PC87591x Host Interrupt Routing via the CR16B Core

National Semiconductor
Application Note 1283
February 2003
Revision 1.1

1.0 Scope

This application note provides an overview of the generation schemes for host interrupts that are routed through the CR16B Core ("core"). This alternate routing results in either an interrupt generated from the core or a core wake-up event. The RTC interrupt is used as an example.

Under normal conditions, a PC87591x interrupt to the South Bridge (SB) is routed in the PC87591x host domain. The interrupt is sent via SuperI/O configuration registers to the IRQ serializer, which outputs it to the host using an SERIRQ signal.

Some SB devices are unable to receive certain IRQ inputs at all; other SB devices are unable to receive IRQ inputs while the system is in a specific power state. In both cases, the interrupt can be alternately routed to the SB via the core which then generates either a parallel interrupt or a wake-up event to the SB. The "module IRQ routing to wake-up event" mechanism is used to send the interrupt via this alternate route to the core. In the core, an interrupt handler can transmit a parallel interrupt to the SB via a GPIO.

Using the RTC interrupt as an example, this application note describes:

- The host interrupt path in System Active mode (Section 2.1)
- The RTC alarm path in System Deep Sleep mode (Section 2.2)
- The alternative path of a host interrupt in System Active mode when the SB does not recognize the IRQ8 interrupt (Section 3.0); sample code for this alternate path is shown in Section 4.0.

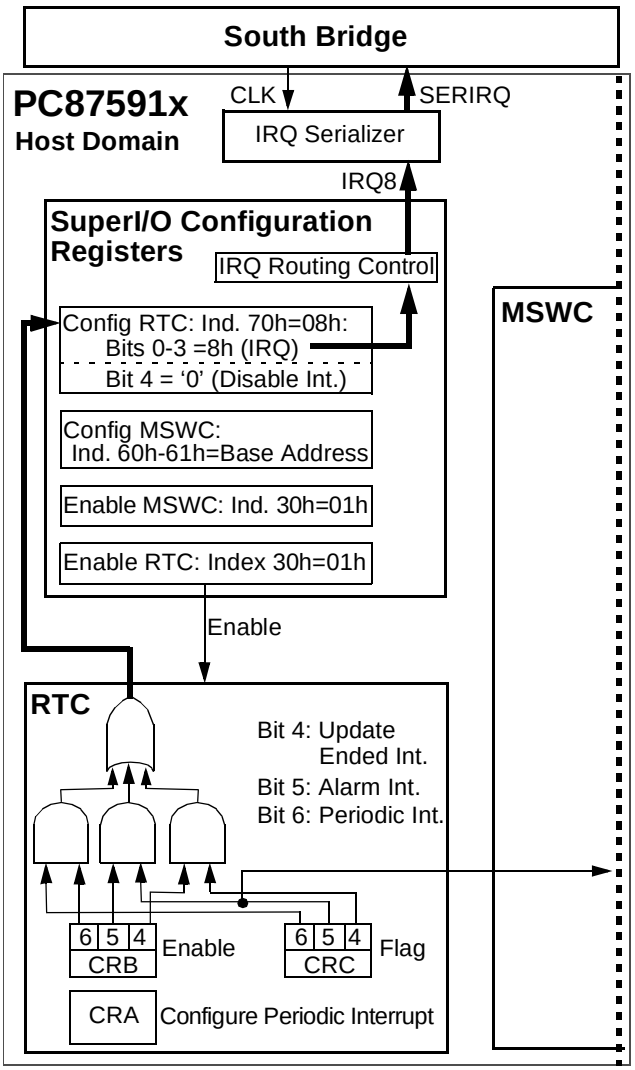
The concept and the global view of IRQ routing alternatives are the focus of this application note. Module details can be found in the relevant sections of the *PC87591E, PC87591S and PC87591L LPC Mobile Embedded Controllers Datasheet*.

2.0 Host Interrupt in Normal Operation

2.1 INTERRUPT GENERATION IN SYSTEM ACTIVE MODE (SLEEP STATES S0-S1)

Figure 2-1 shows the interrupt path in System Active mode during normal operation. The RTC is the interrupt source. The interrupt type and rate are configured in the CRA register. The CRB register enables the interrupts. The CRC register provides the status of the interrupt source.

The RTC output is routed to the host through the SuperI/O configuration registers, where the IRQ number is configured. The IRQ is then sent through the IRQ serializer to the SB.



Bold arrows show interrupt routing.

Figure 2-1. RTC Interrupt Generation: Active Mode

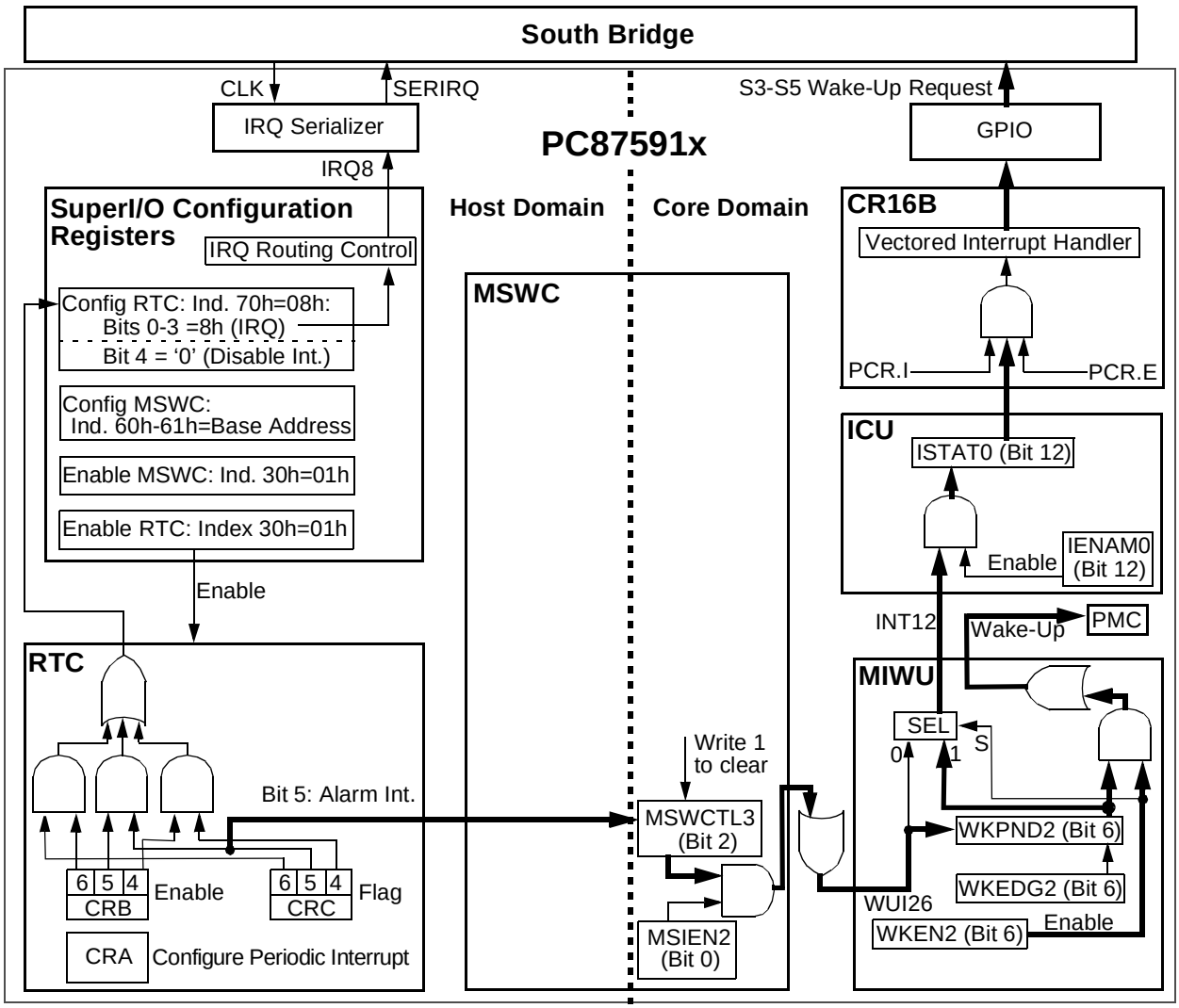
2.2 INTERRUPT GENERATION IN SYSTEM DEEP SLEEP MODE (SLEEP STATES S3-S5)

In System Deep Sleep mode, the LPC bus and the SERIRQ do not work after host domain modules are powered off. Thus an RTC alarm interrupt can not be sent via SERIRQ; instead, it must be sent via a wake-up command to the SB. Figure 2-2 shows the interrupt path.

When the RTC alarm interrupt occurs in Deep Sleep mode, the RTC output is routed to the core through the MSWC module. In the MSWC, the interrupt is enabled for routing to the MIWU module. In the MIWU, the interrupt is enabled as a wake-up event to the PMC, which wakes up the core from Idle mode and allows the core to handle the alarm.

From the MIWU, the signal is also sent to the ICU.

The ICU includes a mask register (IENAM0), which enables the interrupt. If the interrupt is enabled, the pending interrupt is reflected by the ICU status register (ISTAT0), and a maskable vectored interrupt is sent to the core. The core includes two interrupt enable bits: Global (PCR.I) and Local (PCR.E). Both should be enabled for the vectored interrupt to be accepted. The vectored interrupt handler is executed when the interrupt is accepted (i.e., the interrupt is enabled, acknowledged and has the highest priority in the ICU). The vectored interrupt handler can then perform wake-up events to the SB. For this, the code either utilizes a GPIO pin that connects to the SB wake-up input or uses the interrupt as part of the power management communication.



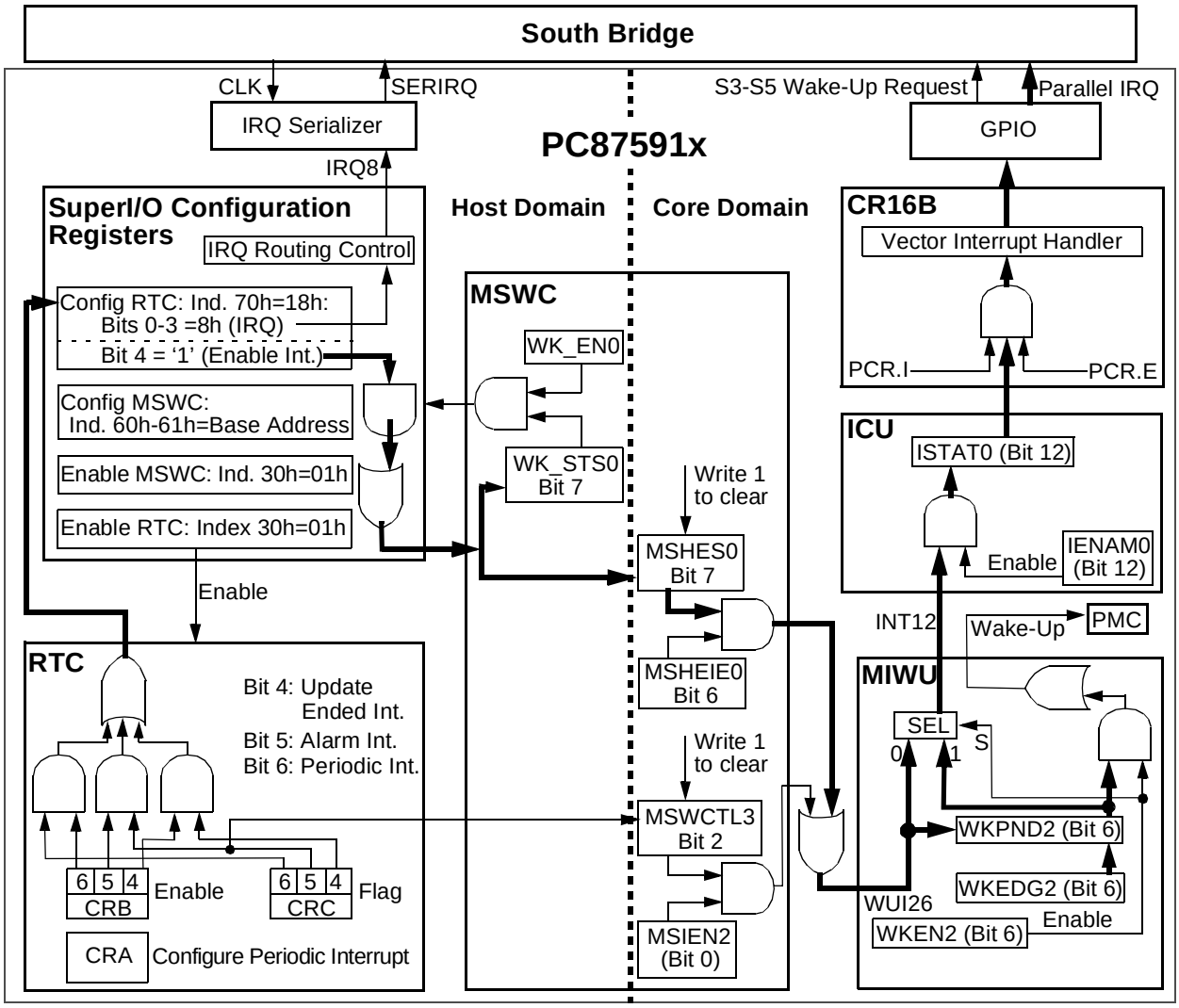
Bold arrows show interrupt routing.

Figure 2-2. RTC Alarm Interrupt Generation: Deep Sleep Mode

3.0 Host Interrupt Alternate Mode

If the SB does not accept a specific interrupt type via the PC87591x host domain, the interrupt can be routed via the core to a parallel input of the SB. Figure 3-1 shows the interrupt's alternate path.

The output of the RTC is a host IRQ, which is routed to the SuperI/O configuration registers, where the IRQ number is defined (typically set to be IRQ8). Alternatively the IRQ can be routed to the core through the MSWC when the Module Interrupt Enable bit is set. The MSWC receives a combined interrupt from all host domain modules in its Module Interrupt input. MSHES0 is a status bit that is set by this interrupt in the core domain, where it is enabled for routing to the MIWU. In the MIWU, the IRQ is enabled for routing to the ICU. The ICU includes a mask register (IENAM0), which enables the interrupt. If the interrupt is enabled, the pending interrupt is reflected by the ICU status register (ISTAT0), and a maskable vectored interrupt is sent to the core. The core accepts the interrupt, when enabled, as discussed in the previous section. The code in the vectored interrupt handle is used to generate any kind of event (including a Parallel IRQ) to the SB, using a GPIO pin.



Bold arrows show interrupt routing.

Figure 3-1. RTC Interrupt Generation: Alternate Mode

4.0 Appendix: Sample Code for Generating an RTC Interrupt

The sample code in this section generates an RTC periodic interrupt on a core GPIO signal.

The code includes two main parts:

- Initialization of the host registers (in the RTC logical device and SuperI/O configuration register). Note: This code sample uses the CR-to-SuperI/O Bridge (SIB); however, for a full-system implementation, initialization of the host registers should be done by the host BIOS code.
- A vector interrupt routine (handler) that accepts the RTC period interrupt and toggles a GPIO in response. In the sample code, the routine touches the RTC Status register to allow repetitive interrupt generation. This touch operation should be done by the host BIOS code. The pulse width should be set to allow detection by the host ICU (Note: Typically, a narrow falling edge is fine; however, if a wider pulse is required, one of the MFT outputs can be used in Single Pulse mode started by the routine).

The code uses the module IRQ routing to wake-up event mechanism, which connects the IRQ of the RTC to the MSWC. The MSWC can generate an interrupt to the core. The code uses the Active mode interrupt generation, with the MIWU in Alternate mode (i.e., bit 6 of WKEN2 register is set but bits 6 of WKPND2 and WKEDG2 registers are not). The MIWU module can also be put in Non-Alternate mode (i.e., bits 6 of WKEN2, WKPND2 and WKEDG2 are set) to allow passing the RTC interrupt over to the core when the device is idle.

Note: Code Comments appear in bold.

```
#include <asm.h>
#include "voyager.h"


unsigned char  SibRead ( unsigned char  offset );
void SibWrite ( unsigned char  offset , unsigned char data );


#pragma interrupt (IRQ18)  // IRQ18 = IRQ8 + enable bit.

// IRQ Generation Interrupt handler (in core), which responds to the MSWC event.
// The code, as written, assumes that the MIWU is in Alternate mode and the
// event is handled as a level event during Active or Wait power
// states of the core. Enable the MIWU (i.e., set bit 6 in WKEN2 register)
// and clear the pending bit for usage in Idle mode as well.

void IRQ18 (void)
{
   unsigned int wait;
   unsigned char temp;

  // Clear GPIO IOPD.6 to '0'
     PDDOUT &= ~0x40;


  // Clear event in MSWC.
     MSHES0=MSHES0 ; // Clear Module IRQ Event Status.

  // Here, add MIWU status clear if used in Idle mode.

#ifdef NoBIOS

      // The following lines are for standalone (non-BIOS) operation only!!
      // The pending flag in the RTC is cleared to enable additional
      // interrupts.
      //
      // Note: In a full-system implementation, this operation
      // should be done by the BIOS.
```

```
    SIBCTRL|=0x01;  // Enable core access to SuperI/O modules.
    LKSIOHA=0x0002; // Lock configuration register host access.
    CRSMAE=0x0002;  // Select RTC.

    SibWrite (0x00,0x0c); // Select CRC via RTC index register.
    SibRead (0x01); // Clear IRQ in RTC by reading CRC register.

      // For PC87591E and PC87591S Rev B2,
      // uncomment the two following lines:
      // CRSMAE=0x0100;
      // SibWrite (0x00,0x80);

    CRSMAE  = 0x0000; // Disable RTC register accesses.
    LKSIOHA = 0x0000; // Enable host access.
    SIBCTRL = 0x00;   // Disable core access to SuperI/O modules.


#endifdef // NoBIOS. End of host replacement code (for standalone non-BIOS
          // operation).

      // Toggle the GPIO IOPD.6 back to '1'.
      // Use a 'wait loop' to control the pulse width, or use an MFT timer
      // in Single Pulse mode to generate a wide enough pulse if the wait
      // loop is not desired.
  PDDOUT |=0x40;

}// End of IRQ Generation Interrupt handler.

// Host domain register initialization (RTC and SuperI/O configuration registers).

int main(void)
 {

  void * my_intbase;
  unsigned int my_psr;
  unsigned char temp;

  // Interrupt configuration
  // Disable interrupts.
  _disable_();

  // Install interrupt handler.

  _spr_("intbase", my_intbase);
  ((void **) (my_intbase))[28] =(void *)IRQ18; // 16(base)+ int 12 (WUI26 -> MSWC
                                               // wake-up).

  // Configure ICU.
  IECLR0 = 0xffff;
  IECLR1 = 0xffff;
  // Enable interrupt 12 in ICU.
  IENAM0 |= 0x1000;

  // Here, if the MIWU is used in Non-Alternate mode, enable the MIWU.

  // Enable global maskable interrupts.
  _spr_("psr", my_psr);
```

```
  my_psr |= 0x800;
  _lpr_("psr", my_psr);

   // Set GPIO IOPD.6 to output default at '1'.
   PDDOUT |=0x40;
   PDDIR |=0x40; // PC for output.

   // MSWC enables the generation of an interrupt on Module IRQ event.
   MSHEIE0 |= 0x80; // Module IRQ Event Enable.

   // Configuration of RTC and Host configuration registers
   // to enable periodic interrupts and RTC interrupt
   // routing to the MSWC.
   // Note: The following lines are for standalone (non-BIOS) operation only; in
   // a full-system implementation, this operation should be done by host BIOS.

   //----------------------------------------
   // Enable the CR-to-SIB for read and write.
   //----------------------------------------
        SIBCTRL|=0x01;  // Enable core access to SuperI/O modules.
        LKSIOHA=0x0001; // Lock configuration register host access.
        CRSMAE=0x0001;  // Select configuration registers.

        // Select RTC logical device.
            SibWrite (0x00,0x07);
            SibWrite (0x01,0x10);

#ifdef Debug

      // Debug code only:
      // Relocate RTC to 0x370, 0x371, 0x372, 0x373 for access in
      // systems that have an additional RTC.
        SibWrite (0x00,0x60);
        SibWrite (0x01,0x03);
        SibWrite (0x00,0x62);
        SibWrite (0x01,0x03);

#endif //Debug

        // Configure RTC IRQ to INT8 and enable generation of
        // module IRQ wake-up event for it.
        // Note: Do not enable the following events for other source code
        //       in order not to lose the events.
        SibWrite (0x00,0x70);
        SibWrite (0x01,0x18);

        // Enable RTC logical device.
        SibWrite (0x00,0x30);
        SibWrite (0x01,0x01);

        // Select MSWC module.
        SibWrite (0x00,0x07);
        SibWrite (0x01,0x04);

        // Configure MSWC base address to 0x360 (select address
        // as required to avoid conflict with other resources).
        SibWrite (0x00,0x60);
        SibWrite (0x01,0x03);
```

```
        SibWrite (0x00,0x61);
        SibWrite (0x01,0x60);

        // Enable MSWC module.
        SibWrite (0x00,0x30);
        SibWrite (0x01,0x01);

        // Enable RTC to generate periodic interrupt.
        CRSMAE=0x0002; // Select RTC registers.

        // Configure RTC
        SibWrite (0x00,0x0d);
        SibRead  (0x01);       // Clear VRT bit.
        SibWrite (0x00,0x0a);
        SibWrite (0x01,0x26);  // Start PC timer for periodic interrupt.
        SibWrite (0x00,0x0b);
        SibWrite (0x01,0x40);  // Enable periodic interrupt.
        SibWrite (0x00,0x0c);
        SibRead  (0x01);       // Read the Status register to restart the
                             // interrupt.

        // Disable CR-to-SIB (and enable host access to devices).
        CRSMAE  =0x0000; // Disable RTC register access.
        LKSIOHA =0x0000; // Enable host access.
        SIBCTRL =0x0000; // Disable core access to SuperI/O modules.

// Enable interrupt in core.
   _enable_();

// Main program waits here for periodic interrupts.
while (1);

}

//-------------------------------------------------

// CR-to-SIB access routines.
/*-----------------------------------------------------------*/
void SibWrite ( unsigned char  offset , unsigned char data )
{
        IHIOA=offset;
        IHD=data;
        while (SIBCTRL & 0x04 ); // Wait for write operation to end.
}
/*-----------------------------------------------------------*/
/*-----------------------------------------------------------*/
unsigned char  SibRead ( unsigned char  offset )
{
      unsigned char data;

        IHIOA=offset;
        SIBCTRL|=0x02; // Start Read.
        while (SIBCTRL & 0x02 ); // Wait for read operation to end.
        data=IHD;

      return;
}
/*-----------------------------------------------------------*/
```

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Email:
new.feedback@nsc.com

National Semiconductor
Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 87 90

National Semiconductor
Asia Pacific Customer
Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor
Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507
Email: nsj.crc@jksmtp.nsc.com

www.national.com