A Semtech Company

Technical Reference Manual
USAR ACPITroller™ 342
System Management Family

UR8HC342
HID & ACPI Embedded Controller

Confidential
Preliminary

Document Number: DOC8-342-TR-080
Date: September 2000

Table of contents

Chapter 4 / AlphaMouse™ pointing devices manager

Chapter 5 / Embedded controller interface

Chapter 6 / SCI & SWI interrupt generation

Chapter 7 / SMBus host controller interface

Chapter 8 / General input/output options

This page intentionally left blank

Introduction

Document revisions

Document revision history

Date	Version	Comments
1999/07/28	0.60	Initial draft
1999/09/14	0.70	Incorporated technical corrections and clarifications; grayed out features that are still in development
2000/09/11	0.80	Incorporated technical corrections and clarifications

This page intentionally left blank

USAR ACPITroller™ technical reference introduction

Overview

The UR8HC342 is a single IC that functions both as an 8042-type Human Input Device Controller (HIDC) and an ACPI-compliant Embedded Controller (EC). The UR8HC342 provides the typical functionality of an 8042-type HID Controller with embedded key and motion scanning. In addition, the UR8HC342 functions as an ACPI compliant Embedded Controller (EC) and SMBus host.

The IC achieves unparalleled minimum power consumption (typically less than 1μA) due to USAR's patented Zero-Power™ technologies for both PS/2 ports and the SMBus port – an industry first. The USAR ACPITroller™ can power down even when devices are connected and active. Based on USAR's patented Zero-Power™ technology, the UR8HC342 always operates in the "STOP" mode, independently of the configuration and without any data or event losses.

The USAR AlphaKey™ keyboard manager, implemented in the UR8HC342, provides OEMs with the most advanced and versatile keyboard management functions, including keyboard matrix programming.

The USAR AlphaMouse™ pointing device manager supports MouseWheel operation, recommended by Microsoft, for both the embedded and the hot-plugged external pointing devices.

Using the UR8HC342, system designers can implement systems that take advantage of the SMBus, the Smart Battery System, and the ACPI specifications, all using a single IC. The UR8HC342 uses USAR's patented Zero-Power™ SMBus technology and is the lowest power consumption IC in the market today. The UR8HC342 can be customized easily through an extensive library of hardware and firmware modules in order to accommodate specialized configurations at low production cost.

SMBus Host

The IC manages one hardware SMBus port. The IC complies with version 1.0 of the Smart Battery System (SBS) and SMBus specifications.

Scope of this document

This is a preliminary document. Although all the functions described in this document have been tested in USAR's laboratories, the document may contain errors and inaccuracies. Prior to committing to design, consult with USAR for up to date revisions. Some features described in this document are still in development; these features are identified by grayed-out text.

USAR has used every effort to design the part in the best manner to serve the current industry needs. Nevertheless, input from OEMs for both function and configuration is appreciated and may result in future modifications and enhancements.

This document is still in a confidential state, as printed on the bottom of each page, and it cannot be distributed without explicit permission from USAR.

Please forward all your comments or questions regarding this product and document to:

acpi@usar.com

USAR Systems, Inc.
568 Broadway
New York, NY 10012
212.226.2042 Telephone
212.226.3215 Telefax

Evaluation kits

For information or to order a USAR ACPITroller™ evaluation kit, send your email to:

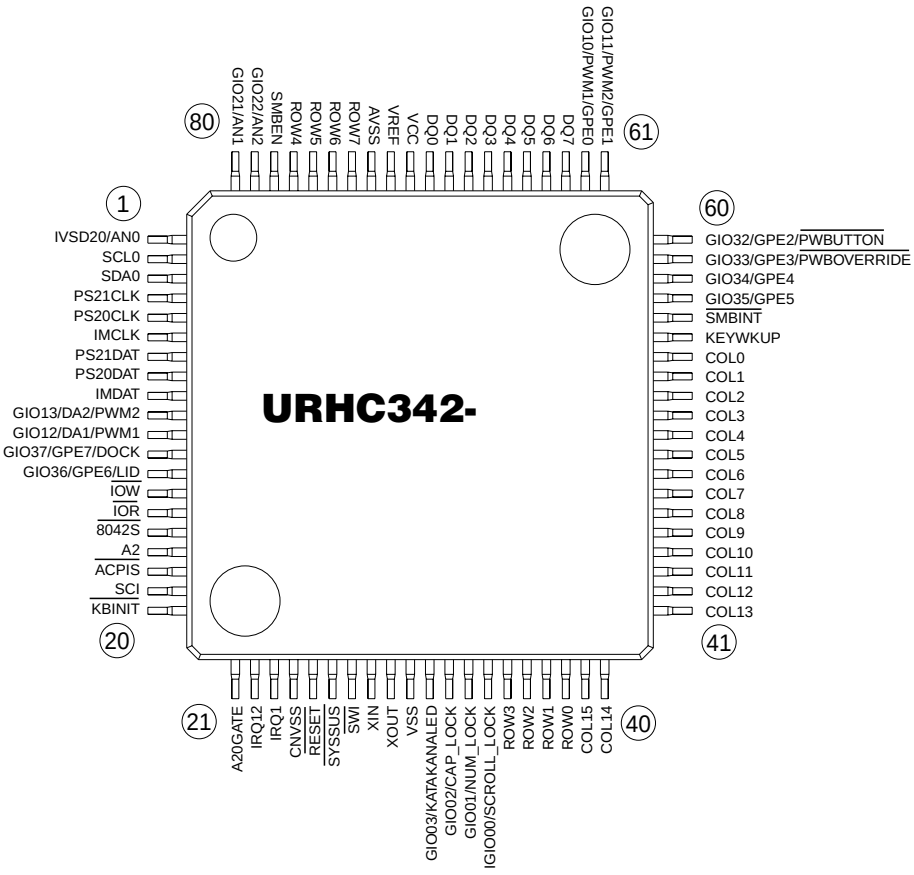
info@usar.com

Features at a glance

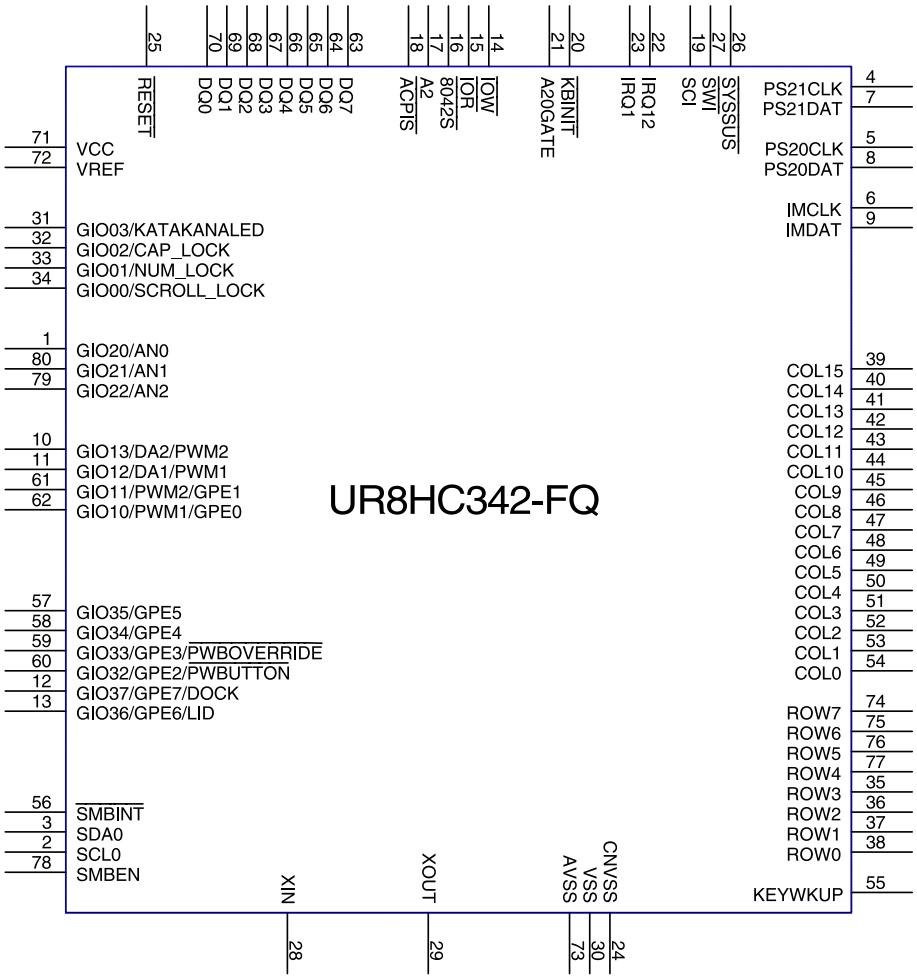
- Typical power consumption of less than 1μA as a result of USAR's patented Zero-Power™ technologies; the IC can power down even when devices are connected and active
- Patented Zero-Power™ operation of all PS/2 ports and the SMBus port – an industry first
- AlphaMouse™ pointing device manager supports hot-plugging and hot-swapping of standard two-button and three-button mice without a special driver; it also supports hot-plugging and hot-swapping of MouseWheel mice with a MouseWheel-capable driver
- Two external PS/2 ports for external keyboard and mouse with auto-detect and hot-plug support
- Simultaneous operation of external and internal input devices
- Support of MouseWheel functionality for both embedded and external pointing devices, even with hot-plug connections, with a MouseWheel-capable driver
- 8042-compatible host interface and functionality
- 8 x 16 fully programmable scanned keyboard matrix
- Support of all three keyboard scan code sets
- ACPI EC host interface
- Support of up to 6 ACPI GPE interrupt inputs
- ACPI Power Button and Power Button Override support
- SMBus compatible host complies with version 1.0 of the SBS/SMBus specifications
- Blocks potentially dangerous Smart Battery System commands (write charger voltage and write charger current) issuing from the host system
- Up to three 10-bit A/D inputs
- Two D/A and two PWM outputs
- Easily customized for specialized applications
- Three-volt and five-volt operation
- AlphaMouse™ performs active PS/2 multiplexing of input from different types of mice simultaneously, with the appropriate driver(s)
- Support of the USAR ScreenCoder™ PS/2 absolute and relative touchscreen encoder, with the appropriate driver
- Support of the five-button mouse, with an appropriate driver

Pin overview

The figure below illustrates the pins for the UR8HC342 implementation for the FQ package.



Pin usage


UR8HC342-FQ
25 RESET
70 DQ0
69 DQ1
68 DQ2
67 DQ3
66 DQ4
65 DQ5
64 DQ6
63 DQ7
18 ACPIS
17 A2
16 8042S
15 IOR
14 IOW
21 A20GATE
20 KBINIT
23 IRQ1
22 IRQ12
19 SCI
27 SWI
26 SYSSUS
71 VCC
72 VREF
31 GIO03/KATAKANALED
32 GIO02/CAP_LOCK
33 GIO01/NUM_LOCK
34 GIO00/SCROLL_LOCK
1 GIO20/AN0
80 GIO21/AN1
79 GIO22/AN2
10 GIO13/DA2/PWM2
11 GIO12/DA1/PWM1
61 GIO11/PWM2/GPE1
62 GIO10/PWM1/GPE0
57 GIO35/GPE5
58 GIO34/GPE4
59 GIO33/GPE3/PWBOVERRIDE
60 GIO32/GPE2/PWBUTTON
12 GIO37/GPE7/DOCK
13 GIO36/GPE6/LID
56 SMBINT
3 SDA0
2 SCL0
78 SMBEN
28 XIN
29 XOUT
73 AVSS
30 VSS
24 CNVSS
4 PS21CLK
7 PS21DAT
5 PS20CLK
8 PS20DAT
6 IMCLK
9 IMDAT
39 COL15
40 COL14
41 COL13
42 COL12
43 COL11
44 COL10
45 COL9
46 COL8
47 COL7
48 COL6
49 COL5
50 COL4
51 COL3
52 COL2
53 COL1
54 COL0
74 ROW7
75 ROW6
76 ROW5
77 ROW4
35 ROW3
36 ROW2
37 ROW1
38 ROW0
55 KEYWKUP

Pin description

The following table describes the pins of the USAR UR8HC342 ACPITroller™.

Note: If a pin is active low, then there is an overscore over the pin name in schematics, and there is a single underscore character (_) preceding the pin name in tables and text; for example, _RESET.

ACPITroller™ pin descriptions

Pin Name	Pin no.	Description
Power Supply		
AVSS	73	Analog signal ground
CNVSS	24	Should be tied to ground
VCC	71	Vcc 3-5 volts
VREF	72	Analog circuitry reference voltage
VSS	30	Ground
Oscillator pins		
XIN	28	Oscillator input (8 MHz operating frequency)
XOUT	29	Oscillator output
Reset		
_RESET	25	Controller hardware reset pin
System bus interface pins		
_8042S	16	8042 keyboard controller port select signal input
_ACPIS	18	ACPI embedded controller port select signal input
_IOR	15	X-bus/ISA I/O read signal input
_IOW	14	X-bus/ISA I/O write signal input
IRQ1	23	Keyboard interrupt output
IRQ12	22	Mouse interrupt output
_KBINIT	20	Keyboard initialize output
A2	17	X-bus/ISA address 2 input
A20GATE	21	A20 Gate output signal
DQ0	70	X-bus/ISA parallel data I/O ports
DQ1	69	
DQ2	68	
DQ3	67	
DQ4	66	
DQ5	65	
DQ6	64	
DQ7	63	

ACPITroller™ pin descriptions

Pin Name	Pin no.	Description
ACPI, SMBus, and general purpose I/O signals		
SCI	19	System control interrupt output
_SMBINT	56	SMBus interrupt input
_SWI	27	System wake-up event interrupt output
_SYSSUS	26	System suspend input
GIO00 /SCROLL_LOCK	34	This pin can be programmed to act as a keyboard LED or as a GPIO pin.
GIO01/NUM_LOCK	33	This pin can be programmed to act as a keyboard LED or as a GPIO pin.
GIO02/CAP_LOCK	32	This pin can be programmed to act as a keyboard LED or as a GPIO pin.
GIO03 /KATAKANALED	31	This pin can be programmed to act as a keyboard LED or as a GPIO pin.
GIO10/PWM1/GPE0	62	GIO1 bit 0 or PWM output or ACPI GPE
GIO11/PWM2/GPE1	61	GIO1 bit 1 or PWM output or ACPI GPE
GIO12/DA1/PWM1	11	This pin can be configured as GPIO, as D/A output, or PWM output.
GIO13/DA2/PWM2	10	This pin can be configured as GPIO, as D/A output, or PWM output.
GIO20/AN0	1	This pin can be configured as a 10-bit A/D input or logic I/O.
GIO21/AN1	80	This pin can be configured as a 10-bit A/D input or logic I/O.
GIO22/AN2	79	GIO2 bit or A/D input
GIO36/GPE6/LID	13	GIO3 bit 0 or ACPI general purpose event (GPE); capable of detecting both negative and positive signal transitions; typically serves the LID ACPI function
GIO37/GPE7/DOCK	12	GIO3 bit 1 or ACPI GPE; capable of detecting both negative and positive signal transitions; typically serves the DOCK ACPI function
GIO32/GPE2 /_PWBUTTON	60	GIO3 bit 2 or ACPI GPE; typically an ACPI ""Power Button"" input.
GIO33/GPE3 /_PWBOVERRIDE	59	GIO3 bit 3 or ACPI GPE; typically an ACPI ""Power Button Override"" input
GIO34/GPE4	58	GIO3 bit 4 or ACPI GPE 4
GIO35/GPE5	57	GIO3 bit 5 or ACPI GPE 5
SCL0	2	This pin acts as the clock line for the SMBus
SDA0	3	This pin acts as the data line for the SMBus

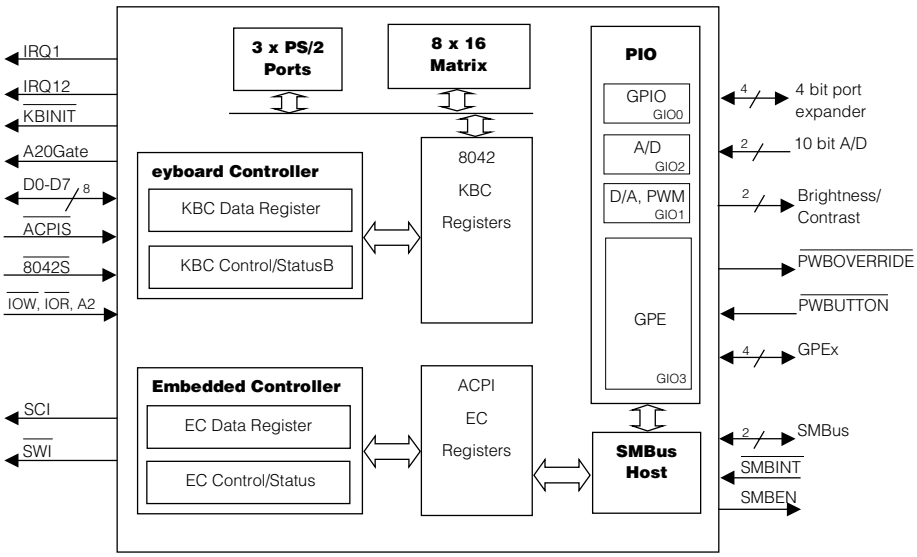
ACPITroller™ pin descriptions

Pin Name	Pin no.	Description
ACPI, SMBus, and general purpose I/O signals		
SMBEN	78	This output pin allows the SMBus latch to notify the ACPITroller™ when SMBus activity has been detected. It is used to wake the ACPITroller™ from sleep mode, and to disable all PS/2 inputs while SMBus processing is taking place.
Scanned matrix pins		
COL0	54	Column matrix outputs
COL1	53	
COL2	52	
COL3	51	
COL4	50	
COL5	49	
COL6	48	
COL7	47	
COL8	46	
COL9	45	
COL10	44	
COL11	43	
COL12	42	
COL13	41	
COL14	40	
COL15	39	
KEYWKUP	55	Key wake-up output
ROW0	38	Row matrix inputs
ROW1	37	
ROW2	36	
ROW3	35	
ROW4	77	
ROW5	76	
ROW6	75	
ROW7	74	
PS/2 ports		
IMCLK	6	PS/2 clock line for internal mouse
IMDAT	9	PS/2 data line for internal mouse
PS20CLK	5	Clock line for external PS/2 port 0; both external PS/2 ports support hot-plug ins and auto-select for keyboard or mouse
PS20DAT	8	Data line for external PS/2 port 0
PS21CLK	4	Clock line for external PS/2 port 1
PS21DAT	7	Data line for external PS/2 port 1

USAR ACPITroller™ sample configuration

The USAR ACPITroller™ is a highly versatile part that can be configured to accommodate different OEM configurations. The block diagram below illustrates the major functional components of the USAR ACPITroller™ in a simple configuration.

USAR ACPITroller™ sample configuration



Simple configuration block diagram of UR8HC342

In the configuration shown above the USAR ACPITroller™ is configured to provide besides the basic EC and KC functionality the following:

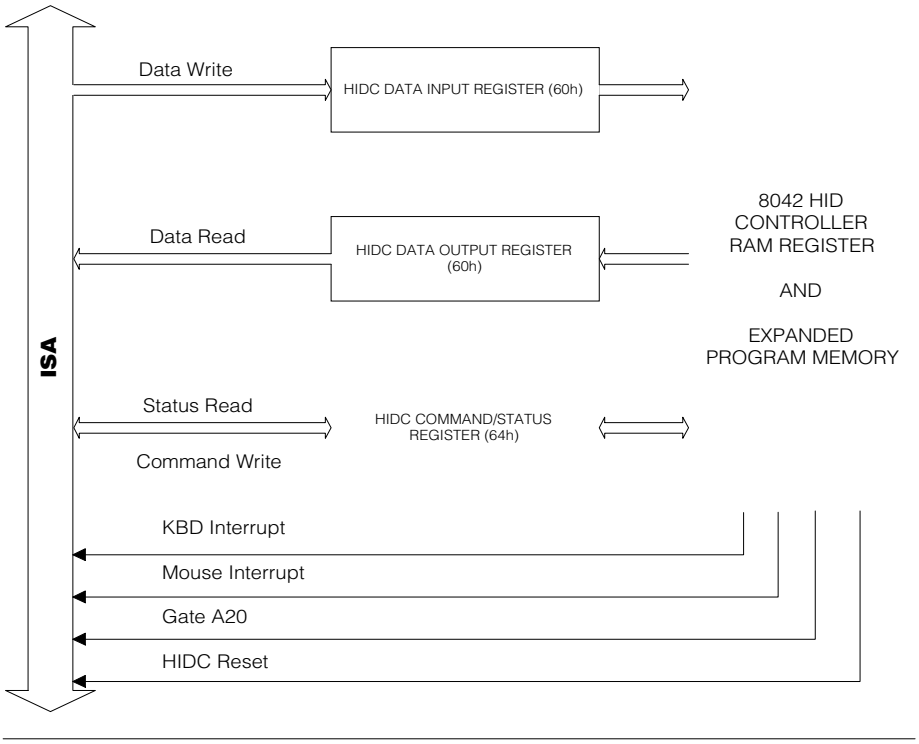
- Power button and power button override function implemented on GPE pins
- 4 general-purpose I/O pins
- 4 additional GPE inputs
- 3 pins that can be configured as 10 bit A/D channels or GPIO
- 2 pins that can be configured as D/A or PWM outputs or GPIO
- One Zero-Power™ SMBus port

This page intentionally left blank

USAR ACPITroller™ human input device controller (HIDC) interface

HIDC bus interface

The Human Input Device Controller (HIDC) portion of the USAR ACPITroller™ interfaces to the host ISA bus via two I/O addresses, generally 0x60 and 0x64. The following diagram illustrates the HIDC register architecture.



USAR ACPITroller™ HID Controller Interface

USAR ACPITroller™ HIDC registers

The HIDC portion of the USAR ACPITroller™ contains three registers that occupy two I/O locations. The registers are listed in the following table:

HIDC registers

Name	Description	I/O function	I/O Port Address
HIDC_SC(W)	Command	IOW	0x64
HIDC_SC(R)	Status	IOR	0x64
HIDC_DATA	Data	IOR/IOW	0x60

HIDC Status Register, HIDC_SC (R)

The HIDC Status Register is a read-only register that indicates the current status of the Keyboard Controller interface. It contains the following fields:

HIDC_SC (R) status register bit definitions

bit7	bit6	bit5	bit4
HIDC_PE	HIDC_GTO	HIDC_AOBF	HIDC_INH

bit3	bit2	bit1	bit0
HIDC_CMD	HIDC_SF	HIDC_IBF	HIDC_OBF

HIDC_SC (R) status register bit descriptions

HIDC_PE	Parity Error 1 – Indicates a parity error. In this case 0xFF is placed in the data register 0 – Indicates that the last byte of data received from the PS/2 device had odd parity
HIDC_GTO	General Time-Out 1 – Indicates that a time-out occurred during a transaction with one of the PS/2 devices. The HIDC_GTO is set in any of the following situations: – Reception of a byte from a PS/2 device started but did not complete within the Receive Time Out limit. The HIDC places 0xFF into the data register – A transmission started by the HIDC to a PS/2 device, but was not completed within the Transmit Time Out limit. The HIDC places 0xFE into the data register – A byte requiring response (command) was clocked out to a PS/2 device but it was not acknowledged within the ACK Time Limit – A command byte was clocked out and a response was received with a parity error. In this case both HIDC_GTO and HIDC_PE are set 0 – No Time-Out
HIDC_AOBF	Auxiliary Output Buffer Full This bit works in conjunction with HIDC_OBF (Output Buffer Full) bit 1 – When HIDC_OBF is also set, indicates that data from the Auxiliary device is pending in the data register 0 – When HIDC_OBF is set, indicates that Keyboard data or a HIDC response is pending in the data register
HIDC_INH	Inhibit Switch 1 – Keyboard is enabled 0 – Password state is active and the keyboard is inhibited (For more information, see section below on Password Protection)
HIDC_CMD	Command/Data 1 – Data register contains a command byte (set by the host, read by the HIDC) 0 – Data register contains a data byte (set by the host, read by the HIDC)
HIDC_SF	System Flag 1 – The System Flag bit in the Controller Command byte is set to one 0 – The System Flag bit in the Controller Command byte is set to zero (default after reset)
HIDC_IBF	Input Buffer Full 1 – Input buffer is full (data has been written in the data register but has not been read by the HIDC) 0 – Input buffer is empty (data has been read by the HIDC)
HIDC_OBF	Output Buffer Full 1 – Data ready for host in the Data Register (generate appropriate interrupt) 0 – Data has been read by the host (set automatically after an IO read operation)

Command register, HIDC_SC (W)

The HIDC_SC (W) command register is a write-only register that allows commands to be issued to the keyboard controller. Write operations to this port are latched into the input data register; the input buffer full (HIDC_IBF) flag is set in the status register. Writes to this location also cause the HIDC_CMD (Command/Data) bit to be set in the status register. This

enables the keyboard controller to differentiate the start of a command sequence from a data byte write operation.

Data Register, HIDC_DATA(R/W)

The HIDC_DATA(R/W) data register is a read/write register that allows command/data bytes to be issued to the keyboard controller while also enabling the host system to read data returned by the keyboard controller. Writes to this port are latched into the input data register; the input buffer full (HIDC_IBF) flag is set in the status register. Data written by the system into this register is generally transmitted to the appropriate PS/2 device, unless the HIDC expects a data byte as part of a command sequence. Reads from this register return data from the output data register and clear the output buffer full (HIDC_OBF) flag in the status register.

HIDC Commands

Any byte written by the host system into the HIDC_SC register is interpreted as a command. The USAR ACPITroller™ supports all the standard 8042 commands, as well as expanded commands, as described below.

Standard 8042 commands

Hex Value	Description
20	Read Controller Command byte
21-3F	Read HIDC RAM Registers. Address is specified by bit0-bit5
60	Write Controller Command byte. The following byte is data
61-7F	Write HIDC RAM Registers. Address is specified by bit0-bit5, and the following byte is data
A4	Test Password Installed. The result is placed in the data buffer as follows: – 0xFA – installed – 0xF1 – not installed (See below)
A5	Load Password. Data follows until a null (0) is detected (See below)
A6	Enable Password. This command is valid only when a password is loaded in the controller (See below)
A7	Disable Auxiliary Device Interface. This command sets bit 5 of the Controller Command byte and disables the Auxiliary Device clock line
A8	Enable Auxiliary Device Interface. This command clears bit 5 of the Controller Command byte and enables the Auxiliary Device clock line
A9	Auxiliary Device Interface Test. Test results are returned in the data buffer as follows: – 00 – No error – 01 – Clock line stuck low – 02 – Clock line stuck high – 03 – Data line stuck low – 04 – Data line stuck high
AA	Controller self test, Return 55.
AB	Keyboard Device Interface Test. Test results are returned in the data buffer as follows: – 00 – No error – 01 – Clock line stuck low – 02 – Clock line stuck high – 03 – Data line stuck low – 04 – Data line stuck high
AD	Disable Keyboard Interface. This command sets bit 4 of the Controller Command byte and disables the Keyboard clock line
AE	Enable Keyboard Interface. This command clears bit 4 of the Controller Command byte and enables the Keyboard clock line
C0	Read Input Port and place data in the data register
C2-C3	Poll Input Port and update bit7-bit4 of the Status Register
D0	Read Output Port and place data in the data register
D1	Write Output Port. The next byte is written to the HIDC Output Port
D2	Write Keyboard Output Buffer. The next byte is played back as if originating from the keyboard
D3	Write Auxiliary Device Output Buffer. The next byte is played back as if originating from the Auxiliary Device

Standard 8042 commands

Hex Value	Description
D4	Write to Auxiliary Device. The next byte is transmitted to the Auxiliary Device
E0	Read Test Inputs. Returns 0 (Type 1 controller)
F0-FF	Pulse Output Port. This command pulses Output Port bits, defined by bit0-bit3, for approximately 6 us.

Keyboard password

The USAR ACPITroller™ supports keyboard password functionality. There are three commands associated with the password operation.

- Load Password (0xA5)
- Test Password Installed (0xA4)
- Enable Password (0xA6)

The password can be loaded into the USAR ACPITroller™ RAM area by the system at any time, using the “Load Password” command. The password can be up to seven bytes long and is loaded using Scan Code 1 (scan codes provided by the system).

The system can check if a password is installed using the “Test Password Installed” command. If a password is already installed, the system can enable the password by issuing the “Enable Password” command.

While the password is enabled, the HIDC enters the secure mode and behaves as follows:

1. The HIDC intercepts any incoming code stream from the keyboard and compares it with what is installed in the RAM password pattern. The HIDC discards any code from the keyboard and the auxiliary device that does not match the password. If an incoming code does not match either the next character in the pattern or the contents of RAM Register addresses 0x16 and 0x17, the HIDC resets its password pointer and the next incoming code is compared to the first byte of the password pattern.
2. While in secure mode, the HIDC does not pass any codes to the system or accept any commands.
3. After a match occurs, the HIDC resumes normal operation and starts passing codes to the host system.

There are four HIDC RAM Registers related to the password operation:

Password associated RAM registers

RAM Register	Address	Description
Security on	13H	When the password is enabled, a non-zero value in this register forces the HIDC to output the contents of the register into the Data Register and issue IRQ1
Security off	14H	When the password is matched, a non-zero value forces the HIDC to output the register contents into the Data Register and issue IRQ1
Make 1 & 2	16H, 17H	These registers can be loaded by the system with scan codes that should be ignored during the password match process (i.e., shift codes)

HIDC RAM registers

The HIDC has on-board RAM registers, listed in the table below. HIDC RAM Registers are accessed for read and write operations through the Read/Write HIDC RAM Register commands. A description of specific RAM registers follows.

HIDC RAM registers

Address Offset	Description of Register
00	Controller Command byte (Read or Write)
01-12	Not defined, system use
13	Security on
14	Security off
16-17	Make 1 and 2

Controller command byte

The Controller Command byte occupies address offset zero in the HIDC RAM Register space. The Controller Command byte can be accessed through general Read/Write or specific commands and contains the following bits:

Controller command byte bit definitions

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
RES	TRS	ADIS	KDIS	RES	SF	AIRQ	KIRQ

Controller command byte bits descriptions

RES	Reserved
TRS	Keyboard Translate 1 – Sets keyboard translation from Scan Code Set 2 to Scan Code Set 1 0 – Disables keyboard scan code translation
ADIS	Disable Auxiliary Device 1 – HIDC disables the Auxiliary device by pulling the clock line low 0 – HIDC enables the Auxiliary device
KDIS	Disable Keyboard 1 – HIDC disables the keyboard by pulling the clock line low 0 – HIDC enables the keyboard
SF	System Flag The value written in this bit is placed in the relevant bit of the Status Register (system use)
AIRQ	Enable Auxiliary Device Interrupts Setting this bit to 1 enables IRQ12 whenever an auxiliary device data byte is placed into the data register
KIRQ	Enable Keyboard Interrupts Setting this bit to 1 enables IRQ1 whenever a keyboard data byte is placed into the data register

Security on

The Security On register is used for Keyboard Password functionality. When the password is enabled, a non-zero value in this register forces the HIDC to output the contents of the register into the Data Register and issue IRQ1.

Security off

The Security Off register is used for Keyboard Password functionality. When the password is matched, a non-zero value forces the HIDC to output the register contents into the Data Register and issue IRQ1.

Make 1 and 2

These registers are used for Keyboard Password functionality and can be loaded by the system with scan codes that should be ignored during the password match process (i.e., shift codes).

AlphaKey™ Control Register

This register controls local handling of the PS/2 keyboard protocol, defines pins as GPIO or keyboard LEDs, and enables and disables programming of the embedded keyboard matrix. Functional details are described in full in the AlphaKey™ Keyboard Manager chapter.

HIDC USAR expanded command set

In addition to the standard 8042 commands, the HIDC section implements a set of Expanded Commands, defined by USAR in order to accommodate the special functions implemented in the USAR ACPITroller™ HID section. These commands provide BIOS, Drivers and application software with access to the following unique features of the USAR ACPITroller™:

- Version control
- OEM Programmable Area
- Scanned Keyboard Matrix programmability

HIDC expanded commands

Hex Value	Description
B0	Read model number, 1Byte Bit7: – 0: Standard part – 1: Customized part Bit6-Bit0: model number A value of 01H denotes the basic configuration
B1	Read revision number, 1byte
B2	Write Program RAM Page Register Valid Values: 0, 1
B3	Read Program RAM Page Register, 1 byte
B4	Write Program RAM Pointer Register Valid Values: 0 – FFh for page 0; 0 – 7Fh for page 1;
B5	Read Program RAM Pointer Register, 1 byte
B6	Write to Program RAM in given page and current address
B7	Read from Program RAM in given page and current address
B8	Write data block to Program RAM, starting at specified address. This command is followed by one byte block size, one byte address and data
B9	Read from Program RAM by specified address
BA	Download Whole Matrix Layout0, follow by 128 bytes matrix
BB	Download Whole Num Lock and Fn Matrix Layout, follow by 96 bytes matrix
BC	Download Whole Macro Function Pointer, follow by 16 bytes pointer
BD	Download Whole Macro Function Code, follow by 128 bytes code

For commands B6 and B7, the RAM pointer register automatically increments after each byte is read or written. The increment wraps from FFh to 00h within RAM Page 0 (you cannot implicitly write to Page 1), and from 7Fh to 00h within RAM Page 1.

For the download commands (BA, BB, BC, and BD), you must issue the command in the command/status register, than send each byte of data, one at a time, in the data register. You must verify that the input buffer full bit (bit 1 of the HIDC status register, as shown earlier in this chapter) has been cleared before sending the next byte. Each of these commands expects a

fixed number of data bytes. The HIDC must read all of the expected bytes before another command issues. If a new command issues before all expected bytes are read, the download command aborts and the rest of the matrix is unchanged.

PS/2 information registers

Starting at offset 15 from the start of the 8042 extended register area, there are three read-only registers with information about the three PS/2 ports: registers R15-R17.

R15: 8042 configuration register 0 bit definitions (R)

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
MXEN	RES	M3PEND	M2PEND	M1PEND	M3DIS	M2DIS	M1DIS

R15: 8042 configuration register 0 bit descriptions (R)

RES	reserved
MXEN	1– Mouse multiplexing mode enabled by command
M3PEND	1– Mouse multiplexing mode command pending port 3
M2PEND	1– Mouse multiplexing mode command pending port 2
M1PEND	1– Mouse multiplexing mode command pending port 1 (internal)
M3DIS	1– Direct port 3 clock disable by command
M2DIS	1– Direct port 2 clock disable by command
M1DIS	1– Direct port 1 clock disable by command

R16: 8042 configuration register 1 bit definitions (R)

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
OK2	KB2	5BUT2	WHEEL2	OK3	KB3	5BUT3	WHEEL3

R16: 8042 configuration register 1 bit descriptions (R)

OK2	1– External device on port 2 initialized OK
KB2	1– Device on port 2 is a keyboard 0– Device on port 2 is a mouse
5BUT2	1– Device on port 2 is a 5-button mouse
WHEEL2	1– Device on port 2 is a wheel mouse
OK3	1– External device on port 3 initialized OK
KB3	1– Device on port 3 is a keyboard 0– Device on port 3 is a mouse
5BUT3	1– Device on port 3 is a 5-button mouse
WHEEL3	1– Device on port 3 is a wheel mouse

R17: 8042 configuration register 2 bit definitions (R)

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RES	RES	RES	RES	RES	OK1	5BUT1	WHEEL1

R17: 8042 configuration register 2 bit descriptions (R)

RES	reserved
OK1	1– Internal device on port 1 initialized OK
5BUT1	1– Device on port 3 is a 5-button mouse
WHEEL1	1– Device on port 3 is a wheel mouse

This page intentionally left blank

USAR UR8HC342 ACPITroller™ AlphaKey™ keyboard manager

Overview

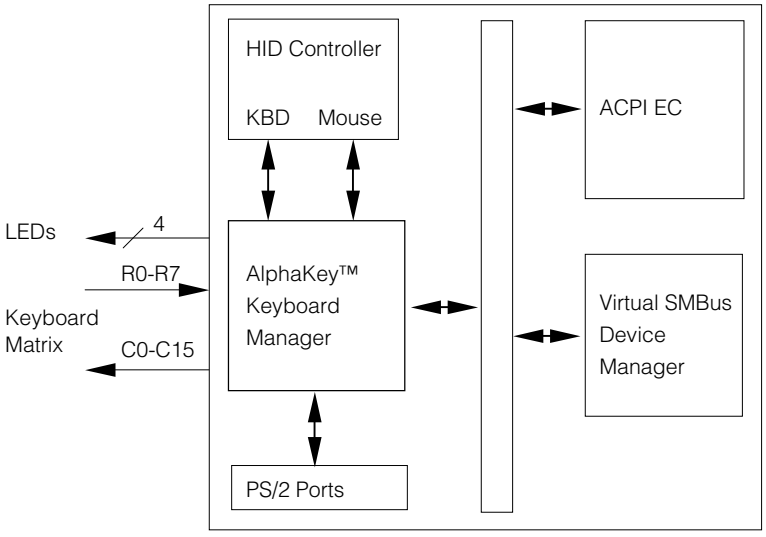
The USAR AlphaKey™ keyboard manager is the most advanced keyboard management module in the industry today and the first one to integrate the laptop keyboard matrix with system management tasks through ACPI and SMBus.

The USAR AlphaKey™ keyboard manager provides OEMs with extreme flexibility both with PS/2 keyboard functionality, as well as with user control, from the keyboard, of system management tasks through ACPI and SMBus.

The USAR AlphaKey™ keyboard manager, as shown in the following diagram, communicates with the HID controller, the ACPI embedded controller, the virtual SMBus device manager and the external PS/2 ports of the USAR ACPITroller™.

AlphaKey™ communications block diagram

The USAR AlphaKey™ keyboard manager can simultaneously support both an external keyboard (including Windows® and Japanese layout keyboards)



and an internal scanned key matrix. The internal scan matrix layout can be programmed through an extended set of keyboard commands. The USAR

AlphaKey™ keyboard manager handles PS/2 keyboard commands, supports external keyboard hot-plug-ins, and merges internal and external keyboard data as if they were coming from one source.

USAR AlphaKey™ features at a glance

- Supports IBM standard 101/102 keyboards including Windows®, On-Now Power keys, Japanese keyboard keys, and Korean keyboard keys.
- External keyboard and internal keyboard operate simultaneously; data is merged
- User can hot-plug external keyboard
- Auto-detection of type of device in any external PS/2 port
- N-key rollover and ghost key detection
- Programmable scan matrix
- Embedded numeric keypad support
- Supports all three scan code sets
- Interoperability between 3-volt systems and 5-volt PS/2 devices without any external level-shifting circuitry
- Unique “Zero-Power™” operation of the scanned matrix AND the PS/2 embedded port
- “Protocol safe” handling of external PS/2 devices

USAR AlphaKey™ hardware configuration

The USAR AlphaKey™ keyboard manager interfaces directly to the signals of the scanned matrix, the external PS/2 ports and the LEDs' port. This section describes the handling of the signal lines and the options OEMs have in configuring the hardware interfaces of the USAR AlphaKey™ keyboard manager.

Scanned matrix

The USAR AlphaKey™ embedded keyboard is organized as an 8 x 16 matrix. While the maximum matrix size and the organization of rows and columns are fixed, smaller matrixes can be accommodated. The USAR AlphaKey™ matrix comes configured with a default key layout described in Appendix B of this document.

OEMs requiring a custom matrix layout are presented with the option to either order the ACPITroller™ delivered with their own matrix specification or use the programming facilities of the USAR ACPITroller™. Configuration of the matrix layout can be accomplished though the USAR Extended Command set of the HID controller (8042), and it is described in detail later in this section. Custom configuration information is stored in the AlphaKey™ RAM and has to be downloaded into the controller each time power is re-cycled. Alternatively, information can be stored permanently in the controller's flash RAM area – if available – or in a EEPROM residing in one of the SMBus ports of the controller.

LED port

The USAR AlphaKey™ keyboard manager supports up to four LEDs. Three LEDs correspond to the standard 101/102 PS/2 keyboard Numeric Lock, Caps Lock and Scroll Lock LEDs. The fourth LED is the Katakana LED used in Japanese keyboards. OEMs have the option either to utilize the LEDs or to disable them in order to use the LED port for other general input/output functions. The configuration of the LED port is in the AlphaKey™ control register, and it is described later in this section.

External PS/2 ports

The USAR AlphaKey™ supports three PS/2 ports, all of them functioning with USAR's patented "Zero-Power™" operation. Each of the three PS/2 ports is available for either a PS/2 keyboard or a PS/2 mouse-type device. The USAR AlphaKey™ keyboard manager detects the hot-plug-in of an external keyboard, and integrates its data input with the input from the embedded keyboard matrix. The USAR AlphaKey™ responds to and implements every PS/2 keyboard command, and individually handles communications with externally connected keyboard devices.

In addition to this mode, the USAR AlphaKey™ can support a pure 8042 type of operation, in which the externally connected keyboard provides the responses to the system-issued PS/2 commands. This mode is implemented in order to support proprietary devices that identify themselves as PS/2 keyboards; however, they require special drivers in order to operate properly. Setting the KLHDIS bit of the control register to one (1) activates the pure 8042 mode. While in this mode, only the first connected external PS/2 keyboard is enabled, in order to avoid conflicting responses from other devices to the custom driver.

Auto-detection of device type in external PS/2 ports

The USAR ACPITroller™ auto-detects the type of PS/2 device connected to either one of the two external PS/2 ports and configures the device properly according to the current system and driver settings. Users can connect any type of PS/2 device (keyboard or mouse) to either port. Data from any keyboard(s) detected are routed to the USAR AlphaKey™ keyboard manager for proper initialization and data handling.

3-volt and 5-volt operation of PS/2 ports

The USAR ACPITroller™ can be powered by either a 3-volt or a 5-volt power supply. Even when powered by a 3-volt power supply, the USAR ACPITroller™ communicates flawlessly with 5-volt powered PS/2 devices, without any additional level-shifting circuit.

PS/2 keyboard protocol command and data handling

The USAR AlphaKey™ fully implements the PS/2 keyboard command protocol including support for scan code set 3 specific commands. Support of scan code set 3 provides compatibility with all operating systems ported to the Intel x86 architecture. The USAR AlphaKey™ keyboard manager, which responds to and executes all PS/2 keyboard.

The table below lists the PS/2 commands supported by AlphaKey™ as well as the corresponding responses to them.

PS/2 command & response codes

Command	Response	Description
FFh	Fah, AAh	Keyboard reset command
FEh	XXh	Resend last byte transmitted
FDh, XXh	Fah, FAh	Set key make (key)
FCh, XXh	Fah, FAh	Set key make/break (key)
FBh, XXh	Fah, FAh	Set key typematic (key)
FAh	Fah	Set all keys typematic/make/break
F9h	Fah	Set all keys make
F8h	Fah	Set all keys make/break
F7h	Fah	Set all keys typematic
F6h	Fah	Set default
F5h	Fah	Disable
F4h	Fah	Enable
F3h , XXh	Fah, FAh	Set typematic delay and rate (value)
F2h	Fah, 8Xh, ABh	Read device ID
F1h	Feh	Invalid command
F0h, 00 – 03	Fah, FAh, (0Xh)	Set scan set (value) 0 = Programmable keyboard matrix
EFh	Feh	Invalid command
EEh	Eeh	Echo
EDh, 0Xh	Fah, FAh	Set LEDs (value)

Protocol Safe™ handling of external PS/2 keyboards

The USAR AlphaKey™ keyboard manager provides a level of protocol isolation between the external PS/2 keyboard(s) and the host system. Even when an external keyboard is connected, commands issued by the system are responded to and executed internally within the USAR AlphaKey™ keyboard manager. The USAR AlphaKey™ keyboard manager subsequently relays only relevant commands to the external device(s) and it processes locally any hand-shaking, including errors and recovery messages.

This unique mode of operation provides the system with two benefits:

- It isolates the system from bad implementations of the PS/2 keyboard protocol from external keyboards. The USAR AlphaKey™ directs to the external keyboard only the simplest commands required for data entry and LED handling and it handles internally the more complicated commands. It basically uses the external device the same way it handles the scanned matrix for pure key entry and it takes full responsibility of the protocol implementation. The system “sees” a consistent PS/2 keyboard protocol implementation independently of the PS/2 keyboard the user has connected to the external PS/2 ports.
- It improves the system performance. Unlike keyboard controller implementations that rely on the external device to provide command responses, the USAR AlphaKey™ communicates quickly and efficiently through the X-bus, and hides from the system the overhead of low level PS/2 error handling.

USAR AlphaKey™ Scan Code support

USAR’s AlphaKey™ supports the IBM Standard Scan Code Sets 1, 2, and 3 for all keys, including Windows® and the On-Now Power keys (Scan Code 1 & 2 only).

USAR AlphaKey™ keyboard matrix

The USAR AlphaKey™ implements three distinct key layouts over the same keyboard matrix. Each keyboard matrix layout can be viewed as a separate keyboard selected and activated by the user through the Fn and Num Lock key. The default layout is QWERTY; this is the most commonly used layout, and it includes all keys needed for regular data entry. The NumPad layout can be invoked by setting the Num Lock LED on, and it implements a numeric keypad over part of the matrix for fast numeric entry. The Fn layout is invoked when the user presses the Fn key of the laptop keyboard, and it is used to activate special keys for system functions and custom data functions. The USAR AlphaKey™ is preprogrammed with a typical laptop keyboard matrix, implementing all three layouts. The preprogrammed matrix and layouts are described in Appendix B of this document.

Keyboard matrix layouts

The USAR AlphaKey™ keyboard manager maintains the keyboard matrix layout information in the controller's Matrix RAM area which is described later in this chapter. After a power-on reset, the USAR AlphaKey™ defaults to Scan Set 1. Subsequently, the OEM can download a custom matrix and/or custom key definitions and assignments through BIOS, TSR programs or device driver software, and select Scan Set 0 to enable the custom matrix.

The following table describes the matrix layouts supported by the USAR AlphaKey™.

USAR AlphaKey™ matrix layouts

Fn Key	Num Lock	Description
Up	Off	QWERTY layout
Up	On	NumPad layout
Down	X	Fn layout

USAR AlphaKey™ key numbers

The action invoked by a key press is determined primarily by the USAR key number assigned to the specific matrix location.

Each matrix location has a USAR key number assigned to it. If a matrix location is empty (no physical switch), or if the switch should produce no action in specific layouts, the null key number can be assigned into the specific position.

USAR has defined four distinct ranges of key numbers. The following table describes the USAR defined key number ranges for keys supported by the AlphaKey™ keyboard manager.

USAR AlphaKey™ key number ranges

Key numbers (hex)	No of keys	Category
00 - 7F	128	Standard PS/2 keys
80 - AF	48	Alternate layout keys (NumPad and Fn layouts)

A description of the codes in each of these ranges follows.

Standard PS/2 keys

USAR AlphaKey™ uses 128 unique numbers 00-7Fh (0-127) to describe the Standard PS/2 keys implemented in most standard PS/2 keyboard layouts. The key numbers for Standard PS/2 keys are fixed and they are re-used. Note the following special key ranges:

0	Null key
1 - 101	101/102 Keyboard keys
102 – 104	Windows keys.
105	Overrun error
106	Function (Fn) key
107	Sticky mode key. This key is used to toggle the “sticky” mode of operation of the PS/2 shift keys (Ctrl, Shift, Alt, Win Keys)
108 – 110	On-now Power keys.
111 – 115	Japanese keyboard key numbers.
	Korean keyboard key numbers.
124 – 132	Extended function keys F13 – F24

A list of the USAR Standard PS/2 key numbers appears in Appendix A of this document.

Alternate layouts keys (NumPad and Fn layouts)

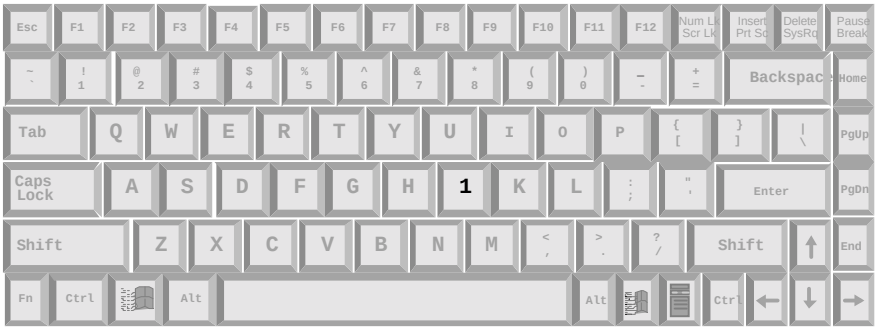
The USAR AlphaKey™ keyboard manager uses 48 distinct codes (80h – Afh) to describe a special range of keys that, in addition to their default QWERTY values, implement the NumPad and Fn layout translations.

Each one of the “alternate layouts” keys has three key numbers associated with them which are programmable and kept in the Matrix RAM area of the controller.

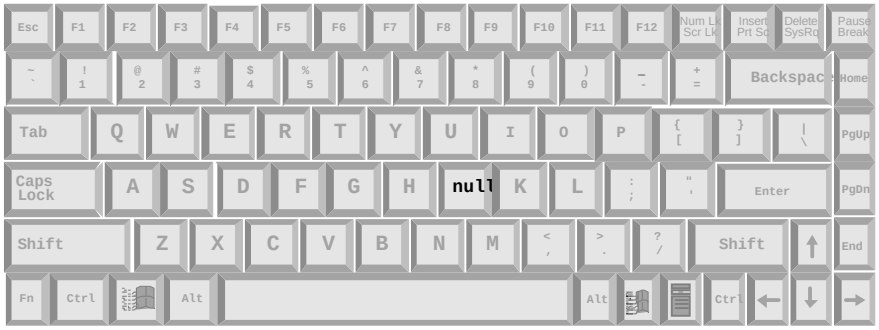
The output of an “alternate layouts” key is determined by the active matrix layout (QWERTY, NumPad or Fn). OEMs can define the output for each layout to be a Standard PS/2 key.

The following drawings illustrate an example of an “alternate layouts” key.

Example 1: “Alternate layouts” key number DBh acts as the “J” when the QWERTY layout is active.



Example 2: The same key acts as the numeric keypad key “1”, when the NumPad layout is selected.



Example 3: When the Fn layout is active, the key acts as a “null” key.

The following table lists the default values of the “alternate layouts” keys. For each “alternate layouts” key number, three matrix RAM locations exist to store its key number values for each matrix layout. The OEM can program these RAM locations, using the “extended protocol” commands.

Alternate layouts key numbers default values

	QWERTY layout			NumPad layout			Fn layout		
Alt. Layout key # (Hex)	**Matrix RAM offset (Hex)**	**USAR key # (Hex)**	**Key label**	**Matrix RAM offset**	**USAR key # (Hex)**	**Key label**	**Matrix RAM offset (Hex)**	**USAR key # (Hex)**	**Key label**
80	80	2F	M	A0	4A	0 / INS	C0	2F	M
81	81	23	J	A1	41	1 / END	C1	23	J
82	82	24	K	A2	42	2 / down arrow	C2	24	K
83	83	25	L	A3	43	3 / PgDn	C3	25	L
D4	84	16	U	A4	44	4 / left arrow	C4	16	U
85	85	17	I	A5	45	5	C5	17	I
86	86	18	O	A6	46	6 / right arrow	C6	18	O
87	87	08	7 / &	A7	47	7 / home	C7	08	7 / &
88	88	09	8 / *	A8	48	8 / up arrow	C8	09	8 / *
89	89	0A	9 / (	A9	49	9 / PgUp	C9	0A	9 / (
8A	8A	19	P	AA	4C	Num dash	CA	19	P
8B	8B	26	/:	AB	40	N.+	CB	26	/:
8C	8C	0B	0 / }	AC	3F	*	CC	0B	0 / }
8D	8D	32	/ **/** ?	AD	4E	/	CD	32	/ **/** ?
8E	8E	31	. / >	AE	4B	. / DEL	CE	31	. / >
8F	8F	3A	up arrow	AF	3A	up arrow	CF	3C	PgUp
90	90	3B	down arrow	B0	3B	down arrow	D0	3D	PgDn
91	91	37	left arrow	B1	37	left arrow	D1	39	END
92	91	3E	right arrow	B2	3E	right arrow	D2	38	HOME
93	91	63	Scrl lock	B3	63	Scrl lock	D3	62	NumLk
94	91	35	INS	B4	35	INS	D4	64	PrtScr
95	91	36	DELr	B5	36	DELr	D5	7E	SysReq
96-AF	92-AF	0	Null	B6-CF	0	Null	D6-EF	0	Null

Programming the keyboard matrix

The keyboard matrix is a contiguous section of ACPITroller™ memory divided into four subsections. You program the keyboard matrix by downloading the desired values into the matrix memory. You do this with the four download commands, BA, BB, BC, and BD, which are part of the HIDC USAR expanded command set. Chapter 2, the HID controller interface chapter, explains the role of each command and how to issue it.

The first of the four sections of the keyboard matrix is 128 bytes long. It maps the row and column coordinates generated by a keyboard into USAR numbers, described above. The section is organized as follows:

Byte Offset	Function
0	USAR number for Row 0, Column 0
1	USAR number for Row 1, Column 0
2	USAR number for Row 2, Column 0
3	USAR number for Row 3, Column 0
4	USAR number for Row 4, Column 0
5	USAR number for Row 5, Column 0
6	USAR number for Row 6, Column 0
7	USAR number for Row 7, Column 0
8	USAR number for Row 0, Column 1
.	.
.	.
.	.
124	USAR number for Row 4, Column 15
125	USAR number for Row 5, Column 15
126	USAR number for Row 6, Column 15
127	USAR number for Row 7, Column 15

USAR numbers 0-7fh correspond to the standard PS/2 key numbers and are used for keys that are unaffected by the Numeric Lock and Function keys.

Key numbers 80h to 9fh are reserved for those keys that are influenced by the Numeric Lock and Function keys. Row and column coordinates that are assigned these numbers then generate a second code lookup from one of sections 2, 3, or 4.

Each of the remaining three sections of the keyboard matrix is 32 bytes long and contains the USAR number generated by a keypress in one of three states:

1. No control keys selected

2. Numeric Lock selected

3. Function selected

For example, on the Fujitsu 7654 keyboard, Key J is at Row 7, Column 8 and is a part of the integrated numeric keypad (number 2). When the keyboard is in Numeric Lock, the J key generates the keycode for Keypad 2. So, at offset 47h of matrix section one (which corresponds to Row 7, Column 8), we assign the value 8ah, which is one of the reserved USAR numbers. Matrix section two contains the keycode to be generated when no control keys are pressed, and the offset into this table is calculated by subtracting 128 from the value we put into matrix section one. Therefore, the offset into matrix section two is 10, which is where we put the value 23h, which is the standard keycode generated by the J key.

If Numeric Lock was selected, then the offset is used to lookup in matrix section three, which would contain the keycode 41h, which is Numeric 2.

If Function was selected, then the offset is used to lookup in matrix section four, which would contain 23h, since the J key is not affected by the Function key.

The four sections comprise Page 0 of the matrix RAM. Matrix section one is at offset 0, section two is at offset 80h, section three is at offset A0h, and section four is at offset C0h.

USAR UR8HC342 ACPITroller™ Basic AlphaMouse™ Pointing Devices Manager

Overview

In legacy mode, the UR8HC342 USAR AlphaMouse™ pointing devices manager can simultaneously support up to three standard mice connected to both the external and internal PS/2 ports of the ACPITroller™. The USAR AlphaMouse™ handles PS/2 mouse commands, supports external mouse hot-plug-ins, and merges internal and external mouse data.

In multiplexing mode, operating with a multiplexing host driver, UR8HC342 USAR AlphaMouse™ pointing devices manager can simultaneously support up to three pointing devices connected to both the external and internal PS/2 ports of the ACPITroller™. These pointing devices can include proprietary and non-standard devices, depending on the driver(s). The USAR AlphaMouse™ does not merge the data, but keeps data reports from separate pointing devices separate for handling by the multiplexing host driver.

Features at a glance

- AlphaMouse™ pointing device manager supports hot-plugging and hot-swapping of standard two-button and three-button mice without a special driver; it also supports hot-plugging and hot-swapping of MouseWheel mice with a MouseWheel-capable driver
- Two external PS/2 ports for external keyboard and mouse with auto-detect and hot-plug support
- Simultaneous operation of external and internal pointing devices, merging internal and external mouse data
- Support of MouseWheel functionality for both embedded and external pointing devices, even with hot-plug connections, with a MouseWheel-capable driver
- Operates safely with PS/2 mouse protocol
- AlphaMouse™ performs active PS/2 multiplexing of input from different types of mice simultaneously, with the appropriate driver(s)
- Support of the USAR ScreenCoder™ PS/2 absolute and relative touchscreen encoder, with the appropriate driver
- Support of the five-button mouse, with an appropriate driver

Multiplex mode

A host driver that supports active PS/2 multiplexing sends a special sequence of commands to the USAR UR8HC342 ACPITroller™ as follows:

The host writes 0xD3 to output port 0x64

The host writes 0xF0 to output port 0x60

The controller responds with 0xF0 on input port 0x60

The host writes 0xD3 to output port 0x64

The host writes 0x56 to output port 0x60

The controller responds with 0x56 on input port 0x60

The host writes 0xD3 to output port 0x64

The host writes 0xA4 to output port 0x60

The controller responds with 0x11 on input port 0x60

By responding to last part of the sequence with 0x11 instead of 0xA4, the ACPITroller™ shows that it supports active PS/2 multiplexing standard, version 1.1. At this point, the ACPITroller™ AlphaMouse enters multiplex mode.

While in multiplex mode, the ACPITroller™ AlphaMouse behaves as follows:

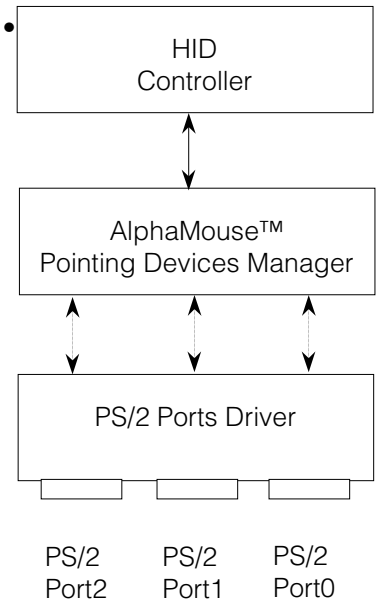
When it receives a data report from a pointing device on a PS/2 port, it adds a prefix to the data report indicating which PS/2 port, and passes the report with the prefix to the multiplexing driver.

When it receives a command from the multiplexing driver, it strips the prefix from the command, and sends the command to the PS/2 port indicated by the prefix.

Legacy Mode

Functional description and handling of PS/2 ports

The USAR AlphaMouse™ pointing devices manager receives its pointing device input through one or more of the USAR ACPITroller™ Basic PS/2 ports. The USAR AlphaMouse™ can handle up to three pointing devices. The PS/2 Ports Driver auto-detects the type of device connected to each PS/2 port. If the device reports itself as a Pointing Device, PS/2 Ports Driver connects it to the USAR AlphaMouse™ pointing devices manager for proper initialization and further data and command handling. The USAR AlphaMouse™ pointing devices manager communicates with the host system through the mouse port of the HID controller.

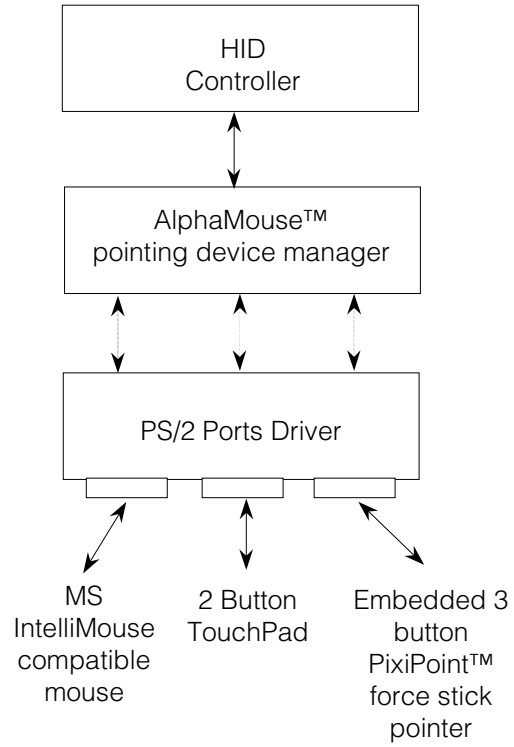


USAR AlphaMouse™ block diagram

Local handling mode

In legacy mode, the USAR AlphaMouse™ supports two distinct modes of pointing devices operation; the “Local handling” and the “System handling” modes. In the “Local handling“ mode, the USAR AlphaMouse™ responds locally to every command issued by the driver, including configuration commands for MouseWheel support. The AlphaMouse™ initializes up to three PS/2 pointing devices and handles all configuration and data transactions individually. If an IntelliMouse® device is hot-plugged in, it is configured by the USAR AlphaMouse™ to enable MouseWheel data reporting. X, Y and Z data reported by all the connected pointing devices is both accumulated and reported to the host driver as if they originated from a single PS/2 device.

This default mode of operation supports all known PS/2 pointing devices, providing the least system intervention and the maximum flexibility in the usage of external PS/2 devices with the USAR ACPITroller™ Basic. The following drawing illustrates a possible configuration that can be supported by the USAR AlphaMouse™ pointing devices manager.



USAR AlphaMouse™ sample user model

In this example, a pointing device is connected to each of the three AlphaMouse™ PS/2 ports, one internal and two external A USAR PixiPoint™ force-stick type of embedded mouse, emulating a three-button mouse, is connected to the internal PS/2 port. A two-button TouchPad is connected to one of the external PS/2 ports and a Microsoft® IntelliMouse® to the other. In our example, we assume that the laptop is loaded with Windows 98 and the Microsoft® IntelliMouse® driver. In this configuration, the USAR ACPITroller™ Basic will operate as described below.

- The PS/2 Ports Driver detects the presence of each pointing device and notifies the pointing device manager of their presence.
- The system mouse driver interrogates the ACPITroller™ Basic about its capabilities. The USAR AlphaMouse™ detects the MouseWheel initialization sequence issued by the driver and reports it to the system driver as an IntelliMouse® compatible mouse, thus changing its report scheme from 3 to 4 bytes.
- The USAR AlphaMouse™ pointing device manager interrogates each connected pointing device about its capabilities. The AlphaMouse™ initializes the Microsoft IntelliMouse® to report MouseWheel data.
- The USAR AlphaMouse™ collects X, Y and Z (where available) data and composes a single report to present to the driver. The PixiPoint™ and the TouchPad report only X and Y coordinates.
- Any change in the configuration, such as the hot-plug-in of another pointing device, is handled internally by the USAR AlphaMouse™ and the transaction details are hidden from the driver.

PS/2 Mouse Commands

The USAR AlphaMouse™ internally handles all PS/2 mouse commands. These commands, as well as the responses to them, are listed in the following table.

USAR AlphaMouse™ PS/2 commands

Command	Response	Description
FFh	FAh, AAh, 00h	Mouse reset command
FEh	XXh	Resends last package
F7h – FDh	FEh	Invalid
F6h	FAh	Set default
F5h	FAh	Disable
F4h	FAh	Enable
F3h , XXh	FAh, FAh	Sampling rate
F2h	FAh, 00h	Read device ID
F1h	Feh	Invalid command
F0h	FAh	Set remote mode
EFh	FEh	Invalid command
EEh	FAh	Set wrap mode
EDh	FEh	Invalid command
ECh	FAh	Reset wrap mode
EBh	FAh, XXh, XXh, XXh,	Read data
EAh	FAh	Set stream mode
E9h	FAh, XXh, XXh, XXh	Status request
E8h, 0Xh	FAh, FAh	Set resolution
E7h	FAh	Set scaling 2:1
E6h	FAh	Reset scaling 1:1

External Mouse Hot-Plug Support

The USAR AlphaMouse™ pointing devices manager supports the hot-plug in of external mice. The pointing devices manager initializes and configures each connected device dynamically, according to the system mouse driver settings. The USAR AlphaMouse™ pointing devices manager auto-detects three-button mice, as well as MouseWheel type of devices, and enables every feature that the system driver supports.

Transparent MouseWheel Support

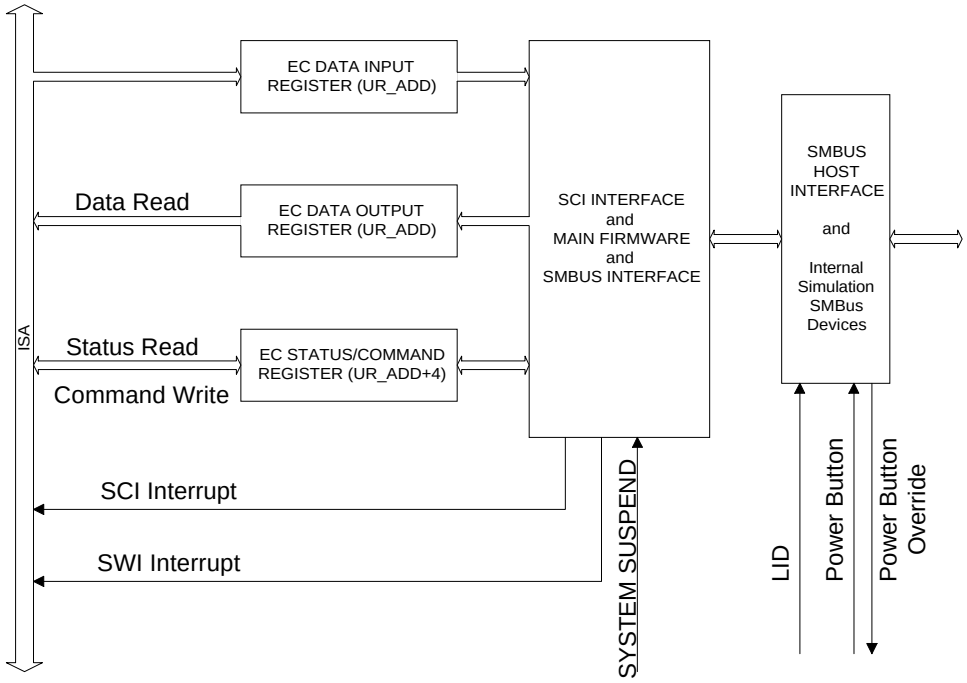
The USAR AlphaMouse™ supports MouseWheel reporting. The USAR AlphaMouse™ pointing device manager generates a special initial sequence to the host in order to alert the driver to utilize the MouseWheel report format.

The MouseWheel has a different report packet than a standard PS/2 mouse. As a result, the two cannot be integrated. The USAR AlphaMouse™ pointing device manager monitors the connected mice. If the system mouse driver is initialized to work in MouseWheel mode, the USAR AlphaMouse™ alters its reports to correspond to the MouseWheel format. In this manner, both the internal and external mice data can be integrated.

USAR UR8HC342 ACPITroller™ Embedded Controller Interface

UR8HC342 EC bus interface

The USAR UR8HC342 ACPI embedded controller (EC) interfaces to the host ISA bus via two I/O addresses. The following diagram illustrates the embedded controller architecture that includes a dedicated ACPI Interface.



Notes: UR_ADD represents the decoded base address for the UR8HC342 Embedded Controller ISA Ports. Default value is 0x62.

UR8HC342 embedded controller registers

The USAR UR8HC342 SMBus EC contains three registers that occupy two I/O locations. The registers are listed in the following table.

USAR UR8HC342 Registers

Name	Description	I/O function	I/O Port Address
EC_SC (R)	Status	IOR	UR_ADD+4
EC_SC (W)	Command	IOW	UR_ADD+4
EC_DATA	Data	IOR/IOW	UR_ADD

EC Status Register, EC_SC (R)

EC_SC (R) is a read-only register that indicates the current status of the Embedded Controller interface. It contains the following fields.

EC_SC (R) EC status register bit definitions

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IGN	SMI_EVT	SCI_EVT	BURST	CMD	IGN	IBF	OBF

EC_SC (R) EC status register bit definitions

IGN	Ignored
SMI_EVT	SMI Event 1 – Indicates SMI event is pending (requesting SMI query) 0 – Indicates no SMI event pending
SCI_EVT	SCI Event 1 – Indicates SCI event is pending (requesting SCI query) 0 – Indicates no SCI events are pending
BURST	Burst Mode 1 – Controller is in burst mode for polled command processing 0 – Controller is in normal mode for interrupt-driven command processing
CMD	Command/Data 1 – Byte in data register is a command byte (only set by host, used by EC) 0 – Byte in data register is a data byte (only clear by host, used by EC)
IBF	Input Buffer Full 1 – Input buffer is full (data ready for EC) 0 – Input buffer is empty, generate SCI interrupt
OBF	Output Buffer Full 1 – Output buffer is full (data ready for host), generate SCI interrupt 0 – Output buffer is empty

Command Register, EC_SC (W)

EC_SC (W) is a write-only register that allows commands to be issued to the embedded controller. Writes to this register are latched in the input data register and the input buffer full (IBF) flag is set in the status register. Writes to this location also cause the command bit to be set in the status register. This enables the embedded controller to differentiate the start of a command sequence from a data byte write operation.

Data Register, EC_DATA(R/W)

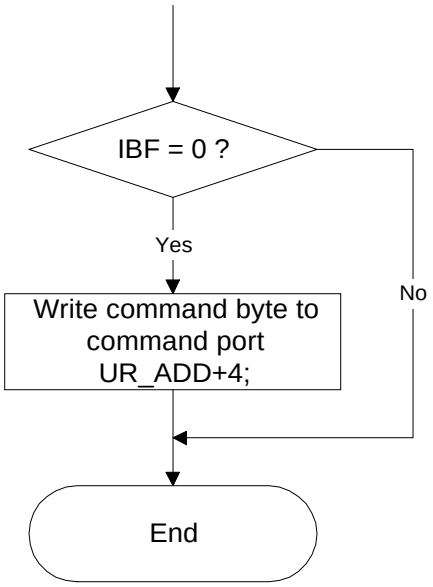
EC_DATA(R/W) is a read/write register that allows both command/data bytes to be issued to the embedded controller and the OS to read data returned by the embedded controller. Writes to this port by the host are latched into the input data register and the input buffer full (IBF) flag is set in the status register. Reads from this register return data from the output data register and clear the output buffer full (OBF) flag in the status register.

Port Operation

The following diagrams illustrate the sequence for issuing commands and data to the EC through the host interface.

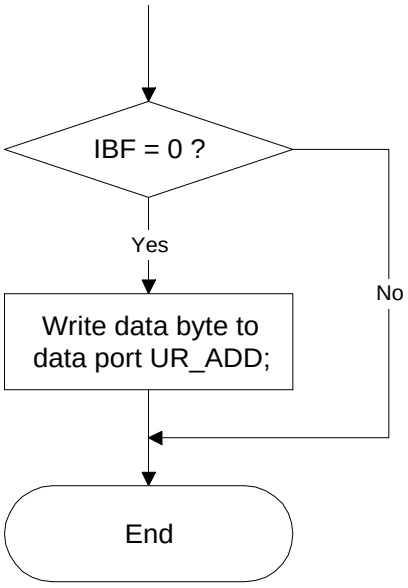
Send Command Byte

The Host can issue a command byte to command register directly.



Send Data Byte

The Host can send a data byte to the data port directly.



EC Commands

The Operating System can communicate with the EC using the standard ACPI embedded controller command set. These commands are listed and detailed below.

USAR UR8HC342 ACPI controller command set

Embedded Controller Command	Command Byte Encoding
Read Embedded Controller (RD_EC)	0x80
Write Embedded Controller (WR_EC)	0x81
Burst Enable Embedded Controller (BE_EC)	0x82
Burst Disable Embedded Controller (BD_EC)	0x83
Query Embedded Controller (QR_EC)	0x84

Read Embedded Controller, RD_EC (0x80)

The RD_EC (0x80) command allows the OS to read a byte in the address space of the embedded controller. The exact command sequence is detailed below. This command is reserved exclusively for use by the OS and proceeds based on interrupts from the EC. The values of the IBF and OBF flags determine the interrupt generation as follows.

RD_EC (0x80) command sequence

Step	Action	Register	Port Address	SCI Interrupt
1	Send command header	Command port	UR_ADD+4	Interrupt on IBF=0, command header has been read
2	Send address to be read	Data	UR_ADD	No interrupt
3	Host read data	Data	UR_ADD	Interrupt on OBF=1, data is available

Write Embedded Controller, WR_EC (0x81)

The WR_EC (0x81) command byte allows the OS to write a byte in the address space of the embedded controller. This command byte is reserved exclusively for use by the OS and proceeds based on interrupts from the EC. The value of the IBF flag determines the interrupt generation as follows.

WR_EC (0x81) command sequence

Step	Action	Register	Port Address	SCI Interrupt
1	Send command header	Command port	UR_ADD+4	Interrupt on IBF=0, command header has been read
2	Send address to be written	Data	UR_ADD	Interrupt on IBF=0, address has been read
3	Host write data	Data	UR_ADD	Interrupt on IBF=0, data has been read

Burst Enable Embedded Controller, BE_EC (0x82)

The BE_EC (0x82) command allows the OS to request dedicated attention from the EC and, except for critical events, the EC from doing tasks other than receiving command and data from the OS. This command is an optimization that allows the host processor to issue several commands back-to-back, in order to reduce latency at the embedded controller interface. When the controller is in the burst mode, it should transition to the burst Disable State if the host does not issue a command within the following guidelines:

- First Access 400 microseconds
- Subsequent Accesses 50 microseconds each
- Total Burst Time 1 millisecond

If the EC disables burst mode for any reason other than the burst disable command, it generates an SCI to the OS to indicate the change.

While in burst mode, the embedded controller follows these guidelines for the OS driver:

- SCIs are generated as normal, including IBF=0 and OBF=1.
- Accesses should be responded to within 50 microseconds.

Following is the burst enable command sequence.

BE_EC (0x82) command sequence

Step	Action	Register	Port Address	SCI Interrupt
1	Send command header	Command port	UR_ADD+4	No interrupt
2	Host read burst ACK byte	Data	UR_ADD	Interrupt on OBF=1, data is available

Burst Disable Embedded Controller, BD_EC (0x83)

This command releases the embedded controller from a previous Burst Enable command and allows it to resume normal processing. The OS sends this command after it has completed its entire queued command sequence to the embedded controller.

Following is the burst disable command sequences.

BD_EC (0x83) command sequence

Step	Action	Register	Port Address	SCI Interrupt
1	Send command header	Command port	UR_ADD+4	Interrupt on IBF=0, command header has been read

Query Embedded Controller, QR_EC (0x84)

The OS driver sends the QR_EC (0x84) command when the SCI_EVT flag in the EC_SC register is set. When the EC has detected a system event that must be communicated to the OS, it first sets the SCI_EVT flag in the EC_SC register, generates an SCI, and then waits for the OS to send the query (QR_EC) command. The OS detects the embedded controller SCI, sees the SCI_EVT flag set, and sends the query command to the embedded controller. Upon receipt of the QR_EC command byte, the EC places a notification byte with a value between 0-255, indicating the cause of the notification. The notification byte indicates which interrupt handler operation the OS should execute in order to process the embedded controller SCI. The query value of zero is reserved for a spurious query result and indicates “no outstanding event.”

Following is the query EC command sequences.

QR_EC (0x84) command sequence

Step	Action	Register	Port Address	SCI Interrupt
1	Send command header	Command port	UR_ADD+4	No interrupt
2	Host read query value	Data	UR_ADD	Interrupt on OBF=1, data is available

SMBus Host Controller Notification Header (Optional), OS_SMB_EVT

The OS_SMB_EVT query command notification header is the special return code that indicates events from an SMBus controller implemented within an embedded controller. These events include the following.

OS_SMB_EVT events

Num	Notification	Event	Description
1	00h	No outstanding event	
2	01h	Command complete	Response to OS SMBus command
3	02h	Command error	Response to OS SMBus command
4	03h	Alarm	Alarm from SMBus device

The actual notification value is declared in the SMBus host controller device object in the ACPI name space. For more details on the SMBus Host functioning in the ACPI system, refer to the SMBus Host Controller Interface section of this Specification.

This page intentionally left blank

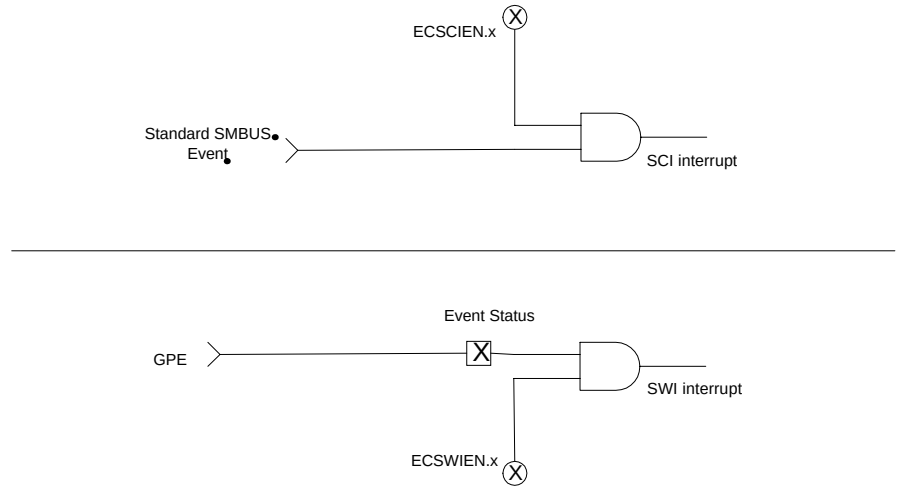
USAR UR8HC342 ACPITroller™ SCI & SWI Interrupt Generation

Event interrupts

The USAR UR8HC342 ACPITroller™ generates two types of interrupts, the SCI and the SWI. The type of interrupt generated depends on the type of event that caused it. When a standard SMBus event occurs, the EC generates an SCI interrupt. For any other General Purpose Events (GPE), an SWI interrupt occurs. Such events may include lid events, dock events, or power button presses.

SWI interrupts have the ability to wake up the system if it is in suspend. SCI interrupts do not do this.

Below is a diagram describing the generation of these two interrupts.



Generation of SCI and SWI event interrupts

SWI sources

There are a number of events that trigger an SWI interrupt: up to eight external events can trigger a SWI interrupt; in addition, the USAR UR8HC342 internally generates an SWI in response to certain SMBus conditions.

Eight pins, labeled GPE0 – GPE7, can be connected to event sources to generate SWI interrupts. Each of these pins has specific traits, making some more suitable than others for certain functions. Pins GPE0 and GPE1 are able to generate GP events on detection of positive- or negative-edge signals. This makes them most suitable for handling occasional events like docking insertion/removal, and lid open/close, where processing is required only when a change in input is detected.

GPE2 to GPE7 trigger on logical low signal level, rather than signal edge. This makes them more suitable for functions like Power Button Override, which requires the Power Button to be held down for four seconds.

GPE4 and GPE7 have no internal pullup resistors and require external pullup to be used. GPE3-6 can enable internal pullup only when SMBus2 is disabled.

In order to use these pins as GPE input pins, the user must enable them as GPE pins in the GPECFG register. If these pins are not used as GPE pins, they are available as General I/O pins.

In addition to these externally generated GPE events, the USAR UR8HC342 generates a SWI interrupt in response to certain SMBus conditions. The EC monitors the SMBus status. If the SMBus becomes stuck, the EC can generate a SWI to the system to report this. If a generic I^2C device (not identified as a smart battery, charger, or selector) generates an alert, the EC can generate an SWI interrupt. In addition, under certain conditions, battery alarms and selector and charger alerts can generate SWI interrupts, as described below.

Special SMBus events

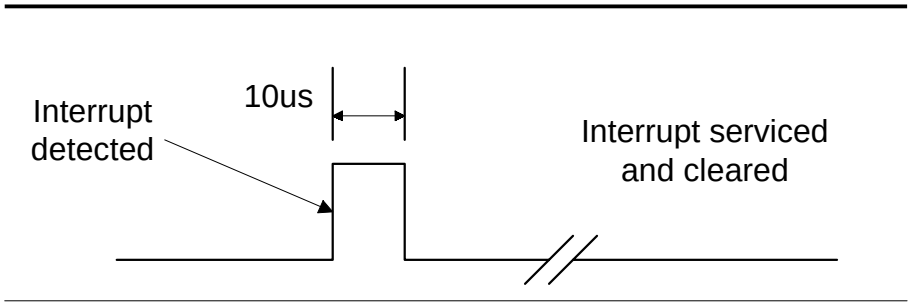
One of the differences between SCI and SWI events is that an SWI interrupt can wake up a host in suspend mode, while an SCI interrupt cannot.

As described previously, standard SMBus events generate SCI interrupts. An SCI interrupt does not wake up the system if it is in suspend mode. Some SMBus events, however, may occur when the system is in suspend, and need to wake up the system. These events are battery alarm and selector and charger alert. For these events, if the system is in suspend, the UR8HC342 first simulates a power button press. This power button press generates an SWI interrupt, which wakes up the system. Once the system is active again, the regular SCI interrupt can be generated to indicate the SMBus event.

Interrupt Generation

The EC Interrupt Model uses pulsed interrupts to speed the clearing process. The SCI interrupts are firmware-generated, using EC general-purpose outputs, and have the waveform shown in the figure below. Figure 13-3.

The OS treats these interrupts as edge events.



EC interrupt process waveform

USAR UR8HC342 ACPITroller™ SMBus Host Controller Interface

Overview

This chapter describes the System Management Bus (SMBus) host controller interface. The SMBus address space is a generic address space defined in the ACPI specification. Following is a description of the USAR UR8HC342 ACPITroller™ implementation of the SMBus Host Controller within the Embedded Controller. This implementation allows the OS directly to address devices on the SMBus.

SMBus overview

The SMBus is a two-wire interface based on I^2C protocol. A low-speed bus, it provides multi-device addressing as well as bus arbitration. For more information on the SMBus generally, refer to the complete set of SMBus Specifications published by the Smart Battery System Implementers Forum (SBS-IF) at http://www.sbs-forum.org/.

The SMBus host controller interface

The SMBus Host Controller interface allows the host processor, under control of the OS, to manage devices on the SMBus. Among typical devices that reside on the SMBus are smart batteries, smart chargers, contrast/backlight control, and temperature sensors. Due to the sensitive nature of some of these devices, the SMBus Host Controller is required to monitor and filter certain SMBus commands addressed to the Smart Battery System. This particular function of the EC is essential for the system's safety, since it prevents errant applications or viruses from creating system hazards through improper control of the battery subsystem.

The SMBus Host Controller interface provides a method of communicating with SMBus devices through a block of registers that reside in the Embedded Controller space. In addition, the SMBus Host Controller handles certain SMBus functions related to alerts and error conditions.

The USAR UR8HC342 ACPITroller™ supports the standard set of registers defined in the ACPI specification that an ACPI-compatible OS can use to communicate with SMBus devices.

The following sections detail the SMBus Host Controller functionality, the Register Interface, and configuration parameters.

SMBus alarm message & SMBus alert process

In an SMBus system, several devices can be programmed to notify the host system when specific events occur. A Smart Battery, for example, issues an alarm message to signal a critical capacity condition. Other SMBus devices, including Smart Selectors and Smart Chargers, issue alerts to inform the SMBus Host Controller of an internal status change. SMBus devices typically notify the SMBus Host Controller that such an event has occurred by asserting the SMBus alert signal.

When the USAR UR8HC342 ACPITroller™ SMBus Host Controller detects that the SMBus alert signal has been asserted, it first checks the CHGQREN and SLCTQREN bits in R0 (EC configuration register 0) in order to determine whether to query for a Smart Charger- or a Smart Selector-generated alert. If either one of these bits is set, the SMBus Host Controller queries the appropriate device and checks its status. The status is then ANDed with the appropriate masks and, if the result is nonzero, the appropriate device status message is placed into the EC SMBus Alarm buffer and an SCI is generated. The address used to query the selector and charger depends on the system configuration. If the system includes a selector and a charger at different addresses, the queries are made to the addresses specified. If the system includes a selector and charger device residing at the same address, the query is to that specific address. This configuration controlled by the CHGSLCT bit in R0.

If the alert did not come from one of these two devices, the alert is considered to come from a generic I^2C device. The alert can generate an SWI interrupt, not an SCI interrupt. The SMBus host generates an SWI interrupt if the DEVICEEN flag in EC_SWIEN is set.

If querying for the selector is disabled (SLCTQREN is clear), alerts from the selector are treated the same as for any other I^2C devices. Similarly, if querying for the charger is disabled (CHGQREN is clear), alerts from the charger are treated in the same manner as any other I^2C device.

In addition to SMBus devices generating alerts, the Smart Battery can generate an alarm to indicate a critical condition. When the USAR UR8HC342 ACPITroller™ receives a Smart Battery alarm, it retrieves the alarm message and ANDs it with the contents of R14 (the EC battery alarm mask register). If the result is nonzero, the EC puts an alarm message into the EC SMBus Alarm buffer and generates an SCI interrupt to the system.

Note that once an alarm message has been received, the SMBus host controller does not receive additional alarm messages until the ALRM status bit is cleared.

SMBus error recovery

The USAR UR8HC342 ACPITroller™ monitors the SMBus. If it detects that the SMBus has gotten stuck, the EC tries to recover the bus. If it fails and the SMBus Stuck event is enabled (by setting STUCKEN in register R05), the EC issues an SWI interrupt to the system.

SMBus host register space

The SMBus host interface is a flat array of registers that are arranged in a block of system address space. The SMBus Register Space base address (SMB_BASE) corresponds to zero (0) in the USAR UR8HC342's address space.

SMBus host registers

The table below lists the registers defined for the SMBus Host with their reset values. Each register is eight bits wide.

The first group of registers contains the standard SMBus Host registers, as listed in the ACPI Specification section on Embedded Controllers. A list of these registers follows. An appendix to this document provides brief descriptions of these registers. Full descriptions are given in the ACPI specification.

The method of initiating the different protocols on the SMBus through these registers is also briefly described in the appendix, and fully provided in the ACPI specification.

Standard SMBus host registers

Location	Register Name	Reset Default	Description
SMB_BASE+0	SMB_PRTCL	00000000	Protocol register
SMB_BASE+1	SMB_STS	00000000	Status register
SMB_BASE+2	SMB_ADDR	00000000	Address register
SMB_BASE+3	SMB_CMD	00000000	Command register
SMB_BASE+4	SMB_DATA[0]	00000000	Data register zero
SMB_BASE+5	SMB_DATA[1]	00000000	Data register one
SMB_BASE+6	SMB_DATA[2]	00000000	Data register two
SMB_BASE+7	SMB_DATA[3]	00000000	Data register three
SMB_BASE+8	SMB_DATA[4]	00000000	Data register four
SMB_BASE+9	SMB_DATA[5]	00000000	Data register five
SMB_BASE+10	SMB_DATA[6]	00000000	Data register six
SMB_BASE+11	SMB_DATA[7]	00000000	Data register seven
SMB_BASE+12	SMB_DATA[8]	00000000	Data register eight
SMB_BASE+13	SMB_DATA[9]	00000000	Data register nine
SMB_BASE+14	SMB_DATA[10]	00000000	Data register ten
SMB_BASE+15	SMB_DATA[11]	00000000	Data register eleven
SMB_BASE+16	SMB_DATA[12]	00000000	Data register twelve
SMB_BASE+17	SMB_DATA[13]	00000000	Data register thirteen
SMB_BASE+18	SMB_DATA[14]	00000000	Data register fourteen
SMB_BASE+19	SMB_DATA[15]	00000000	Data register fifteen
SMB_BASE+20	SMB_DATA[16]	00000000	Data register sixteen
SMB_BASE+21	SMB_DATA[17]	00000000	Data register seventeen
SMB_BASE+22	SMB_DATA[18]	00000000	Data register eighteen
SMB_BASE+23	SMB_DATA[19]	00000000	Data register nineteen
SMB_BASE+24	SMB_DATA[20]	00000000	Data register twenty
SMB_BASE+25	SMB_DATA[21]	00000000	Data register twenty-one
SMB_BASE+26	SMB_DATA[22]	00000000	Data register twenty-two
SMB_BASE+27	SMB_DATA[23]	00000000	Data register twenty-three
SMB_BASE+28	SMB_DATA[24]	00000000	Data register twenty-four
SMB_BASE+29	SMB_DATA[25]	00000000	Data register twenty-five
SMB_BASE+30	SMB_DATA[26]	00000000	Data register twenty-six
SMB_BASE+31	SMB_DATA[27]	00000000	Data register twenty-seven
SMB_BASE+32	SMB_DATA[28]	00000000	Data register twenty-eight
SMB_BASE+33	SMB_DATA[29]	00000000	Data register twenty-nine
SMB_BASE+34	SMB_DATA[30]	00000000	Data register thirty
SMB_BASE+35	SMB_DATA[31]	00000000	Data register thirty-one
SMB_BASE+36	SMB_BCNT	00000000	Block Count Register
SMB_BASE+37	SMB_ALRM_ADDR	00000000	Alarm address
SMB_BASE+38	SMB_ALRM_DATA[0]	00000000	Alarm data register zero
SMB_BASE+39	SMB_ALRM_DATA[1]	00000000	Alarm data register one

The second group of registers contains registers that are USAR-defined SMBus Host registers. These allow the user to access and configure all of the added features available on the UR8HC342, such as SBS device events and interrupt generation.

USAR-defined SMBus host registers

EC offset	Register no.	Register name	Read / Write	Power-up / reset default value
40	R00	EC configuration register 0	R/W	00h
41	R01	EC configuration register 1	R/W	00h
42	R02	EC configuration register 2	R/W	00h
43	R03	EC configuration register 3	R/W	00h
44	R04	EC SWI GPE enable register 0	R/W	00h
45	R05	EC SWI GPE enable register 1	R/W	00h
46	R06	EC SWI GPE status register 0	R/W	00h
47	R07	EC SWI GPE status register 1	R/W	00h
48	R08	EC selector alarm high mask register 0	R/W	00h
49	R09	EC selector alarm high mask register 1	R/W	00h
50	R10	EC selector alarm low mask register 0	R/W	00h
51	R11	EC selector alarm low mask register 1	R/W	00h
52	R12	EC charger alarm high mask register	R/W	00h
53	R13	EC charger alarm low mask register	R/W	00h
54	R14	EC battery alarm mask register	R/W	00h
55	R15	8042 configuration register 0	R	00h
56	R16	8042 configuration register 1	R	00h
57	R17	8042 configuration register 2	R	00h
58	R18	reserved		
59	R19	GIO0 data/direction register	R/W	00h
60	R20-R40	reserved		
81	R41	GIO1 mode register	R/W	00h
82	R42	GIO1 data/direction register	R/W	00h
83	R43	GIO1 PWM1 high byte register	R/W	00h
84	R44	GIO1 PWM1 low byte register	R/W	00h
85	R45	GIO1 PWM0 high byte register	R/W	00h
86	R46	GIO1 PWM0 low byte register	R/W	00h
87	R47	GIO1 DA1 data register	R/W	00h
88	R48	GIO1 DA0 data register	R/W	00h
89	R49	GIO2 data/direction register	R/W	00h
90	R50	GIO2 AD2 data high byte register	R	00h
91	R51	GIO2 AD2 data low byte register	R	00h
92	R52	GIO2 AD1 data high byte register	R	00h
93	R53	GIO2 AD1 data low byte register	R	00h
94	R54	GIO2 AD0 data high byte register	R	00h
95	R55	GIO2 AD0 data low byte register	R	00h
96	R56	GIO3 direction register	R/W	00h
97	R57	GIO3 data register	R/W	00h

R00: EC configuration register 0

This register contains the control bytes that determine the EC and SMBus work modes.

R00: EC configuration register 0 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RES	RES	RES	RES	RES	CHGQREN	SLCTQREN	CHGSLCT

R00: EC configuration register 0 bit descriptions

RES	Reserved
CHGQREN	Charger Alert Query Enable This setting controls whether the SMBus host queries the charger upon receiving an alert. 1– enable 0 – disabled
SLCTQRE	Selector Alert Query Enable This setting controls whether the SMBus host queries the selector upon receiving an alert 1 – enable 0 – disabled
CHGSLCT	Selector and Charger Combined device This selects what configuration is used. In some systems, the charger and the selector are different devices at different addresses, while in some systems, the selector and charger are combined into one device at one address 1 – separate address 0 – combined device at same address

R01: EC configuration register 1 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
KKLEDEN	CLLEDEN	NLLEDEN	SLLEDEN	RES	RES	RES	RES

R01: EC configuration register 1 bit descriptions

RES	Reserved
SLLEDEN	Enable GIO03 as scroll lock LED 1– enable as scroll lock LED 0 – general purpose I/O
NLLEDEN	Enable GIO02 as numeric lock LED 1– enable as numeric lock LED 0 – general purpose I/O
CLLEDEN	Enable GIO01 as caps lock LED 1– enable as caps lock LED 0 – general purpose I/O
KKLEDEN	Enable GIO00 as Katakana LED 1– enable as Katakana LED 0 – general purpose I/O

R02: EC configuration register

This register controls which of the available general purpose event (GPE) pins are selected to function as GPE inputs and which remain general-purpose I/O (GP I/O) pins. Setting a bit in the register to 1 configures the corresponding pin for GPE input, and resetting it to 0 configures it to be used for GP I/O.

Note that a separate register, R04, determines whether a given GPE pin is enabled

R02: EC configuration register 2 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
GPE7AS	GPE6AS	GPE5AS	GPE4AS	GPE3AS	GPE2AS	GPE1AS	GPE0AS

R02: EC configuration register 2 bit descriptions

GPE7AS	Assign pin GPE7 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other
GPE6AS	Assign pin GPE6 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other
GPE5AS	Assign pin GPE5 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other
GPE4AS	Assign pin GPE4 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other
GPE3AS	Assign pin GPE3 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other
GPE2AS	Assign pin GPE2 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other
GPE1AS	Assign pin GPE1 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other
GPE0AS	Assign pin GPE0 to an SWI general purpose event (GPE) 1– assign pin to an SWI general purpose event (GPE) 0 – assign pin to general purpose I/O (GPIO) or other

R03: EC configuration register 3 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
GPE7EDGE	GPE6EDGE	RES	RES	RES	PWBOVER	RES	RES

R03: EC configuration register 3 bit descriptions

RES	RESERVED
GPE7EDGE	Make general purpose event GPE7 rising edge sensitive 1– Make general purpose event GPE7 rising edge sensitive
GPE6EDGE	Make general purpose event GPE6 rising edge sensitive 1– Make general purpose event GPE6 rising edge sensitive
PWBOVER	Enable power button override 1– enable power button override

R04: EC SWI GPE enable register 0 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
GPE7EN	GPE6EN	GPE5EN	GPE4EN	GPE3EN	GPE2EN	GPE1EN	GPE0EN

R04: EC SWI GPE enable register 0 bit descriptions

GPE7EN	Enable SWI general purpose event GPE7 1– Enable SWI general purpose event GPE7
GPE6EN	Enable SWI general purpose event GPE6 1– Enable SWI general purpose event GPE6
GPE5EN	Enable SWI general purpose event GPE5 1– Enable SWI general purpose event GPE5
GPE4EN	Enable SWI general purpose event GPE4 1– Enable SWI general purpose event GPE4
GPE3EN	Enable SWI general purpose event GPE3 1– Enable SWI general purpose event GPE3
GPE2EN	Enable SWI general purpose event GPE2 1– Enable SWI general purpose event GPE2
GPE1EN	Enable SWI general purpose event GPE1 1– Enable SWI general purpose event GPE1
GPE0EN	Enable SWI general purpose event GPE0 1– Enable SWI general purpose event GPE0

R05: EC SWI GPE enable register 1 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RES	MOUSEN	KYBDEN	STUCKEN	ALRTEN	SLCTEN	CHRGEN	BATTEN

R05: EC SWI GPE enable register 1 bit descriptions

RES	RESERVED
MOUSEN	Enable mouse wakeup SWI general purpose event (GPE) 1– Enable mouse wakeup SWI general purpose event (GPE)
KYBDEN	Enable keyboard wakeup SWI general purpose event (GPE) 1– Enable keyboard wakeup SWI general purpose event (GPE)
STUCKEN	Enable SMBus stuck error SWI general purpose event (GPE) 1– Enable SMBus stuck error SWI general purpose event (GPE)
ALRTEN	Enable SMBus alert SWI general purpose event (GPE) 1– Enable SMBus alert SWI general purpose event (GPE)
SLCTEN	Enable selector alarm SWI general purpose event (GPE) 1– Enable selector alarm SWI general purpose event (GPE)
CHRGEN	Enable charger alarm SWI general purpose event (GPE) 1– Enable charger alarm SWI general purpose event (GPE)
BATTEN	Enable battery alarm SWI general purpose event (GPE) 1– Enable battery alarm SWI general purpose event (GPE)

R06 and R07: EC SWI GPE status registers

These registers contain the EC SWI interrupt events status bits, which correspond to the events controlled by the EC SWI GPE enable registers (R04 and R05). For each bit controlling an SWI event in an enable register, the same position bit in the corresponding status register indicates the status of that event. Each status bit is set by the SWI event and can be cleared only by the host writing 1 to its bit position. The status bits are set and reset regardless of whether each event is enabled. Enabling an event controls only whether an interrupt is generated on the event.

R06: EC SWI GPE status register 0 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
GPE7ST	GPE6STN	GPE5ST	GPE4ST	GPE3ST	GPE2ST	GPE1ST	GPE0ST

R06: EC SWI GPE status register 0 bit descriptions

GPE7ST	Flag SWI general purpose event GPE7 1– Flag SWI general purpose event GPE7
GPE6ST	Flag SWI general purpose event GPE6 1– Flag SWI general purpose event GPE6
GPE5ST	Flag SWI general purpose event GPE5 1– Flag SWI general purpose event GPE5
GPE4ST	Flag SWI general purpose event GPE4 1– Flag SWI general purpose event GPE4
GPE3ST	Flag SWI general purpose event GPE3 1– Flag SWI general purpose event GPE3
GPE2ST	Flag SWI general purpose event GPE2 1– Flag SWI general purpose event GPE2
GPE1ST	Flag SWI general purpose event GPE2 1– Flag SWI general purpose event GPE2
GPE0ST	Flag SWI general purpose event GPE1 1– Flag SWI general purpose event GPE1

R07: EC SWI GPE status register 1 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RES	MOUSST	KYBDST	STUCKST	ALRTST	SLCTST	CHRGST	BATTST

R07: EC SWI GPE status register 1 bit descriptions

RES	RESERVED
MOUSST	Flag mouse wakeup SWI general purpose event (GPE) 1– Flag mouse wakeup SWI general purpose event (GPE)
KYBDST	Flag keyboard wakeup SWI general purpose event (GPE) 1– Flag keyboard wakeup SWI general purpose event (GPE)
STUCKST	Flag SMBus stuck error SWI general purpose event (GPE) 1– Flag SMBus stuck error SWI general purpose event (GPE)
ALRTST	Flag SMBus alert SWI general purpose event (GPE) 1– Flag SMBus alert SWI general purpose event (GPE)
SLCTST	Flag selector alarm SWI general purpose event (GPE) 1– Flag selector alarm SWI general purpose event (GPE)
CHRGST	Flag charger alarm SWI general purpose event (GPE) 1– Flag charger alarm SWI general purpose event (GPE)
BATTST	Flag battery alarm SWI general purpose event (GPE) 1– Flag battery alarm SWI general purpose event (GPE)

R08-R11: EC selector alarm mask registers

- R08: EC selector alarm high mask register 0
- R09: EC selector alarm high mask register 1
- R10: EC selector alarm low mask register 0
- R11: EC selector alarm low mask register 1

These registers contain the Smart Selector alert message mask. This controls whether an alert from the Smart Selector generates an SCI interrupt. When the Smart Selector sends an SMBus alert, if the Selector is enabled the SMBus host queries the Selector state. Each bit of the state is masked with the corresponding selector alarm mask flag, and if the result is nonzero, an SCI interrupt is generated. The high mask registers (R08 and R09) control whether a high level for a given bit of the selector state generates an interrupt, while the low mask registers (R10 and R11) control whether a low level for a given bit of the selector state generates an interrupt. By using both of these register pairs, the user can generate an interrupt on either low or high level – or both – for each of the selector state bits.

R08 / R10 EC selector alarm high / low mask register 0 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
BATDCHG	BATCCHG	BATBCHG	BATACHG	BATDPRES	BATCPRES	BATBPRES	BATAPRES

R08 / R10 EC selector alarm high / low mask register 0 bit descriptions

BATDCHG	Mask for Battery D connected to charger state
BATCCHG	Mask for Battery C connected to charger state
BATBCHG	Mask for Battery B connected to charger state
BATACHG	Mask for Battery A connected to charger state
BATDPRES	Mask for Battery D present state
BATCPRES	Mask for Battery C present state
BATBPRES	Mask for Battery B present state
BATAPRES	Mask for Battery A present state

R09/ R11 EC selector alarm high / low mask register 1 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
BATDSMB	BATCSMB	BATBSMB	BATASMB	BATDPWR	BATCPWR	BATBPWR	BATAPWR

R09/ R11 EC selector alarm high / low mask register 1 bit descriptions

BATDSMB	Mask for Battery D connected to SMBus state
BATCSMB	Mask for Battery C connected to SMBus state
BATBSMB	Mask for Battery B connected to SMBus state
BATASMB	Mask for Battery A connected to SMBus state
BATDPWR	Mask for Battery D powering state
BATCPWR	Mask for Battery C powering state
BATBPWR	Mask for Battery B powering state
BATAPWR	Mask for Battery A powering state

R12 and R13: EC charger alarm mask registers

- R12: EC charger alarm high mask register
- R13: EC charger alarm low mask register

These registers contain the Smart Charger alert message masks, which use the same mechanism as the Smart Selector alarm message masks. When the Smart Charger sends an SMBus alert, if the Charger is enabled the SMBus host queries the Charger status. Each bit of its status is masked with the corresponding flag in these registers, and if the result is nonzero, an SCI interrupt is generated. R12 controls whether a high level for a given bit of the charger status generates an interrupt, while R13 controls whether a low level for a given bit of the charger status generates an interrupt. By using both of these register sets, the user can generate an interrupt on either low-or high-level – or both – for each of the charger status bits.

The register contains the following fields.

R12 / R13 EC charger alarm high / low mask register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
AC_PRES	BAT_PRES	PWRFAIL	ALRMINH	THRM_UR	THRM_HOT	THRM_COLD	THRM_OR

R12 / R13 EC charger alarm high / low mask register bit descriptions

AC_PRES	Mask for AC Present bit
BAT_PRES	Mask for Battery Present bit
PWRFAIL	Mask for Power fail bit
ALRMINH	Mask for Alarm inhibited bit
THRM_UR	Mask for Thermistor Hot bit
THRM_HOT	Mask for Battery C powering state
THRM_COLD	Mask for Thermistor Cold bit
THRM_OR	Mask for Thermistor Over-range bit

R14: EC battery alarm mask register

This register contains the Smart Battery alarm message mask. When the SMBus host receives a battery alarm, it ANDs the alarm message with the contents of R14, and if the result is not zero, the EC asserts SCI to the system.

This register contains the following fields.

R14: EC battery alarm mask register bit definitions

bit7	bit 6	bit 5	bit 4
OVERCHARG	TERMCHARG	RES	TEMPCHARG

bit 3	bit 2	bit 1	bit 0
TERMDISCHARG	RES	REMAINCAP	REMAINTIME

R14: EC battery alarm mask register bit descriptions

RES	Reserved
OVERCHARG	Battery over charged alarm mask
TERMCHARG	Battery terminate charge alarm mask
TEMPCHARG	Battery over temperature alarm mask
TERMDISCHARG	Battery terminate discharge alarm mask
REMAINCAP	Battery remaining capacity alarm mask
REMAINTIME	Battery remaining time alarm mask

SMBus Device Access Restrictions

The USAR UR8HC342 ACPITroller™ does not allow access to elements of some specific SMBus devices. These commands should be accomplished among the various SMBus devices and should not be executed through the system. Allowing these commands to be processed through the host could cause damage to some SBS elements and, as such, must be restricted. Often the OS or its drivers must filter these commands. A distinct advantage to the USAR UR8HC342 is its ability to internally shield the SBS devices from these dangerous commands, freeing the OS and drivers from this consideration.

The following commands are inhibited:

- Write charge current command (0x14) to smart battery charger (address 0001001).
- Write charge voltage command (0x15) to smart battery charger (address 0001001).

These commands are disabled because they involve data which can only be accurately calculated and reported only by the smart battery itself.

This page intentionally left blank

USAR ACPITroller™ general input / output options

USAR ACPITroller™ internal virtual SMBus device

A unique feature of the USAR ACPITroller™ is its incorporation of an internal virtual SMBus device. The host can address this device through the ACPI EC interface the same way it would address any external device residing on an SMBus port. Using SMBus commands, the host can read from and write to the register space of the ACPITroller™

The internal virtual SMBus device has the seven-bit binary SMBus address 0100110. It supports the following SMBus protocols:

USAR ACPITroller™ internal virtual SMBus device supported protocols

Protocol	Action
Send byte	Set register page (byte = page number: 0 or 1)
Receive byte	Read register page (byte = page number: 0 or 1)
Write byte	Write to register space (command code = data offset, data = data to write)
Write word	
Write block	
Read byte	Read from register space (command code = data offset, data = data read)
Read word	
Read block	

Page 1 of the ACPITroller™ register space contains the programmable keyboard matrix. Page 0 of the ACPITroller™ register space contains the registers in the following table.

ACPITroller™ registers page 0

8042 offset	Register no.	Register name	Read / Write	Power-up / reset default value
00	R00	EC configuration register 0	R/W	00h
01	R01	EC configuration register 1	R/W	00h
02	R02	EC configuration register 2	R/W	00h
03	R03	EC configuration register 3	R/W	00h
04	R04	EC SWI GPE enable register 0	R/W	00h
05	R05	EC SWI GPE enable register 1	R/W	00h
06	R06	EC SWI GPE status register 0	R/W	00h
07	R07	EC SWI GPE status register 1	R/W	00h
08	R08	EC selector alarm high mask register 0	R/W	00h
09	R09	EC selector alarm high mask register 1	R/W	00h
10	R10	EC selector alarm low mask register 0	R/W	00h
11	R11	EC selector alarm low mask register 1	R/W	00h
12	R12	EC charger alarm high mask register	R/W	00h
13	R13	EC charger alarm low mask register	R/W	00h
14	R14	EC battery alarm mask register	R/W	00h
15	R15	8042 configuration register 0	R	00h
16	R16	8042 configuration register 1	R	00h
17	R17	8042 configuration register 2	R	00h
18	R18	reserved		
19	R19	GIO0 data/direction register	R/W	00h
20	R20-R40	reserved		
41	R41	GIO1 mode register	R/W	00h
42	R42	GIO1 data/direction register	R/W	00h
43	R43	GIO1 PWM1 high byte register	R/W	00h
44	R44	GIO1 PWM1 low byte register	R/W	00h
45	R45	GIO1 PWM0 high byte register	R/W	00h
46	R46	GIO1 PWM0 low byte register	R/W	00h
47	R47	GIO1 DA1 data register	R/W	00h
48	R48	GIO1 DA0 data register	R/W	00h
49	R49	GIO2 data/direction register	R/W	00h
50	R50	GIO2 AD2 data high byte register	R	00h
51	R51	GIO2 AD2 data low byte register	R	00h
52	R52	GIO2 AD1 data high byte register	R	00h
53	R53	GIO2 AD1 data low byte register	R	00h
54	R54	GIO2 AD0 data high byte register	R	00h
55	R55	GIO2 AD0 data low byte register	R	00h
56	R56	GIO3 direction register	R/W	00h
57	R57	GIO3 data register	R/W	00h

GIO0: LED drivers

GIO0 is a 4-bit general-purpose input/output port mapped on the same pins used for the keyboard LEDs. The functions of the pins are determined by register R01, the EC configuration register 1.

R01: EC configuration register 1 bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
KKLEDEN	CLLEDEN	NLLEDEN	SLLEDEN	RES	RES	RES	RES

R01: EC configuration register 1 bit descriptions

RES	Reserved
SLLEDEN	Enable GIO03 as scroll lock LED 1– enable as scroll lock LED 0 – general purpose I/O
NLLEDEN	Enable GIO02 as numeric lock LED 1– enable as numeric lock LED 0 – general purpose I/O
CLLEDEN	Enable GIO01 as caps lock LED 1– enable as caps lock LED 0 – general purpose I/O
KKLEDEN	Enable GIO00 as Katakana LED 1– enable as Katakana LED 0 – general purpose I/O

If a pin is used for general purpose I/O, the direction (input or output) and the data bit of the pin are in register R19, the GIO0 data/direction register. Writing to a bit whose pin is configured as an input has no effect.

R19: GIO0 data/direction register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DIR03	DIR02	DIR01	DIR00	DAT03	DAT02	DAT01	DAT00

R19: GIO0 data/direction register bit descriptions

DIR03	Direction of pin GIO03 1– output 0 – input
DIR02	Direction of pin GIO02 1– output 0 – input
DIR01	Direction of pin GIO01 1– output 0 – input
DIR00	Direction of pin GIO00 1– output 0 – input
DAT03	Data of pin GIO03
DAT02	Data of pin GIO02
DAT01	Data of pin GIO01
DAT00	Data of pin GIO00

GIO1: Analog output for flat panel digital controls

Overview

GIO1 is a two or four channel analog output device. Two pins are configurable as ACPI GPE0 and GPE1 inputs. (This configuration is done in register R02; see Chapter 7.) If they are configured as GPE input, then GIO1 is available only as a two-pin device.

All of the GIO1 pins can operate in a digital input/output mode. If GIO1 is configured as a two-pin device (default mode) its output can function as either a Pulse Width Modulation (PWM) generator or a D/A converter. To select PWM or D/A mode, the OEM must configure the control register defining its operating mode. If the GIO1 is configured as a four-pin device, then two of the pins can be configured as PWM and two function as D/A analog outputs. The following table lists the pin names, power on default assignments as well as the function of each pin in two and four-pin modes.

GIO1 pin usage

Pin name	Default	2-pin analog function	4-pin analog function
GIO13	GIO13	PWM or D/A	D/A
GIO12	GIO12	PWM or D/A	D/A
GPE6/GIO11	GPE6	Not available	PWM
GPE5/GIO10	GPE5	Not available	PWM

Features

- Includes general I/O configuration
- Programmable enhanced general I/O function
- Can be configured as either D/A or PWM
- Two D/A or PWM channels
-

Registers

Register R41 determines the mode of the GIO1x pins. It is not valid to assign the same pin both as D/A and to an enabled PWM channel. If a pin is not assigned as D/A, and it is not assigned to an enabled PWM channel, and it was not configured as a GPE input, then it is used for general I/O (the default).

R41: GIO1 mode register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RES	RES	DA1	DA0	PWM11	PWM1EN	PWM01	PWM0EN

R41: GIO1 mode register bit descriptions

RES	Reserved
DA1	1– Use GIO13 as DA1
DA0	1– Use GIO12 as DA0
PWM11	1 – Use GIO13 as PWM11 0 – Use GIO11 as PWM10
PWM1EN	1 – Enable channel PWM1x
PWM01	1 – Use GIO12 as PWM01 0 – Use GIO10 as PWM00
PWM0EN	1 – Enable channel PWM0x

If a pin is assigned for general I/O, then the appropriate bits in R42 contain the direction and data of the I/O. Writing to a bit whose pin is configured as an input has no effect.

R42: GIO1 data/direction register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DIR13	DIR12	DIR11	DIR10	DAT13	DAT12	DAT11	DAT10

R42: GIO1 data/direction register bit descriptions

DIR13	Direction of pin GIO13 1– output 0 – input
DIR12	Direction of pin GIO12 1– output 0 – input
DIR11	Direction of pin GIO11 1– output 0 – input
DIR10	Direction of pin GIO10 1– output 0 – input
DAT13	Data of pin GIO13
DAT12	Data of pin GIO12
DAT11	Data of pin GIO11
DAT10	Data of pin GIO10

Note: Only two simultaneous PWM outputs are available. PWM Channel 0 can be selected to GIO12 or GIO10, where it is referred to as PWM01 or PWM00 respectively, and PWM Channel 1 can be selected to GIO13 or GIO11, where it is referred to as PWM11 or PWM10 respectively.

PWM operation

For each PWM channel, the PWM generated is controlled by 14 bits, which are contained in registers R43-R46. These 14 bits are further broken down into a group of upper eight bits and a group of lower six bits. For PWM1, the upper eight bits are located in R43, and the lower six bits are located in the lower six bits of R44. . For PWM0, the upper eight bits are located in R45, and the lower six bits are located in the lower six bits of R46.

The manner in which the PWM works is based on a cycle and a sub-cycle. For an oscillation frequency X_{IN} of 8MHz, the cycle period is 4096µs. Each cycle is broken down into sub-cycles of 64 µs. The time resolution available is 250 ns.

The upper eight bits of data determine how long an "H"-level signal is output during each sub-cycle. If the upper eight bits contain the value N, then in each sub-cycle the output signal is H for a time of N*t, where t = 250 ns, the minimum time resolution.

The lower six bits allow the user to lengthen the high, for some sub-cycles, by a duration of t = 250 ns. As indicated in the table below, these bits determine which sub-cycles are lengthened. For each pulse lengthened, the leading edge of the pulse is lengthened. As a result, an accurate wave form can be duplicated without the use of complex external filters – by changing the length of specific sub-periods instead of simply changing the global "H" duration.

For example, if the upper eight bits of the 14-bit data are 03_{16} and the lower six bits are 05_{16}, the length of the "H"- level output in sub-periods t_8, t_{24}, t_{32}, t_{40} and t_{56} is 4*t, and its length 3*t in all other sub-periods.

Relationship between the lower 6 bits of data and the period set by the ADD bit

Lower 6 bits of data	Sub-periods tm length end (m = 0 to 63)
0 0 0 0 0 0 LSB	None
0 0 0 0 0 1	m = 32
0 0 0 0 1 0	m = 16, 48
0 0 0 1 0 0	m = 8, 24, 40, 56
0 0 1 0 0 0	m = 4, 12, 20, 28, 36, 44, 52, 60
0 1 0 0 0 0	m = 2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
1 0 0 0 0 0	m = 1, 3, 5, 7,..., 57, 59, 61, 63

Data written to the lower register is transferred to the PWM latch once during each PWM period (every 4096µs), and data written to the higher register is transferred to the PWM latch once during each sub-period (every 64µs).

The signal output to the PWM output pin corresponds to the contents of the latch. When the lower register is read, the contents of the latch are read. Bit 7 of the lower register indicates whether the transfer to the PWM latch is completed; the transfer is considered complete when bit 7 is zero.

D/A operations

The D/A converter provides 8-bit resolution and can output to two channels.

D/A conversion is initiated by writing a non-zero value into the D/A data register for the D/A channel (R47 for channel 1, R48 for channel 0). The D/A conversion results are output on the corresponding output channel, if the D/A enable flag for the channel is set to 1.

The output analog voltage ("V") is determined by value "n" (n = decimal number) in the D/A conversion resister, as follows:

$V=(V_{REF})(n/256)$ (n=0255), where V_{REF} indicates a reference voltage.

Note: When using a D/A converter, set Vcc to 4.0V or more.

Reset Considerations

At reset, all control registers are cleared to 0 and all data registers are also cleared to 0. Therefore, all pins are placed in a high impedance state. Since the D/A output does not have a buffer, an external buffer should be used when connecting it to a low-impedance load.

GIO2: 3 channel 10-bit analog to digital converter

Overview

GIO2 can function as a 10-bit A/D converter or as a general purpose I/O device. To use each of the A/D channels, the OEM initializes the corresponding channels as described in this section.

Features

- Can be configured as either general I/O or A/D
- 10-bit A/D
- Up to three A/D channels
- Programmable enhanced general I/O function

IVS2 pin usage

Pin name	Power on default function
GIO2:0	Input
GIO2:1	Input
GIO2:2/SS_SDA1	Input. If the internal Smart Selector is enabled this pin acts as the second SMBus data line

Registers

If a pin is used for general purpose I/O, the direction (input or output) and the data bit of the pin are in register R49, the GIO2 data/direction register. Writing to a bit whose pin is configured as an input has no effect.

R49: GIO2 data/direction register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RES	DIR22	DIR21	DIR20	RES	DAT22	DAT21	DAT20

R49: GIO2 data/direction register bit descriptions

RES	reserved
DIR22	Direction of pin GIO22 1– output 0 – input
DIR21	Direction of pin GIO21 1– output 0 – input
DIR20	Direction of pin GIO20 1– output 0 – input
DAT22	Data of pin GIO22
DAT21	Data of pin GIO21
DAT20	Data of pin GIO20

If the pins are configured for A/D conversion, the digital data is returned in the read-only registers R50-R55 (2 bytes of data for each A/D channel).

A/D Comparison Voltage Generator

In 10-bit mode, the A/D function divides the voltage between AVss and Vref by 1024. The result returned signifies the number of these divisions to which the input analog voltage corresponds.

Thus, in 10 bit A/ D mode, with 10 bit read: Vref = Vref /1024 n (n=0 to 1023).

In 10 bit A/D mode with 8 bit read: Vref = Vref / 256 n (n=0 to 255).

GIO3: general purpose I/O

GIO3 is a 6-pin port. The pins can be configured as ACPI general purpose event inputs (GPE) (see Chapter 7), or as general purpose I/O pins (the default).

If a pin is used for general purpose I/O, the direction (input or output) of the pin are specified in register R56, the GIO3 direction register, and the data bit of the pin is in register R57, the GIO3 data register. Writing to a bit whose pin is configured as an input has no effect.

R56: GIO3 direction register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DIR37	DIR36	DIR35	DIR34	DIR33	DIR32	RES	RES

R56: GIO3 direction register bit descriptions

RES	reserved
DIR37	Direction of pin GIO37 1– output 0 – input
DIR36	Direction of pin GIO36 1– output 0 – input
DIR35	Direction of pin GIO35 1– output 0 – input
DIR34	Direction of pin GIO34 1– output 0 – input
DIR33	Direction of pin GIO33 1– output 0 – input
DIR32	Direction of pin GIO32 1– output 0 – input

R57: GIO3 data register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DAT07	DAT06	DAT05	DAT04	DAT03	DAT02	RES	RES

R57: GIO3 data register bit descriptions

RES	reserved
DAT37	Data of pin GIO37
DAT36	Data of pin GIO36
DAT35	Data of pin GIO35
DAT34	Data of pin GIO34
DAT33	Data of pin GIO33
DAT32	Data of pin GIO32

This page intentionally left blank

Electrical characteristics

Absolute maximum ratings

(VSS = 0V, Ambient Temperature T_A is in the range T_{LOW} to T_{HIGH})

Absolute maximum ratings

Parameter	Symbol	Value	Unit
Supply Voltage	V_{DD}	-0.3 to +7.0	V
Input voltage			
All pins except 2-9	V_{IN}	-0.3 to V_{DD}+0.3	V
Pins 2-9			
	V_{IN}	-0.3 to +5.8	V
Output current			
Total peak for all pins	$\sum I_{OH\ (PEAK)}$	-80	
	$\sum I_{OL\ (PEAK)}$	80	mA
Total average for all pins	$\sum I_{OH\ (AVG)}$	-40	
	$\sum I_{OL\ (AVG)}$	40	mA
All pins except 31-34			
Peak for each pin	$I_{OH\ (PEAK)}$	-10	
	$I_{OL\ (PEAK)}$	10	mA
Average for each pin	$I_{OH\ (AVG)}$	-5	
	$I_{OL\ (AVG)}$	5	mA
Pins 31-34			
Peak for each pin	$I_{OH\ (PEAK)}$	-10	
	$I_{OL\ (PEAK)}$	20	mA
Average for each pin	$I_{OH\ (AVG)}$	-5	
	$I_{OL\ (AVG)}$	15	mA
Temperature range			
Operating Temperature	T_{LOW} to T_{HIGH}	-20 to 85	ºC
Storage Temperature	T_{STG}	-40 to 125	ºC

Recommended operating conditions / electrical characteristics

Digital section

(VSS = 0V, Ambient Temperature T_A is in the range T_{LOW} to T_{HIGH})

Recommended operating conditions/electrical characteristics, digital Section

Parameter	Symbol	Min	Typ	Max	Unit
Supply voltage	V_{DD}	2.7	3.0	5.5	V
Input logic high voltage					
All pins except 2-9	V_{IH}	$0.8V_{DD}$		V_{DD}	V
Pins 2-9	V_{IH}	$0.8V_{DD}$		5.5	V
Input logic low voltage					
All pins except 28	V_{IL}	0		$0.2V_{DD}$	V
Pin 28	V_{IL}	0		$0.16V_{DD}$	V
Input current $V_I = V_{SS}, V_{DD}$)	I_{IL} / I_{IH}	-5.0	0	5.0	µA
Input Pull-up Current (pins 56-58 / IP6-IP8, $V_I = V_{SS}$)	I_{PUP}	-120		-10	µA
Output voltage					
I_{OH} = -1.0 mA	V_{OH}	V_{DD}–1.0			V
I_{OL} = 1.6 mA	V_{OL}			0.4	V
Current Consumption	*NOTE1*				
Full Speed Mode (F_{OSC}=4MHz)	I_{DD}		3.5	7.0	mA
Reduced Power Mode (F_{OSC}=4MHz)	I_{DD}		750		µA
Stop Mode (Interrupts active, F_{OSC}=0)	I_{DD}		.1	1.0 (T_A = 25ºC) 10 (T_A = 85ºC)	µA

NOTE1 *Current consumption values do not include any loading on the output pins or Analog Reference Current for the built-in A/D or D/A modules.*

Recommended operating conditions / electrical characteristics

Analog section

(VSS = 0V, Ambient Temperature T_A is in the range T_{LOW} to T_{HIGH})

Recommended operating conditions/electrical characteristics, analog section

Parameter	Symbol	Min	Typ	Max	Unit
Analog Signal Ground	AV_{SS}		0		V
Analog Reference Voltage	AV_{REF}	2.7	V_{DD}	V_{DD}	V
A/D Resolution	-			10	Bits
A/D Absolute Accuracy	-			± 4	LSb
A/D Analog Input Voltage Range	V_{IA}	AV_{SS}		AV_{REF}	V
A/D Analog Input Current	I_{IA}			5.0	µA
Analog Reference Current NOTE[2] (A/D is active)	I_{AVREF}			200	µA
D/A Resolution	-			8	Bits
D/A Absolute Accuracy	-			2.5	%
D/A Output Impedance	R_O	1	2.5	4.0	KOhms
Analog Reference Current NOTE[3] (D/A is active, Output = Full Scale)	I_{AVREF}			3.2	mA

NOTE[2] *Since built-in A/D module consumes current only during short periods of time when A/D conversion is actually requested, the Analog Reference Current for the built-in A/D module is* not *a significant contributor to the overall power consumption.*

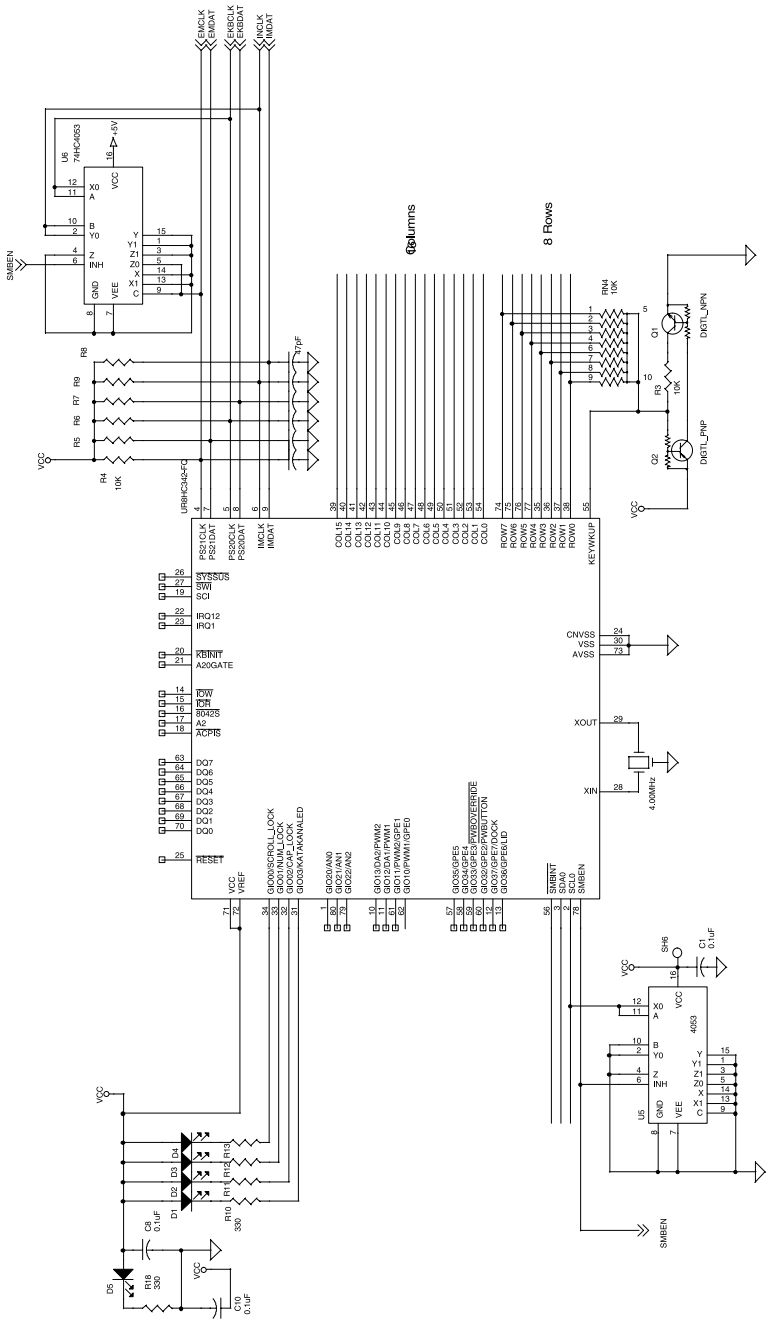
NOTE[3] *The Analog Reference Current for the built-in D/A module correlates linearly to the Output Voltage. For D/A output of 0V, the Analog Reference Current is null. For D/A outputs approaching Full Scale (AVREF), the maximum Analog Reference Current is indicated in this Table. This current is a significant contributor to the overall power consumption.*

Power consumption while operating the PWM channels

Users should consider the built-in PWM channels for generating slowly changing DC control voltages. Since continuous clocking is necessary for the PWM operations, the only penalty for using the built-in PWM channels is the requirement for the chip to operate at least in the Reduced Power Mode, with typical Current Consumption of 750 µA.

This page intentionally left blank

Sample schematic



This page intentionally left blank

Appendix A

USAR AlphaKey™ standard PS/2 key number definitions

The following table lists Standard PS/2 key numbers used by the USAR AlphaKey™ keyboard manager.

Standard PS/2 Key Number Definitions

USAR Key # Dec	Hex	PS/2 Key # Dec	Scan Codes Make/Break SCS1 Hex	Key Label
0	00	1	No code	Null key
1	01	1	29/A9	` / ~
2	02	2	02/82	1 / !
3	03	3	03/83	2 /@
4	04	4	04/84	3 / #
5	05	5	05/85	4 / $
6	06	6	06/86	5 / %
7	07	7	07/87	6 / ^
8	08	8	08/88	7 / &
9	09	9	09/89	8 / *
10	0A	10	0A/8A	9 / (
11	0B	11	0B/8B	0 /)
12	0C	12	0C/8C	- / _
13	0D	13	0D/8D	= / +
14	0E	15	0E/8E	Backspace
15	0F	16	0F/8F	Tab
16	10	17	10/90	Q
17	11	18	11/91	W
18	12	19	12/92	E
19	13	20	13/93	R
20	14	21	14/94	T
21	15	22	15/95	Y
22	16	23	16/96	U
23	17	24	17/97	I
24	18	25	18/98	O
25	19	26	19/99	P
26	1A	27	1A/9A	[/ {
27	1B	28	1B/9B	] / }
28	1C	29	2B/AB	\ / |
29	1D	31	1E/9E	A
30	1E	32	1F/9F	S
31	1F	33	20/A0	D
32	20	34	21/A1	F
33	21	35	22/A2	G
34	22	36	23/A3	H
35	23	37	24/A4	J

Standard PS/2 Key Number Definitions

USAR Key # Dec	Hex	PS/2 Key # Dec	Scan Codes Make/Break SCS1 Hex	Key Label
36	24	38	25/A5	K
37	25	39	26/A6	L
38	26	40	27/A7	; / :
39	27	41	28/A8	‘ / “
40	28	43	1C/9C	Enter
41	29	46	2C/AC	Z
42	2A	47	2D/AD	X
43	2B	48	2E/AE	C
44	2C	49	2F/AF	V
45	2D	50	30/B0	B
46	2E	51	31/B1	N
47	2F	52	32/B2	M
48	30	53	33/B3	, / <
49	31	54	34/B4	. / >
50	32	55	35/B5	/ /?
51	33	61	39/B9	Space
52	34	110	01/81	Esc
53	35	75	E0 52/E0 D2	Insert
54	36	76	E0 53/E0 D3	Delete
55	37	79	E0 4B/E0 CB	Arrow Left
56	38	80	E0 47/E0 C7	Home
57	39	81	E0 4F/E0 CF	End
58	3A	83	E0 48/E0 C8	Arrow Up
59	3B	84	E0 50/E0 D0	Arrow Down
60	3C	85	E0 49/E0 C9	Page Up
61	3D	86	E0 51/E0 D1	Page Down
62	3E	89	E0 4D/E0 CD	Arrow Right
63	3F	100	37/B7	N. Star
64	40	106	4E/CE	N. +
65	41	93	4F/CF	N1 / K.P.End
66	42	98	50/D0	N2 / K.P.Arrow Down
67	43	103	51/D1	N3 / K.P.PgDn
68	44	92	4B/CB	N4 / K.P.Arrow Left
69	45	97	4C/CC	N5
70	46	102	4D/CD	N6 / K.P.Arrow Right
71	47	91	47/C7	N7 / K.P.Home
72	48	96	48/C8	N8 / K.P.Arrow Up
73	49	101	49/C9	N9 / K.P.PgUp
74	4A	99	52/D2	N0 / K.P.Ins
75	4B	104	53/D3	N. Period / K.P.Del
76	4C	105	4A/CA	N. -
77	4D	108	E0 1C/E0 9C	N. Enter
78	4E	95	E0 35/E0 B5	N. /
79	4F	112	3B/BB	F1
80	50	113	3C/BC	F2
81	51	114	3D/BD	F3
82	52	115	3E/BE	F4
83	53	116	3F/BF	F5

Standard PS/2 Key Number Definitions

USAR Key # Dec	Hex	PS/2 Key # Dec	Scan Codes Make/Break SCS1 Hex	Key Label
84	54	117	40/C0	F6
85	55	118	41/C1	F7
86	56	119	42/C2	F8
87	57	120	43/C3	F9
88	58	121	44/C4	F10
89	59	122	57/D7	F11
90	5A	123	58/D8	F12
91	5B	44	2A/AA	L. Shift
92	5C	57	36/B6	R. Shift
93	5D	60	38/B8	L. Alt
94	5E	62	E0 38/E0 B8	R. Alt
95	5F	58	1D/9D	L. Ctrl
96	60	64	E0 1D/E0 9D	R. Ctrl
97	61	30	3A/BA	Caps Lock
98	62	90	45/C5	Num Lock
99	63	125	46/C6	Scroll Lock
100	64	124	E0 2A E0 37 / E0 B7 E0 AA	Print Scr / SysReq
101	65	126	E1 1D 45 E1 9D C5	Pause / Break
102	66		E0 5B/E0 DB	Left Win
103	67		E0 5C/E0 DC	Right Win
104	68		E0 5D/E0 DD	Win Application
105	69		FF	Overrun Error
106	6A		No code	Function Key
107	6B		No code	Sticky Key
108	6C		E0 5E/E0 DE	Power event
109	6D		E0 5F/E0 DF	Sleep event
110	6E		E0 63/E0 E3	Wake event
111	6F		70/F0	Katakana
112	70		7B/FB	NFER, F20
113	71		79/F9	XFER, F17
114	72		7D/FD	Yen, F23
115	73		73/F3	/ / _
116	74		5B/DB	F13
117	75		5C/DC	F14
118	76		5D/DD	F15
119	77		63/E3	F16
120	78		65/E5	F18
121	79		66/E6	F19
122	7A		68/E8	F21
123	7B		69/E9	F22
124	7C		6B/EB	F24
125–127	7D – 7F			Reserved

This page intentionally left blank

Appendix B

USAR AlphaKey™ default matrix & layout

USAR AlphaKey™ default scan matrix

Columns	Rows	Matrix RAM offset	USAR key number	QWERTY layout key label	NumPad layout key label	Fn layout key label
COL 0	ROW 0	00	0			
	ROW 1	01	0			
	ROW 2	02	0			
	ROW 3	03	0			
	ROW 4	04	0			
	ROW 5	05	0			
	ROW 6	06	0			
	ROW 7	07	106	Function	Function	Function
COL 1	ROW 0	08	0			
	ROW 1	09	0			
	ROW 2	0A	0			
	ROW 3	0B	0			
	ROW 4	0C	0			
	ROW 5	0D	102	Left Win	Left Win	Left Win
	ROW 6	0E	0			
	ROW 7	0F	0			
COL 2	ROW 0	10	15	Tab	Tab	Tab
	ROW 1	11	97	Caps Lock	Caps Lock	Caps Lock
	ROW 2	12	2	1 / !	1 / !	1 / !
	ROW 3	13	30	S	S	S
	ROW 4	14	41	Z	Z	Z
	ROW 5	15	29	A	A	A
	ROW 6	16	16	Q	Q	Q
	ROW 7	17	52	escape	escape	escape
COL 3	ROW 0	18	0			
	ROW 1	19	0			
	ROW 2	1A	0			
	ROW 3	1B	0			
	ROW 4	1C	94	Right Alt	Right Alt	Right Alt
	ROW 5	1D	0			
	ROW 6	1E	0			
	ROW 7	1F	93	Left Alt	Left Alt	Left Alt

USAR AlphaKey™ default scan matrix

Columns	Rows	Matrix RAM offset	USAR key number	QWERTY layout key label	NumPad layout key label	Fn layout key label
COL 4	ROW 0	20	79	F1	F1	F1
	ROW 1	21	80	F2	F2	F2
	ROW 2	22	81	F3	F3	F3
	ROW 3	23	18	E	E	E
	ROW 4	24	31	D	D	D
	ROW 5	25	17	W	W	W
	ROW 6	26	3	2 / @	2 / @	2 / @
	ROW 7	27	42	X	X	X
COL 5	ROW 0	28	141	8 / *	8 / up arrow	8 / *
	ROW 1	29	140	9 / (	9 / PGUP	9 / (
	ROW 2	2A	136	I	5	I
	ROW 3	2B	48	, / <	, / <	, / <
	ROW 4	2C	51	space	space	space
	ROW 5	2D	134	K	2 / dn arrow	K
	ROW 6	2E	137	U	4 / left arrow	U
	ROW 7	2F	135	M	0 / INS	M
COL 6	ROW 0	30	84	F6	F6	F6
	ROW 1	31	4	3 / #	3 / #	3 / #
	ROW 2	32	5	4 / $	4 / $	4 / $
	ROW 3	33	32	F	F	F
	ROW 4	34	43	C	C	C
	ROW 5	35	19	R	R	R
	ROW 6	36	6	5 / %	5 / %	5 / %
	ROW 7	37	82	F4	F4	F4
COL 7	ROW 0	38	87	F9	F9	F9
	ROW 1	39	83	F5	F5	F5
	ROW 2	3A	7	6 / ^	6 / ^	6 / ^
	ROW 3	3B	44	V	V	V
	ROW 4	3C	45	B	B	B
	ROW 5	3D	33	G	G	G
	ROW 6	3E	20	T	T	T
	ROW 7	3F	85	F7	F7	F7
COL 8	ROW 0	40	88	F10	F10	F10
	ROW 1	41	89	F11	F11	F11
	ROW 2	42	86	F8	F8	F8
	ROW 3	43	46	N	N	N
	ROW 4	44	34	H	H	H
	ROW 5	45	21	Y	Y	Y
	ROW 6	46	142	7 / &	7 / HOME	7 / &
	ROW 7	47	138	J	1 / END	J

USAR AlphaKey™ default scan matrix

Columns	Rows	Matrix RAM offset	USAR key number	QWERTY layout key label	NumPad layout key label	Fn layout key label
COL 9	ROW 0	48	90	F12	F12	F12
	ROW 1	49	143	0 /)	*	0 /)
	ROW 2	4A	133	O	6 / rt arrow	O
	ROW 3	4B	145	. / >	. / DEL	. / >
	ROW 4	4C	0			
	ROW 5	4D	0			
	ROW 6	4E	129	L	3 / PGDN	NFER [1], F20
	ROW 7	4F	139	num lock	num lock	scroll lock
COL 10	ROW 0	50	101	Pause / Break	Pause / Break	Pause / Break
	ROW 1	51	13	= / +	= / +	= / +
	ROW 2	52	27	] / }	] / }	] / }
	ROW 3	53	28	\ / |	\ / |	\ / |
	ROW 4	54	0			
	ROW 5	55	0			
	ROW 6	56	26	[/ {	[/ {	[/ {
	ROW 7	57	1	` / ~	` / ~	` / ~
COL 11	ROW 0	58	14	Back Space	Back Space	Back Space
	ROW 1	59	146	down arrow	down arrow	PGDN
	ROW 2	5A	147	up arrow	up arrow	PGUP
	ROW 3	5B	39	' / "	' / "	' / "
	ROW 4	5C	104	Win appl.	Win appl.	Win appl.
	ROW 5	5D	40	ENTER	ENTER	ENTER
	ROW 6	5E	0			
	ROW 7	5F	13	insert	insert	insert
COL 12	ROW 0	60	0			
	ROW 1	61	0			
	ROW 2	62	0			
	ROW 3	63	103	RWIN	RWIN	RWIN
	ROW 4	64	0			
	ROW 5	65	0			
	ROW 6	66	0			
	ROW 7	67	0			
COL 13	ROW 0	68	148	right arrow	Num Lock	end
	ROW 1	69	149	left arrow	left arrow	home
	ROW 2	6A	150	delete	delete	SysRq
	ROW 3	6B	144	/ / ?	/	/ / ?
	ROW 4	6C	0			
	ROW 5	6D	130	; / :	+	; / :
	ROW 6	6E	151	P	-	P
	ROW 7	6F	12	- / _	- / _	- / _

USAR AlphaKey™ default scan matrix

Columns	Rows	Matrix RAM offset	USAR key number	QWERTY layout key label	NumPad layout key label	Fn layout key label
COL 14	ROW 0	70	0			
	ROW 1	71	95	left ctrl	left ctrl	left ctrl
	ROW 2	72	0			
	ROW 3	73	0			
	ROW 4	74	0			
	ROW 5	75	0			
	ROW 6	76	96	right ctrl	right ctrl	right ctrl
	ROW 7	77	0			
COL 15	ROW 0	78	91	left shift	Page Up	Page Up
	ROW 1	79	0			
	ROW 2	7A	92	right shift	] / }	] / }
	ROW 3	7B	0			
	ROW 4	7C	0			
	ROW 5	7D	0			
	ROW 6	7E	0			
	ROW 7	7F	0			

Appendix C

Standard SMBus registers

Protocol register, SMB_PRTCL

This register determines the type of SMBus transaction generated on the SMBus. In addition to indicating the protocol type to the SMBus host controller, a write to this register initiates the transaction on the SMBus.

Bit descriptions for protocol register SMB_PRTCL

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
PROTOCOL							

The values of the PROTOCOL are as follows:

PROTOCOL definitions for protocol register SMB_PRTCL

Value (hex)	Meaning
00	Controller Not In Use
01	Reserved
02	Write Quick Command
03	Read Quick Command
04	Send Byte
05	Receive Byte
06	Write Byte
07	Read Byte
08	Write Word
09	Read Word
0A	Block Write
0B	Block Read
0C	Process Call

When the OS initiates a new command such as write to the SMB_PRTCL register, the SMBus Controller first updates the SMB_STS register and then clears the SMB_PRTCL register. After the SMB_PRTCL register is cleared, the host controller query value is raised.

Status register, SMB_STS

This register indicates general status on the SMBus. This includes SMBus host controller command completion status, alarm received status, and error detection status (the error codes are defined later in this section). Whenever a new command is issued using a write to the protocol register (SMB_PRTCL), bits 7 and 2 of this register are reset to 0. This register is always written with the error code before clearing the protocol register. The SMBus host controller query event (that is, an SMBus host controller interrupt) is raised after the clearing of the protocol register.

NOTE: The OS driver must ensure the ALRM bit is cleared after it has been serviced by writing '00' to the SMB_STS register.

Status register SMB_STS bit descriptions

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
DONE	ALRM	RES	RES	RES	STATUS	RES	RES

Status register SMB_STS bit descriptions

RES	Reserved.
DONE	When set to 1, indicates the last command has completed and no error.
ALRM	When set to 1, indicates an SMBus alarm message has been received.
STATUS	When set to 1, indicates SMBus communication status for one of the reasons listed in the following table.

The following table shows SMBus Status Codes.

SMBus Status Codes

Status Code	Name	Description
00h	SMBus OK	Indicates the transaction has been successfully completed.
07h	SMBus Unknown Failure	Indicates failure because of an unknown SMBus error.
10h	SMBus Device Address Not Acknowledged	Indicates the transaction failed because the slave device address was not acknowledged.
11h	SMBus Device Error Detected	Indicates the transaction failed because the slave device signaled an error condition.
12h	SMBus Device Command Access Denied	Indicates the transaction failed because the SMBus host does not allow the specific command for the device being addressed. For example, the SMBus host might not allow a caller to adjust the Smart Battery Charger's output.
13h	SMBus Unknown Error	Indicates the transaction failed because the SMBus host encountered an unknown error.

SMBus Status Codes

Status Code	Name	Description
17h	SMBus Device Access Denied	Indicates the transaction failed because the SMBus host does not allow access to the device addressed. For example, the SMBus host might not allow a caller to directly communicate with an SMBus device that controls the system's power planes.
18h	SMBus Timeout	Indicates the transaction failed because the SMBus host detected a timeout on the bus.
19h	SMBus Host Unsupported Protocol	Indicates the transaction failed because the SMBus host does not support the requested protocol.
1Ah	SMBus Busy	Indicates that the transaction failed because the SMBus host reports that the SMBus is presently busy with some other transaction. For example, the Smart Battery might be sending charging information to the Smart Battery Charger.

All other status codes are reserved.

Address register, SMB_ADDR

This register contains the 7-bit address to be generated on the SMBus. This is the first byte to be sent on the SMBus for all of the different protocols.

SMB_ADDR address register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
ADDRESS (A6:A0)							RES

RES *Reserved*
ADDRESS *7-bit SMBus address*

Command register, SMB_CMD

This register contains the command byte to be sent to the target device on the SMBus and is used for all of the protocols except for Receive Byte, Read Quick Command, and Write Quick Command. For those protocols, the value in SMB_CMD has no effect.

SMB_CMD command register bit definitions

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
COMMAND							

Data register array, SMB_DATA[i], i=0-31

This bank of registers contains the remaining bytes to be sent or received in any of the different protocols that can be run on the SMBus. The SMB_DATA registers are defined on a per-protocol basis and, as such, provide efficient use of register space.

SMB_DATA[I] data register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DATA							

Block count register, SMB_BCNT

This register contains the block count, used in the Block Read and Block Write protocols.

SMB_BCNT block count register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RES			BCNT				

RES *Reserved*
BCNT *Block Count*

Alarm address register, SMB_ALRM_ADDR

This register contains the source address of an alarm message received by the host controller from the SMBus master that initiated the alarm. The address indicates the slave address of the device on the SMBus that initiated the alarm message. The status of the alarm message is contained in the SMB_ALRM_DATAx registers. Once an alarm message has been received, the SMBus host controller must clear the ALRM status bits to receive further alarm messages.

The OS driver does not read the alarm address and alarm data registers until the alarm status bit is set. The OS driver then reads the three bytes, and clears the alarm status bit to indicate that the alarm registers are now available for the next event.

SMB_BCNT block count register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
ADDRESS (A6:A0)							RES

RES *Reserved*
ADDRESS *7-bit slave address (A6:A0) of the SMBus device that initiated the alarm message*

Alarm data registers, SMB_ALRM_DATA[0], SMB_ALRM_DATA[1]

These registers contain the two data bytes of an alarm message received by the host controller, from the SMBus master that initiated the alarm. These data bytes indicate the specific reason for the alarm message, to allow the OS to take corrective action. Once an alarm message has been received, the SMBus host controller must clear the ALRM status bits to receive further alarm messages.

SMB_ALRM_DATA alarm data register bit definitions

bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DATA (D7:0)							

Standard SMBus protocol

Protocol description

All registers should be written with the appropriate values before writing the protocol value that starts the SMBus transaction. All transactions can be completed in one pass.

Write Quick

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_PRTCL: Write 0x02 to initiate quick write protocol.

Data Returned:

- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Read Quick

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_PRTCL: Write 0x03 to initiate quick read protocol.

Data Returned:

- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Send Byte

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_PRTCL: Write 0x04 to initiate send byte protocol.

Data Returned:

- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Receive Byte

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_PRTCL: Write 0x05 to initiate receive byte protocol.

Data Returned:

- SMB_DATA[0]: Data byte received.
- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Write Byte

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_DATA[0]: Data byte to be sent.
- SMB_PRTCL: Write 0x06 to initiate write byte protocol.

Data Returned:

- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Read Byte

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_PRTCL: Write 0x07 to initiate read byte protocol.

Data Returned:

- SMB_DATA[0]: Data byte received.
- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Write Word

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_DATA[0]: Low data byte to be sent.
- SMB_DATA[1]: High data byte to be sent.
- SMB_PRTCL: Write 0x08 to initiate write word protocol.

Data Returned:

- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Read Word

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_PRTCL: Write 0x09 to initiate read word protocol.

Data Returned:

- SMB_DATA[0]: Low data byte received.
- SMB_DATA[1]: High data byte received.
- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Write Block

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_DATA[0- Data bytes to write (1-32).
- SMB_BCNT: Number of data bytes (1-32) to be sent.
- SMB_PRTCL: Write 0x0A to initiate write block protocol.

Data Returned:

- SMB_PRTCL: 0x00 to indicate command completion.
- SMB_STS: Status code for transaction.

Read Block

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_PRTCL: Write 0x0B to initiate read block protocol.

Data Returned:

- SMB_BCNT: Number of data bytes (1-32) received.
- SMB_DATA[0:31 Data bytes received (1-32).
- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

Process Call

Data Sent:

- SMB_ADDR: Address of SMBus device.
- SMB_CMD: Command byte to be sent.
- SMB_DATA[0]: Low data byte to be sent.
- SMB_DATA[1]: High data byte to be sent.
- SMB_PRTCL: Write 0x0C to initiate process call protocol.

Data Returned:

- SMB_DATA[0]: Low data byte received.
- SMB_DATA[1]: High data byte received.
- SMB_STS: Status code for transaction.
- SMB_PRTCL: 0x00 to indicate command completion.

This page intentionally left blank

A Semtech Company

For sales information and product literature, contact:

USAR – A Semtech Company
568 Broadway
New York, NY 10012
info@usar.com
http://www.usar.com
212 226 2042 Telephone
212 226 3215 Telefax

In Japan:
Semtech Japan
Tel: +81-45-948-5925
Fax: +81-45-948-5930

In Taiwan:
Semtech Asia/Pacific Sales
Tel: +886-2-2748-3380
Fax: +886-2-2748-3390

Koryo Electronics Co., Ltd.
Tel: +886-2-2698-1143
E-mail: alex.chen@koryo.com.tw

In Korea:
Semtech Korea
Tel: +82-2-527-4377
Fax: +82-2-527-4377

In Europe:
Semtech Limited
+44-1592-630350
+44-1592-774781