Introduction

This document details the design and usage scenarios for the ATSAMB11 peripheral interrupts and ULP feature.
# Table of Contents

Introduction......................................................................................................................1

1. Overview....................................................................................................................3

2. Interrupts................................................................................................................... 4
   2.1. Interrupt Vector Map.............................................................................................4
   2.2. Task Background....................................................................................................4
   2.3. Peripheral Interrupts.............................................................................................4

3. Power Management.................................................................................................. 8
   3.1. Wake-up Sources.................................................................................................8
   3.2. ULP API................................................................................................................9
   3.3. LP_GPIO Pin Status.............................................................................................9

4. Document Revision History......................................................................................11

The Microchip Web Site................................................................................................ 12

Customer Change Notification Service........................................................................12

Customer Support......................................................................................................... 12

Microchip Devices Code Protection Feature................................................................. 12

Legal Notice...................................................................................................................13

Trademarks................................................................................................................... 13

Quality Management System Certified by DNV.............................................................14

Worldwide Sales and Service........................................................................................15
1. **Overview**

This document describes the peripheral interrupt functionality in ATSAMB11. In addition, it provides a sample guide for how to implement interrupt handling for GPIO and UART, including details on how to handle those interrupts in the application written for the ATSAMB11 platform. Finally, this document details the ULP feature and its enable/disable functionality.
2. **Interrupts**

Refer to the ATSAMB11XR/ZR Ultra-Low Power BLE SiP/Module Datasheet.

2.1 **Interrupt Vector Map**

ATSAMB11 platform has an ARM Cortex®-M0 CPU capable of running at 26 MHz. The Cortex-M0 NVIC is connected to the above 32 IRQ sources. Some of the interrupt sources are marked reserved for internal operation of the BLE functionality and ROM code of the ATSAMB11. Each word represents an ISR address.

2.2 **Task Background**

ATSAMB11 supports multiple tasks.

- Idle Task – idle task is responsible for ULP sleep feature of ATSAMB11. It is the lowest priority task.
- FW Task – firmware task responsible for BLE functionality. It is the highest priority task.
- Application Task – ATSAMB11 application task. Application developed for ATSAMB11 run as an application task. Its priority is higher than idle task but lower than the FW task.

On boot-up the firmware task is initialized, started, and then waits for any event to happen. If any application is loaded onto the platform, it will start the application task followed by the user developed application. When both tasks are waiting for event/interrupt, the system goes to idle task where it will try to put the entire system into ULP Sleep state.

2.3 **Peripheral Interrupts**

In order for users to develop the application, the ATSAMB11 development platform is provided with the two libraries listed below.

- driver_lib.lib – platform layer library providing API for access to peripherals available on the platform and API for power management of ATSAMB11.
  
  **Note:** This is provided as a library for Keil® IDE. ASF in Atmel Studio has the driver implemented for all supported peripherals in ATSAMB11. ASF drivers are available as source code.

- ble_api.lib – library providing API for supporting BLE functionality.

The driver library provides access to the peripherals available on the platform. Each peripheral exports API to register, enable and disable interrupts supported by those peripherals. The following sections explain and provide examples for how to use interrupts on the ATSAMB11 platform.

2.3.1 **Interrupt Mechanism**

The ATSAMB11 driver offers simple APIs to manage interrupts generated by the set of available GPIOs and peripherals on ATSAMB11. For these interrupts, the developer is allowed to register an interrupt callback that is to be executed upon triggering the configured interrupt source. Callbacks are executed from ISR context and then this interrupt event is messaged to the application task.

**Note:** Callbacks are executed from ISR context, so it is important not to execute long routines, add delays, or print trace messages inside the interrupt callback functions as it will impact the timing of FW task. Instead, post a platform event to the main event handling loop and add any routines inside the platform event handler. These interrupts are sent as a platform event to the application task to handle it in task context.
2.3.2 GPIO Interrupt Management

The following example code shows how to configure a GPIO 23 that is connected to a press button, and to trigger an interrupt on a button press event. When the button is pressed, the callback function is executed and a platform event is sent to the application task.

Once the interrupt occurs, it is deferred to the application task context by sending a message from the callback function. This message is handled in the application task context as mentioned in the following code.

```c
#include <asf.h>
#include "at_ble_api.h"
#include "platform.h"
#include "gpio.h"
#define APP_STACK_SIZE (2048)
volatile unsigned char app_stack_patch[APP_STACK_SIZE];
#define BUTTON_0_PIN 23 // button connected on GPIO 23
static void button_callback(void)
{
    send_plf_int_msg_ind(BUTTON_0_PIN,GPIO_CALLBACK_RISING,NULL,0);
}
static void gpio_button_init(gpio_callback_t callback)
{
    struct gpio_config config_gpio_pin;
    gpio_get_config_defaults(&config_gpio_pin);
    config_gpio_pin.direction  = GPIO_PIN_DIR_INPUT;
    config_gpio_pin.input_pull = GPIO_PIN_PULL_NONE;
    gpio_pin_set_config(BUTTON_0_PIN, &config_gpio_pin);
    gpio_init();
    gpio_register_callback(BUTTON_0_PIN, callback, GPIO_CALLBACK_RISING);
    gpio_enable_callback(BUTTON_0_PIN);
}
static void resume_cb(void)
{
    init_port_list();
}
int main(void)
{
    uint16_t plf_event_type;
    uint8_t plf_event_data[16];
    uint16_t plf_event_data_len;
    platform_driver_init();
    gpio_button_init(button_callback);
    register_resume_callback(resume_cb);
    acquire_sleep_lock();
    while(platform_event_get(&plf_event_type,plf_event_data,&plf_event_data_len))
    {
        if(plf_event_type == ((GPIO_CALLBACK_RISING << 8)| BUTTON_0_PIN))
        {
            //Do what you wanted to do on button press event can be handled here.
            volatile uint8_t variable_for_breakpoint = 0;
        }
    }
    return 0;
}
```

Note:
Platform_driver_init () function must be called before using driver different APIs. If at_ble_init is called, do not call platform_driver_init () as it is implicitly called inside BLE init.

2.3.3 UART Interrupt Management

ATSAMB11 driver offers simple APIs to manage a UART interface. The developer is capable of registering an interrupt callback to be executed upon receiving data. The following example code shows how to configure a UART to interrupt when the receive buffer is filled with 4 bytes of data.

Once the interrupt occurs, it is deferred to the application task context by sending a message from the callback function. This message is handled in the application task context as shown in the following code.

```c
#include "at_ble_api.h"
#include <asf.h>
#include <string.h>
#include <conf_uart_serial.h>
#include <uart.h>
#include "platform.h"
#define UART_CALLBACK 0x11
#define APP_STACK_SIZE (1024)
volatile unsigned char app_stack_patch[APP_STACK_SIZE];
struct uart_module uart_instance;
static uint8_t uart_rx_input[5] = {0, };

static void configure_uart(void)
{
    struct uart_config config_uart;
    system_clock_config(CLOCK_RESOURCE_XO_26_MHZ, CLOCK_FREQ_26_MHZ);
    uart_get_config_defaults(&config_uart);
    config_uart.baud_rate = 115200;
    config_uart.pin_number_pad[0] = EDBG_CDC_SERCOM_PIN_PAD0;
    config_uart.pin_number_pad[1] = EDBG_CDC_SERCOM_PIN_PAD1;
    config_uart.pin_number_pad[2] = EDBG_CDC_SERCOM_PIN_PAD2;
    config_uart.pin_number_pad[3] = EDBG_CDC_SERCOM_PIN_PAD3;
    config_uart.pinmux_sel_pad[0] = EDBG_CDC_SERCOM_MUX_PAD0;
    config_uart.pinmux_sel_pad[1] = EDBG_CDC_SERCOM_MUX_PAD1;
    config_uart.pinmux_sel_pad[2] = EDBG_CDC_SERCOM_MUX_PAD2;
    config_uart.pinmux_sel_pad[3] = EDBG_CDC_SERCOM_MUX_PAD3;
    stdio_serial_init(&uart_instance, CONF_STDIO_USART_MODULE, &config_uart);
}

static void register_uart_rx_callback(uart_callback_t callback_func)
{
    uart_register_callback(&uart_instance, callback_func, UART_RX_COMPLETE);
    uart_enable_callback(&uart_instance, UART_RX_COMPLETE);
}

static void uart_rx_callback(struct uart_module *const module)
{
    send_plf_int_msg_ind(UART_CALLBACK, UART_RX_COMPLETE, NULL, 0);
}

int main(void)
{
    uint16_t plf_event_type;
    uint8_t plf_event_data[16];
    uint16_t plf_event_data_len;
    platform_driver_init();
    acquire_sleep_lock();
}
configure_uart();
register_uart_rx_callback(uart_rx_callback);
uart_read_buffer_job(&uart_instance, uart_rx_input, 4);

while(platform_event_get(&plf_event_type,plf_event_data,&plf_event_data_len))
{
    if(plf_event_type == ((UART_RX_COMPLETE << 8) | UART_CALLBACK))
    {
        //Do what you wanted to do on receiving 4 characters over UART.
        printf("uart_rx_input = \%s\r\n", uart_rx_input);
        uart_read_buffer_job(&uart_instance, uart_rx_input, 4);
    }
}

return 0;

---

**Important:** Prolonged IRQ handler execution will lead to unexpected reset with message BOUT. For more details, refer to ATSAMB11 BluSDK Smart Release Notes 6.2 and above, in Section 11.4 Recommended Code Implementation for Handling Platform Event.
### Power Management

ATSAMB11 supports the following sleep states for power saving.

- **Normal Sleep/WFI – Wait for Interrupt.** In this state, the ARM Cortex-M0 is clock gated and just waiting for interrupt. BLE core, GPIOs, and other peripheral interface cores such as I2C, SPI, and UART are still powered and functional.

- **ULP sleep state – Ultra-Low Power sleep state.** This is the lowest possible power state the system can go. In this state, ARM Cortex-M0, BLE core, GPIO’s, and all other peripheral cores such as I2C, SPI, and UART are all powered-down. Only AO_GPIO_0 and AON-Sleep timer are functional. Before entering this state, idle task saves the context and it is retained in a self-refreshing memory region. On wake-up idle task restores the context and continues execution.

### 3.1 Wake-up Sources

ATSAMB11 has the following wake-up sources that can wake-up the system from ULP mode.

- BLE events
- AO_GPIO_0
- AON-Sleep Timer expiry

AO_GPIO_0 is HW dominant factor in wake-up/sleep operations. When the application is blocked waiting for the next event and BLE is idle, the ATSAMB11 can go to ULP. Wake-up is caused by one of the previous mentioned sources.

AO_GPIO_0 is a special case in sleep/wake-up. When AO_GPIO_0 is set to high (logic 1), it shall issue a wake-up operation and cause ATSAMB11 to leave ULP state into active state. As long as AO_GPIO_0 is high (logic 1), it will prohibit ATSAMB11 of going to ULP. If AO_GPIO_0 is high, ARM will remain on, and all peripherals are in powered state. When AO_GPIO_0 becomes low (logic 0) again, ULP is able to regain sleep control and decides to go ULP based on BLE and application activities.

Techniques to prevent ATSAMB11 from going to ULP state:

- set AO_GPIO_0 to high
- use acquire_sleep_lock() API from application

**Note:** AO_GPIO_0 must never be left floating as it produces undefined behavior and introduces instabilities.

The following table describes the different states given the different inputs of wake-up and sleep sources.

<table>
<thead>
<tr>
<th>AO_GPIO_0 state</th>
<th>No API called</th>
<th>Call to acquire_sleep_lock()</th>
<th>Call to release_sleep_lock()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic 0</td>
<td>Prevent ULP</td>
<td>Prevent ULP</td>
<td>Enter ULP</td>
</tr>
<tr>
<td>Logic 1</td>
<td>Prevent ULP</td>
<td>Prevent ULP</td>
<td>Prevent ULP</td>
</tr>
<tr>
<td>Left unconnected</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
</tr>
</tbody>
</table>
3.2 ULP API

An application developed on ATSAMB11 can utilize the ULP feature in order to save power. The ATSAMB11 platform supports the following API's to block and unblock ULP mode by acquiring the sleep lock and releasing the sleep lock respectively:

- **acquire_sleep_lock()** – acquires the sleep lock and thereby disables or blocks the system from entering into ULP mode.
- **release_sleep_lock()** – releases the sleep lock and thereby enables or unblocks the system from entering into ULP mode.
- **register_resume_callback (cb)** – registers a callback function that will be called when the system resumes from ULP sleep state.

The application can register a resume callback function in order to use peripherals in ATSAMB11 along with the ULP feature. Since these peripherals are powered-down when entering ULP mode, it needs to be re-initialized before being used after resume operation. This callback provides an opportunity for the application tasks to set up their peripheral interfaces before the application task gets an opportunity to execute.

**acquire_sleep_lock()** and **release_sleep_lock()** is managed by ble_manager.c.

- **ble_set_ulp_mode(BLE_ULP_MODE_CLEAR)** - if this API is called before while() in main(), ble_manager.c cannot allow the device to go to ULP mode by calling acquire_sleep_lock().
- **ble_set_ulp_mode(BLE_ULP_MODE_SET)** - if this API is called before while() in main(), ble_manager.c can control the ULP mode.

**Note:** Peripherals that have to drive a GPIO to a certain level and maintain the output cannot operate while ULP mode is enabled.

Refer to the iBeacon™ App that utilizes the ULP feature along with UART interface that is used for logging.

3.3 LP_GPIO Pin Status

ATSAMB11 has LP_GPIOs. Some of the GPIOs are not available for application usage. The rest of the GPIOs are available for applications. These GPIOs can be used either as regular GPIOs or can be reconfigured to any other peripheral cores available.

The GPIOs are controlled by GPIO core, which is in a power domain and will again be turned off when the system enters low-power mode. This is why the GPIO controller or peripheral cores using those GPIOs during wake-up sequence need to be reconfigured. This can be achieved by doing the necessary initialization in the apps_resume() callback function, which is registered using register_resume_callback() API; however, ATSAMB11 provides a way to maintain logical states on those GPIOs even when the controller is turned off during the ULP mode.

This GPIO state is retained using latches on the ATSAMB11 pads. These GPIOs can be held in the required logic state during the sleep duration. When the resume happens the GPIO can be reconfigured to the state before it went to ULP mode, and after that the latch in the pads are turned off. Now the GPIO is again controlled by the GPIO cores and it can be used as normal.

The following figure illustrates the sequence of steps followed to retain the GPIO status when entering ULP.
The following figure illustrates the sequence of steps to follow in order to maintain the same state on GPIO.

**Figure 3-1. GPIO status when entering ULP**

1. Turn on latches for ATSAMB11 pads.
2. Turn off GPIO controllers and other peripheral cores.
3. Turn off all Power domains except the AON domain. System enters ULP mode.

**Figure 3-2. Maintain the Same State on GPIO**

1. ARM wakes up from power down mode.
2. GPIO controller and other peripheral cores are turned on and in default state.
3. ARM calls the application registered callback “apps_resume”. The application can reconfigure the GPIOs or the peripheral cores to a similar state prior to going into ULP mode.
4. ARM disables the latch on the ATSAMB11 pads.
# Document Revision History

Table 4-1. Revision History

<table>
<thead>
<tr>
<th>Doc Rev</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS50002676B</td>
<td>4/2018</td>
<td>• Revised contents to match various Bluetooth Low Energy standard versions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated 3.2 ULP API.</td>
</tr>
<tr>
<td>DS50002676A</td>
<td>9/2017</td>
<td>Updated from Atmel to Microchip template and assigned new Microchip number.</td>
</tr>
<tr>
<td>42663B</td>
<td>03/2016</td>
<td>Section 3.3 LP_GPIO Pin Status added.</td>
</tr>
<tr>
<td>42663A</td>
<td>02/2016</td>
<td>Initial document release.</td>
</tr>
</tbody>
</table>
The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user’s guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip’s customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.


Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.


SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.
Quality Management System Certified by DNV

ISO/TS 16949
Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company’s quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001:2000 certified.
<table>
<thead>
<tr>
<th>AMERICAS</th>
<th>ASIA/PACIFIC</th>
<th>ASIA/PACIFIC</th>
<th>EUROPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate Office</td>
<td>Australia - Sydney</td>
<td>Tel: 61-2-9868-6733</td>
<td></td>
</tr>
<tr>
<td>2355 West Chandler Blvd.</td>
<td>China - Beijing</td>
<td>Tel: 86-10-8569-7000</td>
<td></td>
</tr>
<tr>
<td>Chandler, AZ 85224-6199</td>
<td>China - Chengdu</td>
<td>Tel: 86-28-8665-5511</td>
<td></td>
</tr>
<tr>
<td>Tel: 480-792-7200</td>
<td>China - Chongqing</td>
<td>Tel: 86-23-8980-9588</td>
<td></td>
</tr>
<tr>
<td>Fax: 480-792-7277</td>
<td>China - Dongguan</td>
<td>Tel: 86-769-8702-9880</td>
<td></td>
</tr>
<tr>
<td>Technical Support:</td>
<td>China - Guangzhou</td>
<td>Tel: 86-20-8755-8029</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.microchip.com/support">http://www.microchip.com/support</a></td>
<td>China - Hangzhou</td>
<td>Tel: 86-571-8792-8115</td>
<td></td>
</tr>
<tr>
<td>Web Address:</td>
<td>China - Hong Kong SAR</td>
<td>Tel: 852-2943-5100</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.microchip.com">www.microchip.com</a></td>
<td>China - Nanjing</td>
<td>Tel: 86-25-8473-2460</td>
<td></td>
</tr>
<tr>
<td>Atlanta</td>
<td>China - Qingdao</td>
<td>Tel: 86-632-8502-7355</td>
<td></td>
</tr>
<tr>
<td>Duluth, GA</td>
<td>China - Shanghai</td>
<td>Tel: 86-21-3326-8000</td>
<td></td>
</tr>
<tr>
<td>Tel: 678-957-9614</td>
<td>China - Shenyang</td>
<td>Tel: 86-24-2334-2829</td>
<td></td>
</tr>
<tr>
<td>Fax: 678-957-1455</td>
<td>China - Shenzhen</td>
<td>Tel: 86-755-8864-2200</td>
<td></td>
</tr>
<tr>
<td>Austin, TX</td>
<td>China - Suzhou</td>
<td>Tel: 86-186-6233-1526</td>
<td></td>
</tr>
<tr>
<td>Westborough, MA</td>
<td>China - Wuhan</td>
<td>Tel: 86-27-5980-5300</td>
<td></td>
</tr>
<tr>
<td>Tel: 774-760-0087</td>
<td>China - Xian</td>
<td>Tel: 86-29-8833-7252</td>
<td></td>
</tr>
<tr>
<td>Fax: 774-760-0088</td>
<td>China - Xiamen</td>
<td>Tel: 86-992-2388138</td>
<td></td>
</tr>
<tr>
<td>Chicago</td>
<td>China - Zuhai</td>
<td>Tel: 86-756-3210040</td>
<td></td>
</tr>
<tr>
<td>Itasca, IL</td>
<td>India - Bangalore</td>
<td>Tel: 91-60-3090-4444</td>
<td></td>
</tr>
<tr>
<td>Tel: 630-285-0071</td>
<td>India - New Delhi</td>
<td>Tel: 91-11-4160-8631</td>
<td></td>
</tr>
<tr>
<td>Fax: 630-285-0075</td>
<td>India - Pune</td>
<td>Tel: 91-20-4121-0141</td>
<td></td>
</tr>
<tr>
<td>Dallas</td>
<td>Japan - Osaka</td>
<td>Tel: 81-6-6152-7160</td>
<td></td>
</tr>
<tr>
<td>Addison, TX</td>
<td>Japan - Tokyo</td>
<td>Tel: 81-3-6880-3770</td>
<td></td>
</tr>
<tr>
<td>Tel: 972-818-7423</td>
<td>Korea - Daegu</td>
<td>Tel: 82-53-744-4301</td>
<td></td>
</tr>
<tr>
<td>Fax: 972-818-2924</td>
<td>Korea - Seoul</td>
<td>Tel: 82-2-554-7200</td>
<td></td>
</tr>
<tr>
<td>Detroit</td>
<td>Malaysia - Kuala Lumpur</td>
<td>Tel: 60-3-7651-7906</td>
<td></td>
</tr>
<tr>
<td>Novi, MI</td>
<td>Malaysia - Penang</td>
<td>Tel: 60-4-227-8870</td>
<td></td>
</tr>
<tr>
<td>Tel: 248-848-4000</td>
<td>Philippines - Manila</td>
<td>Tel: 63-2-634-9065</td>
<td></td>
</tr>
<tr>
<td>Houston, TX</td>
<td>Singapore</td>
<td>Tel: 65-6334-8870</td>
<td></td>
</tr>
<tr>
<td>Tel: 281-894-5983</td>
<td>Taiwan - Hsin Chu</td>
<td>Tel: 886-3-577-8366</td>
<td></td>
</tr>
<tr>
<td>Tel: 214-641-5000</td>
<td>Taiwan - Kaohsiung</td>
<td>Tel: 886-7-213-7830</td>
<td></td>
</tr>
<tr>
<td>Los Angeles</td>
<td>Taiwan - Taipei</td>
<td>Tel: 886-2-2508-8600</td>
<td></td>
</tr>
<tr>
<td>Tel: 317-773-8323</td>
<td>Thailand - Bangkok</td>
<td>Tel: 66-2-694-1351</td>
<td></td>
</tr>
<tr>
<td>Fax: 317-773-5453</td>
<td>Vietnam - Ho Chi Minh</td>
<td>Tel: 84-28-5448-2100</td>
<td></td>
</tr>
<tr>
<td>Tel: 317-536-2380</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mission Viejo, CA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tel: 949-462-9523</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tel: 949-462-9608</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Raleigh, NC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tel: 919-844-7510</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New York, NY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tel: 631-435-6000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>San Jose, CA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tel: 408-735-9110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tel: 408-436-4270</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Canada - Toronto</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tel: 905-695-1980</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fax: 905-695-2078</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>