Open Firmware

Recommended Practice:

SCSI-3 Parallel Interface

Version 1

December 23, 1996 2:49 pm

Published by the Open Firmware Working Group

This document is a voluntary-use recommended practice of the Open Firmware Working Group. The Open Firmware Working Group is an ad hoc committee composed of individuals interested in Open Firmware as defined by IEEE 1275-1994, related standards, and their application to various computer systems.

The Open Firmware Working Group is involved both in IEEE sanctioned standards activities, whose final results are published by IEEE, and in informal recommendations such as this, which are published on the Internet at:

```
http://playground.sun.com/pub/1275
```

Membership in the Open Firmware Working Group is open to all interested parties. The working group meets at regular intervals at various locations. For more information send email to:

```
p1275-wg@risc.sps.mot.com
```

Revision History

1. Introduction

1.1. Overview and References

This draft describes the application of Open Firmware to the SCSI-3 protocol as implemented on the SCSI-3 Parallel Interface, and addresses nodes representing SPI host adapters.

2. References and Definitions

2.1. Definitions of Terms

bus node: A device node that represents the interface, or "host adapter," between an SPI bus and its parent (which may be another bus).

child node: A device node that represents an SPI "target" device.

2.2. References

[1] IEEE Std 1275-1994 IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices
[2] Device Support Extensions to: IEEE Std 1275-1994... Revision 0.6 DRAFT
[3] Generic Names...Version 1.3 Uapproved Draft
[4] ANS X3T10/994D SCSI-3 Architecture Model (SAM)
[5] ANS X3T10/855D SCSI-3 Parallel Interface (SPI)
[6] ANS X3T10/865D SCSI-3 Interlocked Protocol (SIP)

3. Bus Characteristics

SPI is not a memory-mapped bus. Operations on target devices are performed by executing a transaction consisting of multiple phases, including selecting a particular target device, sending a multibyte command to the target, possibly transferring multiple data bytes to or from the target, and returning status.

3.1. Physical Address Formats and Representations

SPI devices are addressed with one 5-bit "target" number and one 6-bit "logical unit" number.

The text representation of an SPI physical address is *target*,*lun*. Each of *target* and *lun* is a string of characters composed of lower case hexadecimal digits, with leading zeros deleted.

The numerical representation of an SPI bus physical address consists of the target number in the high number and the logical unit number in the low number.

3.2. Bus-specific Configuration Variables

None.

3.3. Format of a Probe List

None.

3.4. Interrupt Specification Format

None (SPI has no interrupts).

3.5. FCode Interpretation Semantics

None (SPI has no provision for device identification via FCode).

4. Bus Nodes

4.1. Properties

4.1.1 Open Firmware-defined Properties for Bus Nodes

The following standard property, as defined in Open Firmware, has special meaning or interpretation for SPI:

`"device_type"` S

Standard prop-name to specify the implemented interface.

The meaning of this property is as defined in Open Firmware. A package conforming to this specification and corresponding to a device that implements an SPI bus shall implement this property with the string value "spi".

4.1.2 Bus-specific Properties for Bus Nodes

`"differential"` S

prop-name, indicates that the SPI node supports differential signaling.

prop-encode-array: <none>

This property shall be present if the SPI controller represented by his node supports differential signaling.

`"scsi-initiator-id"` S

prop-name, indicating the initiator-id to be used for SCSI transfers by this controller.

prop-encode-array: An integer, encoded as with encode-int, that specifies the value of the initiator-id for subsequent transfers by this controller. The value is in the range of 0..31. The initial value of the property is implementation dependent.

4.2. Methods

4.2.1 Open-Firmware-defined Methods for Bus Nodes

A package implementing the "spi" device type shall implement the following standard methods as defined in Open Firmware, with physical address representations as specified in 3.1:

`open` (-- okay?) M

Prepare this device for subsequent use.

In addition to the standard Open Firmware behavior, the open method shall set the host adapter's own selection ID as follows:

Attempt to locate a "scsi-initiator-id" property by executing "get-inherited-property" with the string "scsi-initiator-id" as its argument. If such a property is found, decode its value as with "decode-int", and use the decoded value as the host adapter's own selection ID. Otherwise, use the value 7.

Note: The use of "get-inherited-property" to get the "scsi-initiator-id" property makes it possible to choose the set of SCSI host adapters to which the property applies. If the property is in the root node, for example, it applies to all the SCSI host adapters in the system. If the property is elsewhere in the device tree, it applies only to host adapters in the subtree below and including the location of the property.

`close` (--) M

Close this previously opened device.

`dma-alloc` (...size -- virt) M

Allocate a memory region for later use.

`dma-free` (virt size --) M

Free memory allocated with `dma-alloc`.

`decode-unit` (addr len -- lun target) M

Convert text unit-string to physical address.

`encode-unit` (lun target -- addr len) M

Convert physical address to text unit-string.

4.2.2 Bus-specific Methods for Bus Nodes

A package implementing the "spi" device type shall implement the following bus-specific methods:

`max-transfer` (-- n) M

Returns the maximum DMA transfer length supported by the hardware.

`set-address` (unit# target# --) M

Sets the SCSI target number (0x0..0x1f) and logical unit number (0x0..0x3f) to which subsequent commands apply.

`set-timeout` (msecs --) M

Sets the maximum length of time in milliseconds that the driver will wait for the completion of a command. The default value of zero means to wait indefinitely. A hardware error result is reported for a command that times out.

`show-children` (--) M

Searches the SPI bus for attached target devices and their associated logical units. Displays the information that the SCSI "inquiry" command reports for those devices.

`execute-command` (buf-addr buf-len dir cmd-addr cmd-len -- hw-err? | statbyte 0) M

Executes the SCSI command, which is stored in memory at *cmd-addr* and whose length is *cmd-len*. *Dir* is true if the data transfer phase of the SCSI command will transfer data from the device to memory, and false otherwise. *Buf-addr* is the address of the memory buffer to be used for the data transfer phase, and *buf-len* is the expected maximum length of the data transfer phase. The memory buffer must be contained within a DMA-accessible region that was returned by a previous execution of `dma-alloc`. If *buf-len* is zero, indicating that the command is not expected to have a data transfer phase, both *buf-addr* and *dir* are ignored. *Hw-err?*, the returned hardware error status, is nonzero if the command could not be executed at all (perhaps due to the device not responding to selection attempt). If *hw-err?* is zero, *statbyte* is the status byte returned by the status phase of the command.

`retry-command` (buf-addr buf-len dir cmd-addr cmd-len #retries -- 0 | hw-err? stat | sensebuf 0 stat) M

Executes a SCSI command, automatically retrying under certain conditions. `Retry-command` is similar to `execute-command` except that `retry-command` automatically retries under certain failure conditions and automatically executes the "request sense" SCSI command as necessary. *#retries* is the maximum number of times that the command will be retried; if *#retries* is -1, the command will be retried indefinitely. `Retry-`

`command` returns 0 if the command eventually succeeds. Otherwise, it returns the status byte returned by the last attempted command on top of the stack (-1 if the command failed due to a hardware error). The second number on the stack (*hw-err?*) indicates whether or not the extended sense information is available. If *hw-err?* is zero, the third number on the stack (*sensebuf*) is the address of a memory buffer containing the extended sense information returned by the "request sense" command that was executed after the last attempt to execute the desired command. The criteria for whether or not to retry the command are as follows:

a) If the requested number of retries have already been performed, do not retry.
b) If the failure is due to a hardware error, do not retry.
c) If the failure was due to a "device busy" condition reported in the status byte, retry.
d) Otherwise, execute the "get extended status" command and attempt to determine whether or not the failure could be retried based on the data in the returned sense buffer, as follows:

1) Unknown error class (not 7) is not retryable.
2) Filemark is not retryable.
3) End of media is not retryable.
4) Illegal length indicator is not retryable.
5) Sense key = No Sense is retryable.
6) Sense key = Recoverable error is retryable.
7) Sense key = Not Ready is retryable.
8) Sense key = Unit Attention is retryable.
9) Transaction aborted due to Incoming SCSI Bus reset is retryable.
10) Otherwise, the error is not retryable.

`no-data-command` (cmd-addr -- error?) M

Executes a simple SCSI command, automatically retrying under certain conditions. *Cmd-addr* is the address of a 6-byte command buffer containing a SCSI command that does not have a data transfer phase. Executes the command, retrying indefinitely with the same retry criteria as `retry-command`. *Error?* is nonzero if an error occurred, zero otherwise.
NOTE -- `no-data-command` is a convenience function. It provides no capabilities that are not present in `retry-command`, but for those commands that meet its restrictions, it is easier to use.

`short-data-command` (data-len cmd-addr cmd-len -- error? | data-addr 0) M

Executes a simple SCSI command, automatically retrying under certain conditions. *Cmd-addr* is the address and *cmd-len* is the length of a command buffer containing a SCSI command whose data transfer phase is expected to transfer less than 256 bytes in an incoming direction. *data-len* is the expected length (1..255) of the data transfer. Executes the command, retrying indefinitely with the same retry criteria as `retry-command`. *Error?* is nonzero if an error occurred, zero otherwise. If *error?* is zero, *data-addr* is the address of a buffer containing the data transferred by the execution of the command.
NOTE -- `short-data-command` is a convenience function, eliminating the need for allocating a DMA buffer. It is primarily intended for use with "informational" SCSI commands like "read block limits" and "inquiry".

`diagnose` (-- error-code | 0) M

Performs a simple self-test for a generic SPI device. Perform a SCSI "test-unit-ready" command on the currently selected target and logical unit (see `set-address`). If that fails, display a message indicating the details of the failure and return a nonzero error code. Otherwise, perform a SCSI "send-diagnostic" command, returning zero if it succeeds or a nonzero error code if it fails.

5. Child Nodes

5.1. Properties

5.1.1 Open Firmware-defined Properties for Child Nodes

None.

5.1.2 Bus-specific Properties for Child Nodes

None.

5.2. Methods

5.2.1 Open Firmware-defined Methods for Child Nodes

None.

5.2.2 Bus-specific Methods for Child Nodes

None.

6. User Interface Extensions

None.