now be off. The DBC98C51 is now in state 4(device removal allowed).

SMB bus interface of the DBC98C51

SMB bus is very similar to the IIC bus. SMB is a two wire, open drain, low speed, multi node bus. A device on the SMB bus is either a host or a slave (most hosts can be a slave, but slave are always slaves only). The DBC98C51 is a slave device, that gets data transferred to/from it by a master device. The choice of the SMB interface for the host interface for the DBC98C51 is an ideal one. The bay management function of the Device Bay concept, is intrinsically a slow one, as it deals with a human interface (put a device in the bay, press a device removal request button….). SMB is slow, with the benefit of minimal pin requirements. Actual high-speed data transfer to the device of the bay has nothing to do with the DBC98C51.

Please see SMSC Application note 7.3, "Access bus primer", for a more thorough description of the IIC bus, which is very similar to the SMB bus.

Source code key

If the user cares to examine the source code for "devtest" they can observe exactly how the DBC98C51 is being tested. However, as it was written as an engineering debug program, it will be somewhat difficult to follow. In this section, I will try to describe some of the procedures used in the program. The full name of the procedure is given so that the reader can search the code for these text strings. By describing some of the important procedures, the reader can get a feel for how the program was written.

All of the procedures described below are in module "FRA24.C".

The following 5 procedures, form the core of the test program, as they perform the low level control of the DBC98C51 (writing/reading its registers, and driving/monitoring its inputs and outputs)

void wriic(int offset,int data), int rdiic2(int offset)
If the host wants to write or read a register within the DBC98C51, it must do an IIC read/write thorough the Phillips 8584. These two procedures do the IIC transfer.

void drivedbin(int,int);
The host has the ability to drive DBC98C51 input pins by writing to on onboard 74LS245. This procedure writes a specific Device Bay pin either high or low.

int chkdbout(int,int);
The host has the ability to check DBC98C51 output pins by reading an onboard 74LS245. This procedure checks that a specific Device Bay output pin is either high or low.

int checkstate0(int);
This procedure verifies that the DBC98C51 is in state 0. It does this by verifying that the green and yellow LED's are in the correct state, and that the Device Bay status register is reflecting the state correctly. Similar procedures exist for verifying states 1-4.

The following 6 procedures, are the actual tests that get done on the DBC98C51, as described in an earlier section (User's guide for SMSC's in house DBC98C51 test program:DEVBAY test). They will not be described again here, but are included so that the user knows how to search for them in the source code.

UI write_reg(void);	Write register test
UI read_reg(void);	Read register test
UI rw_reg(void);	Write/read test
UI chkgpio(void);	Check Device Bay input test
UI chkout(void);	Check Device Bay output test
UI chkstate(void);	Check state machine test