

SuperH Interfaces Guide

Paul Mundt

`lethal@linux-sh.org`

SuperH Interfaces Guide

by Paul Mundt

Copyright © 2008 Paul Mundt

Copyright © 2008 Renesas Technology Corp.

This documentation is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For more details see the file COPYING in the source distribution of Linux.

Table of Contents

1. Memory Management	1
1.1. SH-4	1
1.1.1. Store Queue API	1
sq_flush_range	1
sq_remap	2
sq_unmap	3
1.2. SH-5	3
1.2.1. TLB Interfaces	4
sh64_tlb_init	4
sh64_next_free_dtlb_entry	4
sh64_get_wired_dtlb_entry	5
sh64_put_wired_dtlb_entry	6
sh64_setup_tlb_slot	6
sh64_tearardown_tlb_slot	7
for_each_dtlb_entry	8
for_each_itlb_entry	9
__flush_tlb_slot	10
2. Clock Framework Extensions	11
clk_set_rate_ex	11
3. Machine Specific Interfaces	13
3.1. mach-dreamcast	13
aica_rtc_gettimeofday	13
aica_rtc_settimeofday	13
3.2. mach-x3proto	14
ilsel_enable	14
ilsel_enable_fixed	15
ilsel_disable	16
4. Busses	19
4.1. SuperHyway	19
superhyway_add_device	19
superhyway_register_driver	20
superhyway_unregister_driver	21
4.2. Maple	21
maple_driver_register	22
maple_driver_unregister	22
maple_getcond_callback	23
maple_add_packet	24
maple_add_packet_sleeps	25

Chapter 1. Memory Management

1.1. SH-4

1.1.1. Store Queue API

sq_flush_range

LINUX

Kernel Hackers Manual May 2009

Name

`sq_flush_range` — Flush (prefetch) a specific SQ range

Synopsis

```
void sq_flush_range (unsigned long start, unsigned int len);
```

Arguments

start

the store queue address to start flushing from

len

the length to flush

Description

Flushes the store queue cache from *start* to *start + len* in a linear fashion.

sq_remap

LINUX

Kernel Hackers Manual May 2009

Name

`sq_remap` — Map a physical address through the Store Queues

Synopsis

```
unsigned long sq_remap (unsigned long phys, unsigned int size,  
const char * name, unsigned long flags);
```

Arguments

phys

Physical address of mapping.

size

Length of mapping.

name

User invoking mapping.

flags

Protection flags.

Description

Remaps the physical address *phys* through the next available store queue address of *size* length. *name* is logged at boot time as well as through the sysfs interface.

sq_unmap

LINUX

Kernel Hackers ManualMay 2009

Name

sq_unmap — Unmap a Store Queue allocation

Synopsis

```
void sq_unmap (unsigned long vaddr);
```

Arguments

vaddr

Pre-allocated Store Queue mapping.

Description

Unmaps the store queue allocation *map* that was previously created by `sq_remap`. Also frees up the pte that was previously inserted into the kernel page table and discards the UTLB translation.

1.2. SH-5

1.2.1. TLB Interfaces

sh64_tlb_init

LINUX

Kernel Hackers Manual May 2009

Name

`sh64_tlb_init` — Perform initial setup for the DTLB and ITLB.

Synopsis

```
int sh64_tlb_init ( void );
```

Arguments

void

no arguments

sh64_next_free_dtlb_entry

LINUX

Kernel Hackers ManualMay 2009

Name

`sh64_next_free_dtlb_entry` — Find the next available DTLB entry

Synopsis

```
unsigned long long sh64_next_free_dtlb_entry ( void );
```

Arguments

void

no arguments

sh64_get_wired_dtlb_entry

LINUX

Kernel Hackers ManualMay 2009

Name

`sh64_get_wired_dtlb_entry` — Allocate a wired (locked-in) entry in the DTLB

Synopsis

```
unsigned long long sh64_get_wired_dtlb_entry ( void );
```

Arguments

void

no arguments

sh64_put_wired_dtlb_entry

LINUX

Kernel Hackers ManualMay 2009

Name

`sh64_put_wired_dtlb_entry` — Free a wired (locked-in) entry in the DTLB.

Synopsis

```
int sh64_put_wired_dtlb_entry (unsigned long long entry);
```

Arguments

entry

Address of TLB slot.

Description

Works like a stack, last one to allocate must be first one to free.

sh64_setup_tlb_slot

LINUX

Kernel Hackers Manual May 2009

Name

`sh64_setup_tlb_slot` — Load up a translation in a wired slot.

Synopsis

```
void sh64_setup_tlb_slot (unsigned long long config_addr,  
unsigned long eaddr, unsigned long asid, unsigned long paddr);
```

Arguments

config_addr

Address of TLB slot.

eaddr

Virtual address.

asid

Address Space Identifier.

paddr

Physical address.

Description

Load up a virtual<->physical translation for *eaddr*<->*paddr* in the pre-allocated TLB slot *config_addr* (see `sh64_get_wired_dtlb_entry`).

sh64_teardown_tlb_slot

LINUX

Kernel Hackers ManualMay 2009

Name

`sh64_teardown_tlb_slot` — Teardown a translation.

Synopsis

```
void sh64_teardown_tlb_slot (unsigned long long config_addr);
```

Arguments

config_addr

Address of TLB slot.

Description

Teardown any existing mapping in the TLB slot *config_addr*.

for_each_dtlb_entry

LINUX

Kernel Hackers ManualMay 2009

Name

`for_each_dtlb_entry` — Iterate over free (non-wired) DTLB entries

Synopsis

```
for_each_dtlb_entry ( tlb );
```

Arguments

tlb

TLB entry

for_each_itlb_entry

LINUX

Kernel Hackers Manual May 2009

Name

`for_each_itlb_entry` — Iterate over free (non-wired) ITLB entries

Synopsis

```
for_each_itlb_entry ( tlb );
```

Arguments

tlb

TLB entry

__flush_tlb_slot

LINUX

Kernel Hackers Manual May 2009

Name

`__flush_tlb_slot` — Flushes TLB slot *slot*.

Synopsis

```
void __flush_tlb_slot (unsigned long long slot);
```

Arguments

slot

Address of TLB slot.

Chapter 2. Clock Framework Extensions

clk_set_rate_ex

LINUX

Kernel Hackers Manual May 2009

Name

`clk_set_rate_ex` — set the clock rate for a clock source, with additional parameter

Synopsis

```
int clk_set_rate_ex (struct clk * clk, unsigned long rate, int algo_id);
```

Arguments

clk

clock source

rate

desired clock rate in Hz

algo_id

algorithm id to be passed down to `ops->set_rate`

Description

Returns success (0) or negative errno.

Chapter 3. Machine Specific Interfaces

3.1. mach-dreamcast

aica_rtc_gettimeofday

LINUX

Kernel Hackers Manual May 2009

Name

`aica_rtc_gettimeofday` — Get the time from the AICA RTC

Synopsis

```
void aica_rtc_gettimeofday (struct timespec * ts);
```

Arguments

ts

pointer to resulting timespec

Description

Grabs the current RTC seconds counter and adjusts it to the Unix Epoch.

aica_rtc_settimeofday

LINUX

Kernel Hackers Manual May 2009

Name

`aica_rtc_settimeofday` — Set the AICA RTC to the current time

Synopsis

```
int aica_rtc_settimeofday (const time_t secs);
```

Arguments

secs

contains the `time_t` to set

Description

Adjusts the given *tv* to the AICA Epoch and sets the RTC seconds counter.

3.2. mach-x3proto

ilsele_enable

LINUX

Name

`ilssel_enable` — Enable an ILSEL set.

Synopsis

```
int ilssel_enable (ilssel_source_t set);
```

Arguments

set

ILSEL source (see `ilssel_source_t` enum in `include/asm-sh/ilssel.h`).

Description

Enables a given non-aliased ILSEL source (\leq ILSEL_KEY) at the highest available interrupt level. Callers should take care to order callsites noting descending interrupt levels. Aliasing FPGA and external board IRQs need to use `ilssel_enable_fixed`.

The return value is an IRQ number that can later be taken down with `ilssel_disable`.

`ilssel_enable_fixed`

LINUX

Name

`ilssel_enable_fixed` — Enable an ILSEL set at a fixed interrupt level

Synopsis

```
int ilssel_enable_fixed (ilssel_source_t set, unsigned int
level);
```

Arguments

set

ILSEL source (see `ilssel_source_t` enum in `include/asm-sh/ilssel.h`).

level

Interrupt level (1 - 15)

Description

Enables a given ILSEL source at a fixed interrupt level. Necessary both for level reservation as well as for aliased sources that only exist on special ILSEL#s.

Returns an IRQ number (as `ilssel_enable`).

`ilssel_disable`

LINUX

Name

`ilssel_disable` — Disable an ILSEL set

Synopsis

```
void ilssel_disable (unsigned int irq);
```

Arguments

irq

Bit position for ILSEL set value (retval from enable routines)

Description

Disable a previously enabled ILSEL set.

Chapter 4. Busses

4.1. SuperHyway

superhyway_add_device

LINUX

Kernel Hackers Manual May 2009

Name

superhyway_add_device — Add a SuperHyway module

Synopsis

```
int superhyway_add_device (unsigned long base, struct  
superhyway_device * sdev, struct superhyway_bus * bus);
```

Arguments

base

Physical address where module is mapped.

sdev

SuperHyway device to add, or NULL to allocate a new one.

bus

Bus where SuperHyway module resides.

Description

This is responsible for adding a new SuperHyway module. This sets up a new struct `superhyway_device` for the module being added if `sdev == NULL`.

Devices are initially added in the order that they are scanned (from the top-down of the memory map), and are assigned an ID based on the order that they are added. Any manual addition of a module will thus get the ID after the devices already discovered regardless of where it resides in memory.

Further work can and should be done in `superhyway_scan_bus`, to be sure that any new modules are properly discovered and subsequently registered.

superhyway_register_driver

LINUX

Kernel Hackers Manual May 2009

Name

`superhyway_register_driver` — Register a new SuperHyway driver

Synopsis

```
int superhyway_register_driver (struct superhyway_driver *  
drv);
```

Arguments

drv

SuperHyway driver to register.

Description

This registers the passed in *drv*. Any devices matching the id table will automatically be populated and handed off to the driver's specified probe routine.

superhyway_unregister_driver

LINUX

Kernel Hackers Manual May 2009

Name

`superhyway_unregister_driver` — Unregister a SuperHyway driver

Synopsis

```
void superhyway_unregister_driver (struct superhyway_driver *  
drv);
```

Arguments

drv

SuperHyway driver to unregister.

Description

This cleans up after `superhyway_register_driver`, and should be invoked in the exit path of any module drivers.

4.2. Maple

maple_driver_register

LINUX

Kernel Hackers Manual May 2009

Name

`maple_driver_register` — register a maple driver

Synopsis

```
int maple_driver_register (struct maple_driver * drv);
```

Arguments

drv

maple driver to be registered.

Description

Registers the passed in *drv*, while updating the bus type. Devices with matching function IDs will be automatically probed.

maple_driver_unregister

LINUX

Name

`maple_driver_unregister` — unregister a maple driver.

Synopsis

```
void maple_driver_unregister (struct maple_driver * drv);
```

Arguments

drv

maple driver to unregister.

Description

Cleans up after `maple_driver_register`. To be invoked in the exit path of any module drivers.

maple_getcond_callback

LINUX

Name

`maple_getcond_callback` — setup handling
`MAPLE_COMMAND_GETCOND`

Synopsis

```
void maple_getcond_callback (struct maple_device * dev, void  
(*callback) (struct mapleq *mq), unsigned long interval,  
unsigned long function);
```

Arguments

dev

device responding

callback

handler callback

interval

interval in jiffies between callbacks

function

the function code for the device

maple_add_packet

LINUX

Kernel Hackers Manual May 2009

Name

`maple_add_packet` — add a single instruction to the queue

Synopsis

```
int maple_add_packet (struct maple_device * mdev, u32
function, u32 command, size_t length, void * data);
```

Arguments

mdev

maple device

function

function on device being queried

command

maple command to add

length

length of command string (in 32 bit words)

data

remainder of command string

maple_add_packet_sleeps

LINUX

Kernel Hackers Manual May 2009

Name

maple_add_packet_sleeps — add a single instruction to the queue

Synopsis

```
int maple_add_packet_sleeps (struct maple_device * mdev, u32  
function, u32 command, size_t length, void * data);
```

Arguments

mdev

maple device

function

function on device being queried

command

maple command to add

length

length of command string (in 32 bit words)

data

remainder of command string

Description

Same as `maple_add_packet`, but waits for the lock to become free.