ALPHI_IO(C31_AMCC)
DSP Library User's Guide

SOFTWARE REFERENCE

745-05-001-4000
Revision F
May 1999

ALPHI Technology Corp.
6202 S. Maple Ave., #120
Tempe, AZ 85283
Tel: (480) 838-2428
Fax: (480) 838-4477
info@alphitech.com
www.alphitech.com

Table of Contents

Description

This DSP input / output library allows a DSP program to make use of several features in a relatively independent manner. The features supported are as follows:

Use of the onboard serial port as a source for standard input and output.

Ability to easily utilize additional 8530 resources, such as those on SCC-04 IPs.

Ability to access card hardware, especially for access with the HOST, in a relatively independent manner.

The documentation is divided into the following sections.

Modules The source files required to build the DSP Library portion of this board support package. Also, any global variables are listed here. Full source is provided for these files in the DSP source library called ALPHI_IO.SRC. Use the TI supplied AR30 utility to extract the source files from this file as follows.

AR30 x alphi_io.src

Functions C functions implementing the functionality of this library.

Structures and Enumerations Data objects used by this library.

Defines Constant data.

Hardware Independence, AMCC Access, Interrupts

When the alphi_io library is included in the build, several features are added to the DSP code running on the card. One main feature added is group of global variables which describe the environment that the DSP is operating in. Additional hardware functions allow for manipulation of the environment in a card independent manner. This facilitates an easy port of an application to a different ALPHI card or hardware platform.

The following functions will help in communication with the software running on the HOST.

host_ReadyIMbox host_ReadyOMbox host_ReadIMbox host_WriteOMbox host_ReadFIFO host_WriteFIFO

The following functions will help support DSP interrupts.

EnableIrq DisableIrq Forbid Permit c_int01 and the other default handlers.

To program DSP Timers, the #defines **TIMER_MICROSECONDS TIMER_MILLISECONDS TIMER_SECONDS** supply the values to be written to the period registers **T1_PERIOD** and **T2_PERIOD**. **T1_CTRL** and **T2_CTRL** access the control register, and **TIMER_GO TIMER_HLD CP_MODE** and **CLK_INT** are helpful control bits. The timer count registers are **T1_CNTR** and **T2_CNTR**.

To delay a programmed amount, the #defines **MICRO_SEC MILLI_SEC SECONDS** supply the values to call **delay** with.

The functions **SetST GetST SetIE GetIE SetIF** and **GetIF** allow access to the DSP registers.

Linking and Initialization

The library can be set to either redirect console output to the serial port available on many products, or to keep the serial output directed at the DSP emulator, when it is being used for software development.

When a DSP application is linked with the alphi_io library before the C runtime library (RTS30.lib), a hardware identification function **hardware_FindSystemInfo** is called which initializes the global variables based on information found in the NVRAM used by the AMCC chip. The Serial port redirection is also set up and initialized. This is done by the function c_int00 in alphi_bt.asm. c_int00 in RTS30.lib is not linked in because the alphi_io.lib was used first to resolve undefined symbols.

If the redirection of the standard I/O to the serial port is not desired, link the RTS30.lib file first, then alphi_io.lib. In your main() function, make a call to **hardware_FindSystemInfo** to initialize the hardware library. c_int00 in RTS30.lib is used, and the initialization call is not made automatically before main().

Serial Operation

When the alphi_io library is utilized to redirect the console output to the serial port, several features are added to the DSP code running on the card.

A SERIAL device handler is added to the runtime library allowing the user to open any device that the SERIAL driver knows about. Usually this is the two halves of the 8530 on the card, but other devices could be added by writing a device handler meeting the function prototype **Serial_Handler_t** and then calling **Serial_AddDevice** to add it to the list of devices.

An 8530 handler which knows how to handle 8530 class devices in a polled manner (without interrupt support) is made available to the SERIAL driver. This handler now supports backspaces and line kill (ESC).

And finally, a replacement HOST device for the TI supplied one (which normally communicates with the debugger through the emulator) redirects the console stdin, stdout, and stderr to the first serial device.

The 8530 driver initializes both SERIAL ports to 19200 baud, no parity, 8 bits, 1 stop bit. Textual mode is set which translates newlines into return newline pairs on output, and returns into newlines on input. Input characters are echoed. However, other modes can be set by changing the flags for the stream by calling **Serial_SetStreamFlags**.

Additional 8530 devices, such as those resident on the SCC-04 IP module, can be added to the SERIAL device by creating a **SC8530_t** structure and calling **Serial_AddDevice**.

Serial devices are opened by calling the standard IO library **fopen** or TI's **open** with a name "SERIAL:n" where n is the device number starting with 1. "SERIAL:1" is the serial port for the console device. "SERIAL:2" is the other half of the onboard 8530 chip. Any additional SERIAL devices, if present and made known to the SERIAL driver start with "SERIAL:3".

Remember that the standard IO routines buffer output by default. Sometimes it is necessary to call **fflush** to force output. Calling any of the input functions automatically

flushes output. Output to stderr is not buffered in the standard I/O library, and can be used as a work around.

Standard I/O Library

In the most recent version of the TI development tools (Version 5.0) there finally is support for the standard I/O library with the textual input and output directed to the debugger. The alphi_io library allows the user to replace the console supplied by the debugger with the serial port available on almost all of ALPHI Technology PCI cards. The library does this by replacing the lowest levels of the TI library with routines to direct output to the 8530 chip. No actual replacement of TI libraries are performed, instead, by placing the alphi_io.lib file before the rts30.lib file in the link command, the TI supplied functionality of debugger output is replaced by output to the serial port.

Look at the SerialEx DSP example to demonstrate simple debugging output to the serial port.

Software Modules

The source files required to build the DSP Library portion of this board support package. Also, any global variables are listed here. Full source is provided for these files in the DSP source library called ALPHI_IO.SRC. Use the TI supplied AR30 utility to extract the source files from this file as follows.

AR30 x alphi_io.src

Module 8530.c

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/8530.C

Description Implementation file for an 8530 class serial device.

Module 8530.h

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/8530.H

Description Include file for an 8530 class serial device.

Module alphi_io.c

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/ALPHI_IO.C

Description Implementation of the redirection of the standard input and output to the 8530 serial device.

Module alphi_io.h

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/ALPHI_IO.H

Description Declaration of the default serial device for standard I/O.

Module amcc.c

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Description Routines to access the add-on side of the AMCC.

Module def_int01.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT01.ASM

Description Default interrupt handler.

Module def_int02.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT02.ASM

Description Default interrupt handler.

Module def_int03.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT03.ASM

Description Default interrupt handler.

Module def_int04.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT04.ASM

Description Default interrupt handler.

Module def_int05.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT05.ASM

Description Default interrupt handler.

Module def_int06.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT06.ASM

Description Default interrupt handler.

Module def_int09.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT09.ASM

Description Default interrupt handler.

Module def_int10.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT10.ASM

Description Default interrupt handler.

Module def_int11.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT11.ASM

Description Default interrupt handler.

Module def_int99.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT99.ASM

Description Interrupt handler for unexpected interrupts.

Module flash.c

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/FLASH.C

Description Implementation of FLASH programming functions.

Module hardware.c

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.C

Description Hardware dependent variables initialized from AMCC NVRAM.

Global Variables

hardware_VendorId_t hardware_VendorId
Vendor ID of this board.

hardware_DeviceId_t hardware_DeviceId
Device ID of this board.

ulong hardware_SizeStaticRam
Number of DWORDS of DSP static memory.

void * hardware_AddrDualPortRam
Location of the dual ported RAM.

ulong hardware_SizeDualPortRam
Size of the dual ported RAM, or 0 if not present.

ulong hardware_ClockSpeedDsp
Clock speed of the DSP in Hertz.

ulong hardware_ClockSpeedSerial
Clock speed of the serial device in Hertz.

ubyte * hardware_AddrSerial
Address of the serial device.

hardware_SerialType_t hardware_SerialType
Type of the serial device. See **hardware_SerialType_t**.

AMCC_REG * hardware_AmccRegisters
Address of the AMCC add-on registers.

char * hardware_BoardName
Character stream identifying board name.

ulong hardware_NumberIpSlots
Number of IP slots accessible by DSP.

hardware_IpSlot_t hardware_IpSlot[]
Information describing each IP slot.

ubyte * hardware_pSystemFlash
Location of the system area of the FLASH.

ubyte * hardware_pUserFlash
Location of the user area of the FLASH.

ulong hardware_UserBoardId
User's board identification.

ulong hardware_BootOption
What to do at boot time.

Module hardware.h

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Description Declarations of the hardware identification code.

Module host.h

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HOST.H

Description Declarations of the command interface between the HOST and the DSP via the mailboxes.

Module nvram.c

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/NVRAM.C

Description Routines to access the NVRAM of the AMCC.

Module regs.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Description C callable functions to alter TMS machine registers.

Module serial.c

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/SERIAL.C

Description Implementation of a serial device.

Module serial.h

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/SERIAL.H

Description Declarations of the generic SERIAL device.

Module vecs.asm

Filename: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/VECS.ASM

Description Interrupt vector table.

C Functions and Callbacks

C functions implementing the functionality of this package.

c_int01

void c_int01()

Interrupt handler for External Interrupt 0.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT01.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_EINT0**

c_int02

void c_int02()

Interrupt handler for External Interrupt 1.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT02.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_EINT1**

c_int03

void c_int03()

Interrupt handler for External Interrupt 2.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT03.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_EINT2**

c_int04

void c_int04()

Interrupt handler for External Interrupt 3.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT04.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_EINT3**

c_int05

void c_int05()

Interrupt handler for Serial port 0 Transmitter.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT05.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_EXINT0**

c_int06

void c_int06()

Interrupt handler for Serial port 0 Receiver.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT06.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_ERINT0**

c_int09

void c_int09()

Interrupt handler for DSP Timer 0.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT09.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_ETINT0**

c_int10

void c_int10()

Interrupt handler for DSP Timer 1.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT10.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_ETINT1**

c_int11

void c_int11()

Interrupt handler for DSP DMA engine.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT11.ASM

Comments This default handler does nothing.

The customer replaces this function by defining his own function with this name. The linker will not extract this function if it is previously defined.

See Also **EnableIrq DisableIrq Permit Forbid CPU_EDINT**

c_int99

void c_int99()

Interrupt handler for unexpected interrupts.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/DEF_INT99.ASM

Comments This handler does nothing.

delay

void delay(
ulong *NumCycles***)**

Delay for the specified number of H1/H3 cycles.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Parameters

NumCycles
Number of cycles to delay.

See Also **MICRO_SEC MILLI_SEC SECONDS**

DisableIrq

void DisableIrq(
register uint *mask***)**

Select interrupt causes to be disabled.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.C

Parameters

mask
Interrupt causes to be disabled.

Comments More than one interrupt cause can be selected by ORing them together.

See Also **Defines**:CPU_E* and DMA_E* bits.

EnableIrq

void EnableIrq(
register uint *mask***)**

Select interrupt causes to be enabled.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.C

Parameters *mask*
Interrupt causes to be enabled.

Comments More than one interrupt cause can be selected by ORing them together.

See Also **Defines**:CPU_E* and DMA_E* bits.

Forbid

void Forbid()

Disable global interrupts.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.C

Comments Use this function to disable global interrupts. TI errata state that there is a race condition where certain operations which affect the GIE flag concurrent with a hardware interrupt will cause the GIE to be forced to a 0, disabling further interrupts. Using this function to control the global interrupts alleviates the problem.

See Also **Permit**

GetIE

ulong GetIE()

Get the present value of the DSP IE register.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Return Value The register contents.

GetIF

ulong GetIF()

Get the present value of the DSP IF register.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Return Value The register contents.

GetST

ulong GetST()

Get the present value of the DSP ST register.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Return Value The register contents.

---

hardware_FindSystemInfo

void hardware_FindSystemInfo()

Initialize all the hardware global variables based on values found in the AMCC's NVRAM.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.C

Comments This function is normally called by the overall library initialization code in **HOSTinit**, which is in turn called by the alphi_io C startup routine in alphi_bt.asm. This normally occurs before even the user's main() function is called.

However, sometimes it is desired to utilize alphi_io without redirecting the stdio output to the serial port, and thereby allowing for a smaller code image. To do this, the user should place the C runtime library BEFORE the alphi_io library, and call this function first in main() before doing anything else.

---

hardware_SelectAddressSpace

int hardware_SelectAddressSpace(
int *AddressSpace***)**

Switch between IP memory spaces and FLASH.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.C

Return Value Returns 0 or greater if successful. Returns less than zero if failed due to unrecognized hardware or parameter out of range.

Parameters *AddressSpace*
Desired address space.

Comments On intelligent IP carriers, each IP memory space and the boot FLASH device reside at the same starting address (0x400000). This function virtualizes switching between IP memory spaces and the FLASH device in a device independent manner. This allows code to perform this operation regardless of the platform differences.

This function is presently applicable to the CPCI-IPC, PCI-4Pack, CPCI-QIPC, and CPCI-Dual_IPC.

The *AddressSpace* 0 refers to the FLASH memory.

The *AddressSpace* 1 - 4 refers to IP memory spaces for A - D in 16 bit mode.

The *AddressSpace* 5 - 6 refers to IP memory spaces for AB and CD in 32 bit mode.

See Also **hardware_IpSlot_t hardware**:hardware_NumberIpSlots

host_ReadFIFO

ulong host_ReadFIFO()

Read a value from the 8 location AMCC FIFO.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Return Value The read in value.

Developer Notes There is no provision for HOST timeouts.

host_ReadIMbox

ulong host_ReadIMbox(
ushort *WhichMailbox***,**
boolean *wait***)**

Read a value from the specified inbound mailbox.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Return Value The read in value.

Parameters

WhichMailbox
Which mailbox is of interest.

wait
Should we wait for the HOST to write something fresh or return with the present value?

Developer Notes There is no provision for HOST timeouts.

host_ReadyIMbox

boolean host_ReadyIMbox(
ushort *WhichMailbox***)**

Is there fresh data in the specified inbound mailbox?

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Parameters

WhichMailbox
Which mailbox is of interest.

Return Codes

TRUE	Yes, there is.
FALSE	No, there isn't.

host_ReadyOMbox

boolean host_ReadyOMbox(
ushort *WhichMailbox***)**

Is it safe to write fresh data to the specified outbound mailbox because the HOST has read the previous value?

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Parameters

WhichMailbox
Which mailbox is of interest.

Return Codes

TRUE	Yes, there is.
FALSE	No, there isn't.

host_WriteFIFO

void host_WriteFIFO()

Write a value to the 8 location AMCC FIFO.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Developer Notes

There is no provision for HOST timeouts.

host_WriteOMbox

void host_WriteOMbox(
ushort *WhichMailbox***,**
ulong *val***,**
boolean *wait***)**

Write a value to the specified outbound mailbox.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Parameters

WhichMailbox
Which mailbox is of interest.

val
The value to be written.

wait
Should we wait for the HOST to read the present value or just overwrite?

Developer Notes

There is no provision for HOST timeouts.

nvram_ReadByte

ubyte nvram_ReadByte(
ulong *Address***)**

Read a byte from the NVRAM.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/NVRAM.C

Return Value Byte read.

Parameters

Address
Address to read.

nvram_WriteByte

void nvram_WriteByte(
ulong *Address***,**
ubyte *Data***)**

Write a byte to the NVRAM.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/NVRAM.C

Parameters

Address
Address to write.

Data
Data to write.

Permit

void Permit()

Enable global interrupts.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.C

Comments Use this function to enable global interrupts. TI errata state that there is a race condition where certain operations which affect the GIE flag concurrent with a hardware interrupt will cause the GIE to be forced to a 0, disabling further interrupts. Using this function to control the global interrupts alleviates the problem.

See Also **Forbid**

ReadFlashSized

void ReadFlashSized(
ubyte * *src*,
ulong * *dst*,
ulong *N*,
int *size*)

Read a entire buffer from the FLASH device, combining consecutive bytes into the target size.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/FLASH.C

Parameters

src
Location in the FLASH to read to the buffer.

dst
Location in the user's buffer.

N
Number of [bytes;words;longs] to transfer.

size
1, 2, or 4 for size of *dst*: [bytes;words;longs].

Return Codes

0	Success.
-1	Failure.

SetIE

void SetIE(
ulong *Value*)

Set the DSP IE register.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Parameters

Value
New value for the IE register.

SetIF

void SetIF(
ulong *Value*)

Set the DSP IF register.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Parameters

Value
New value for the IF register.

SetST

void SetST(
ulong *Value***)**

Set the DSP ST register.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

Parameters

Value
New value for the ST register.

WriteFlash

int WriteFlash(
ubyte * *src***,**
ubyte * *dst***,**
ulong *N***,**
boolean *fOutput***)**

Write a entire buffer to the device.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/FLASH.C

Parameters

src
Location in the user's buffer.

dst
Location in the FLASH to write the buffer.

N
Number of bytes to transfer

fOutput
Do we output to the console port?

Comments

Tracks progress or failure to the standard output.

Return Codes

0	Success.
-1	Failure.

WriteFlashSized

int WriteFlashSized(
ulong * *src***,**
ubyte * *dst***,**
ulong *N***,**
boolean *fOutput***,**
int *size***)**

Write a entire buffer to the FLASH device, splitting bufer into consecutive bytes.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/FLASH.C

Parameters

src
Location in the user's buffer.

dst
Location in the FLASH to write the buffer.

N
Number of [bytes;words;longs] to transfer.

fOutput
Do we output to the console port?

size
1, 2, or 4 for size of *src*: [bytes;words;longs].

Comments

Tracks progress or failure to the standard output.

Return Codes

0	Success.
-1	Failure.

xf0_off

void xf0_off()

Set hardware bit XF0 low.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

xf0_on

void xf0_on()

Set hardware bit XF0 high.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/REGS.ASM

IMBOX_MASK[] constant

ulong IMBOX_MASK[];

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Bitfields of interest for incoming mailboxes.

OMBOX_MASK[] constant

ulong OMBOX_MASK[];

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/AMCC.C

Bitfields of interest for outgoing mailboxes.

Defines

Constant data.

CACHE_CLEAR

#define CACHE_CLEAR (1 << 12)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

ST bit to clear the cache contents.

CACHE_ENABLE

#define CACHE_ENABLE (1 << 11)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

ST bit to enable fetching from the DSP cache.

CACHE_FREEZE

#define CACHE_FREEZE (1 << 10)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

ST bit to freeze the DSP cache.

CLK_FREQ

#define CLK_FREQ (hardware_ClockSpeedDsp)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Clock speed supplied to DSP.

CLK_INT

#define CLK_INT (1 << 9)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Timer control bit to select the clock source.

Comments When set to 1, the DSP clock is used by the timer. When set to 0, an external source provides the clock.

See Also **T1_CTRL T2_CTRL**

CNT_PERIOD

#define CNT_PERIOD (H1_PERIOD * 2.0)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Clock period supplied to internal timers.

CP_MODE

#define CP_MODE (1 << 8)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Timer control bit to set pulse or square wave mode.

Comments When set to 1, square wave mode is selected. When set to 0, pulse mode is selected. Note that the output frequency is one half when in square wave mode.

See Also **T1_CTRL T2_CTRL**

CPU_EDINT

#define CPU_EDINT (1 << 10)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from DSP DMA engine.

CPU_EINT0

#define CPU_EINT0 (1 << 0)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from External Interrupt 0.

CPU_EINT1

#define CPU_EINT1 (1 << 1)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from External Interrupt 1.

CPU_EINT2

#define CPU_EINT2 (1 << 2)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from External Interrupt 2.

CPU_EINT3

#define CPU_EINT3 (1 << 3)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from External Interrupt 3.

CPU_ERINT0

#define CPU_ERINT0 (1 << 5)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from DSP Serial 0 Receive.

CPU_ETINT0

#define CPU_ETINT0 (1 << 8)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from DSP Timer 0.

CPU_ETINT1

#define CPU_ETINT1 (1 << 9)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from DSP Timer 1.

CPU_EXINT0

#define CPU_EXINT0 (1 << 4)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP CPU interrupt from DSP Serial 0 Transmit.

DMA_EDINT

#define DMA_EDINT (1 << 26)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from DSP DMA engine.

DMA_EINT0

#define DMA_EINT0 (1 << 16)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from External Interrupt 0.

DMA_EINT1

#define DMA_EINT1 (1 << 17)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from External Interrupt 1.

DMA_EINT2

#define DMA_EINT2 (1 << 18)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from External Interrupt 2.

DMA_EINT3

#define DMA_EINT3 (1 << 19)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from External Interrupt 3.

DMA_ERINT0

#define DMA_ERINT0 (1 << 21)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from DSP Serial 0 Receive.

DMA_ETINT0

#define DMA_ETINT0 (1 << 24)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from DSP Timer 0.

DMA_ETINT1

#define DMA_ETINT1 (1 << 25)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from DSP Timer 1.

DMA_EXINT0

#define DMA_EXINT0 (1 << 20)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

IE bit to enable DSP DMA interrupt from DSP Serial 0 Transmit.

GLOBAL_IE

#define GLOBAL_IE (1 << 13)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

ST bit to enable global interrupts.

Comments Do not call **SetST** with this flag. Instead call **Permit** and **Forbid**.

H1_PERIOD

#define H1_PERIOD ((1.0 / CLK_FREQ) * 2.0)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

H1 / H3 clock period.

MBOX1

#define MBOX1 0

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Mailbox 1.

MBOX2

#define MBOX2 1

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Mailbox 2.

MBOX3

#define MBOX3 2

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Mailbox 3.

MBOX4

#define MBOX4 3

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Mailbox 4.

MICRO_SEC

#define MICRO_SEC ((int)((x*1E-6)/H1_PERIOD)-12)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Determine the number of H1/H3 cycles necessary for the specified delay.

Comments Since the calculations are somewhat lengthy, it is wise to calculate and save this value before delaying for short intervals.

See Also **delay**

MILLI_SEC

#define MILLI_SEC ((int)((x*1E-3)/H1_PERIOD)-12)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Determine the number of H1/H3 cycles necessary for the specified delay.

See Also **delay**

SECONDS

#define SECONDS ((int)((x)/H1_PERIOD)-12)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Determine the number of H1/H3 cycles necessary for the specified delay.

See Also **delay**

T1_CNTR

#define T1_CNTR ((volatile int *)0x808024)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

DSP Address for DSP Timer 0 count register.

Developer Notes Beware of the naming mismatch.

T1_CTRL

#define T1_CTRL ((volatile int *)0x808020)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

DSP Address for DSP Timer 0 control register.

Developer Notes Beware of the naming mismatch.

T1_PERIOD

#define T1_PERIOD ((volatile int *)0x808028)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

DSP Address for DSP Timer 0 period register.

Developer Notes Beware of the naming mismatch.

T2_CNTR

#define T2_CNTR ((volatile int *)0x808034)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

DSP Address for DSP Timer 1 count register.

Developer Notes Beware of the naming mismatch.

T2_CTRL

#define T2_CTRL ((volatile int *)0x808030)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

DSP Address for DSP Timer 1 control register.

Developer Notes Beware of the naming mismatch.

T2_PERIOD

#define T2_PERIOD ((volatile int *)0x808038)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

DSP Address for DSP Timer 1 period register.

Developer Notes Beware of the naming mismatch.

TIMER_GO

#define TIMER_GO (1 << 6)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Timer control bit to reset and start the timer, if it is not held.

See Also **T1_CTRL T2_CTRL**

TIMER_HLD

#define TIMER_HLD (1 << 7)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Timer control bit to hold the timer.

Comments If this bit is 0, the timer is held. If this bit is 1, the timer is free to start or continue.

See Also **T1_CTRL T2_CTRL**

TIMER_MICROSECONDS

#define TIMER_MICROSECONDS (int)((x*1E-6)/CNT_PERIOD + 0.5)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Determine the number of H1/H3 cycles necessary for the specified delay.

Comments This function calculates the value to be programmed in the timer register for the specified delay, given the DSP frequency. The value determined is that for pulse output mode. Square wave output will have a period twice as long.

TIMER_MILLISECONDS

#define TIMER_MILLISECONDS (int)((x*1E-3)/CNT_PERIOD + 0.5)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Determine the number of H1/H3 cycles necessary for the specified delay.

Comments This function calculates the value to be programmed in the timer register for the specified delay, given the DSP frequency. The value determined is that for pulse output mode. Square wave output will have a period twice as long.

TIMER_SECONDS

#define TIMER_SECONDS (int)((x)/CNT_PERIOD + 0.5)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Determine the number of H1/H3 cycles necessary for the specified delay.

Comments This function calculates the value to be programmed in the timer register for the specified delay, given the DSP frequency. The value determined is that for pulse output mode. Square wave output will have a period twice as long.

USERS_TEXT

#define USERS_TEXT (0x6000)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Starting location of the users code.

XFER_BUFFER_USERS_BSS

#define XFER_BUFFER_USERS_BSS (0x10000)

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Starting location of the TEKTRONICS transfer area.

Structures and Enumerations

Data objects used by this library.

AMCC_REG Structure

```
typedef struct {
    volatile ulong aimb[4];
    volatile ulong aomb[4];
    volatile ulong afifo;
    volatile ulong bmwar;
    volatile ulong apta;
    volatile ulong aptd;
    volatile ulong bmrar;
    volatile ulong ambef;
    volatile ulong aint;
    volatile ulong agcsts;
} AMCC_REG;
```

AMCC S5933 Operation Registers as seen from the DSP.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Members

aimb[4]
Add-On Incoming Mailbox Registers

aomb[4]
Add-On Outgoing Mailbox Registers

afifo
Add-On FIFO port (bidirectional)

bmwar
Bus Master Write Address Register

apta
Add-On Pass-Through Address

aptd
Add-On Pass-Through Data

bmrar
Master Read Address Register

ambef
Add-On Mailbox Empty/Full Status

aint
Add-On Interrupt Control/Status Register

agcsts
Add-On General Control/Status Register

hardware_DeviceId_t

```
enum hardware_DeviceId_t {
    DeviceID_CPCI_IPC,
    DeviceID_CPCI_SIP,
    DeviceID_CPCI_DIO,
    DeviceID_CPCI_AD16,
    DeviceID_CPCI_ADDA,
    DeviceID_CPCI_AD32,
    DeviceID_CPCI_ISC8,
    DeviceID_CPCI_QIPC,
    DeviceID_CPCI_QSIP,
    DeviceID_CPCI_Dual_IPC,
    DeviceID_CPCI_MFIO,
    DeviceID_PMC_1553,
    DeviceID_PCI_4PACK,
    DeviceID_PCI_QSIP,
    DeviceID_PCI_DIO,
    DeviceID_PCI_AD16,
    DeviceID_PCI_ISC8,
    DeviceID_Amcc
};
```

Recognized PCI Device IDs.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Members

DeviceID_CPCI_IPC
CPCI-IPC

DeviceID_CPCI_SIP
CPCI-SIP (No DSP)

DeviceID_CPCI_DIO
CPCI-DIO

DeviceID_CPCI_AD16
CPCI-AD16

DeviceID_CPCI_ADDA
CPCI-DSC4 or CPCI-ADDA (same card, really)

DeviceID_CPCI_AD32
CPCI-AD32

DeviceID_CPCI_ISC8
CPCI-ISC8

DeviceID_CPCI_QIPC
CPCI-QIPC

DeviceID_CPCI_QSIP
CPCI-QSIP (No DSP)

DeviceID_CPCI_Dual_IPC
CPCI-Dual IPC (One IPC of the two present on the card)

DeviceID_CPCI_MFIO
CPCI-MFIO

DeviceID_PMC_1553
PMC-1553 (No DSP)

DeviceID_PCI_4PACK
PCI-4Pack

DeviceID_PCI_QSIP
PCI-QSIP (No DSP)

DeviceID_PCI_DIO
PCI-DIO

DeviceID_PCI_AD16
PCI-AD16

DeviceID_PCI_ISC8
PCI-ISC8

DeviceID_Amcc
Default (unprogrammed) AMCC.

hardware_IpSlot_t Structure

```
typedef struct {
    ulong * pIdSpace;
    ulong * pIoSpace;
    ulong * pMemSpace;
    ulong * pIntAckSpace;
} hardware_IpSlot_t;
```

Definition of an IP slot (as seen from the DSP).

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Members

pIdSpace
Where to find the ID space.

pIoSpace
Where to find the IO space.

pMemSpace
Where to find the Memory space. See **hardware_SelectAddressSpace** to switch address spaces.

pIntAckSpace
Where to find the interrupt vector space.

hardware_SerialType_t

```
enum hardware_SerialType_t {
    SERIAL_NONE,
    SERIAL_8530,
    SERIAL_INTERNAL
};
```

Recognized serial devices.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Members

SERIAL_NONE
None.

SERIAL_8530
Dual channel SC8530, with the first as console.

SERIAL_INTERNAL
Internal simulated serial device in ALTERA programmed logic.

hardware_VendorId_t

```
enum hardware_VendorId_t {
    VendorId_Alphi,
    VendorId_Amcc
};
```

Recognized PCI Vendor IDs.

Defined in: D:/ALPHIPCI/ALPHI_IO/C31_AMCC/HARDWARE.H

Members

VendorId_Alphi
ALPHI Technology PCI Card.

VendorId_Amcc
Default (unprogrammed) AMCC.