AN1412

How to Calculate UNIX® Time Using a PIC18 Microcontroller and the MCP795W20 SPI RTCC

Author: Eugen Ionescu
Microchip Technology Inc.

INTRODUCTION

This application note is a UNIX® time conversion tutorial, compiled with the Daylight Saving Time standard. The application can be used as a UNIX tutorial or as a standard electronic watch, using the PIC18 demo board and the MCP795W20 RTCC device.

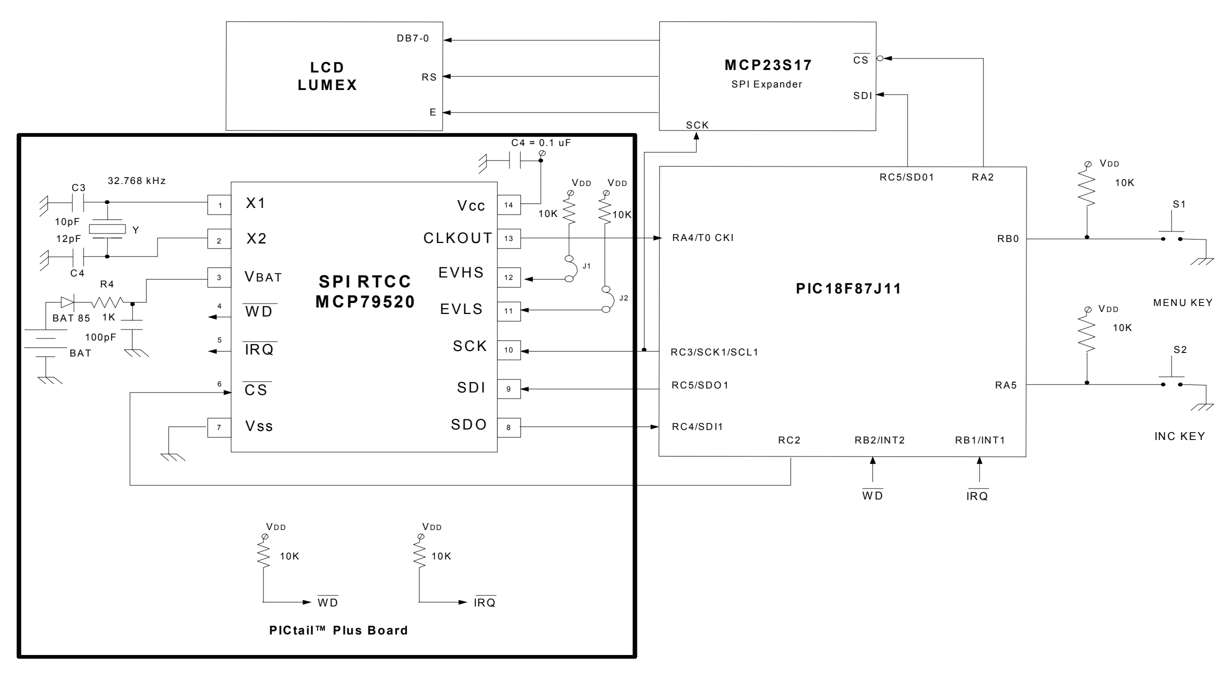
FEATURES OF THE RTCC STRUCTURE

- Real-Time Clock/Calendar:
  - Hours, Minutes, Seconds, Hundredths of Seconds, Day of Week, Month, and Year
  - Support for Leap year
- Leap Year Calculation up to 2399
- Time-Stamp Function
- 2 Kbit (256 x 8) EEPROM Memory
- 64-Byte x 8 Organization Battery Backed SRAM
- Input for External Battery Backup
- On-Board Crystal Oscillator for RTCC Functions:
  - Battery operated when VCC removed
  - Operated down to 1.3V to maximize battery life
  - Requires external 32,768 kHz tuning fork crystal
- Clock Out Function:
  - 1Hz
  - 4.096 kHz
  - 8.192 kHz
  - 32.768 kHz
- Two Programmable Alarms:
  - Open-drain alarm/interrupt pin
  - Programmable to IRQ or WD pin
- 64-Bit Unique ID in Protected Area:
  - Support EUI-48/64
  - Separate unlock sequence
  - Factory or user programmed
- Programmable Watchdog Timer:
  - Dedicated open-drain Watchdog output pin
  - Reset over the SPI interface or I/O input (Event Detect)
- On-Board Event Detection:
  - Dual configurable inputs
  - High-Speed Digital Event detection, on 1, 4, 16 or 32nd event, (glitch filter)
  - Low-speed detection with programmable debounce time
  - Operates from VBAT when VCC removed
  - Edge triggered (rising or falling)
- On-Chip Digital Trimming/Calibration:
  - Single point calibration
  - +/- 256 bits of calibration
- Sequential Read of all Memory
- Software Block Write Protection for ¼, ½, or Entire Array

SCHEMATIC

The schematic includes a PIC18 Explorer demo board and the SPI RTCC PICtail™ daughter board as shown in Figure 1.

FIGURE 1: SCHEMATIC



The hardware modules used on the demo board are:

- LCD
- 2 push buttons
- SPI RTCC PICtail™ daughter board

To access the LCD through a minimum of pins, the SPI on the MSSP1 module is used, in conjunction with a 16-bit I/O expander with SPI interface (MCP23S17). The two on-board push buttons are S1 and S2, connected to RB0, RA5 GPIOs. The SPI RTCC is part of the RTCC PICtail evaluation board and is directly connected to the MSSP1 module of the MCU.

The RTCC PICtail daughter board has two other components:

- a 32,768 Hz crystal driving the internal clock of the RTCC
- a 3-volt battery sustaining the RTCC when VDD is not present on the demo board

DETAILS ABOUT IMPLEMENTATION

The application implements a UNIX Time Tutorial, showing how to convert your date and time to UNIX time-stamp.

The application is performed on a PIC18 Explorer demo board on which is mounted a PIC18F87J11 MCU. The code is written in C using the C18 compiler.

FUNCTIONAL DESCRIPTION

The MCP795W20 is an SPI slave device, working on the related unidirectional 4-wire bus. SDI and SDO are pins used to transfer addresses and data in and out of the device. For normal data transfers, the $\overline{CS}$ pin must be set to `0` by the master device. SCK input is used to synchronize the data transfer from and to the device. The related internal structures have the following device addresses/control bytes (the RTCC is included in the SRAM bank):

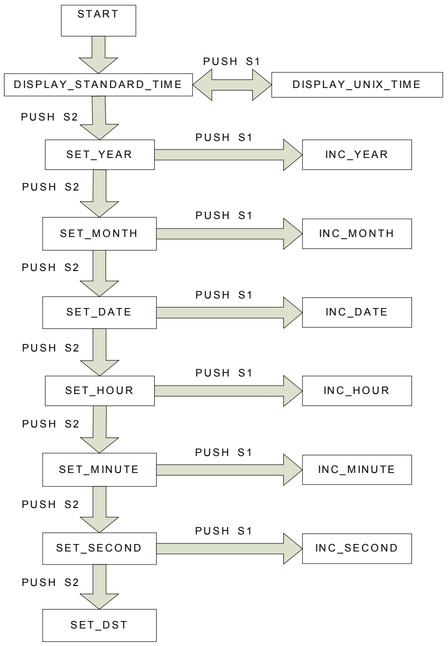
- RTCC + SRAM: 0x12 for writes, 0x13 for reads
- EEPROM: 0x02 for writes, 0x03 for reads

APPLICATION DESCRIPTION

This application performs a UNIX Time Tutorial. At start-up, the standard time is displayed on the on-board LCD.

The S1 push button changes the displaying mode from standard to UNIX time and vice versa. The S2 push button allows the setting of the date and time. After all the time variables are set, the user must decide if the Daylight Savings Time for the current location has started or not.

FIGURE 2: APPLICATION FLOWCHART



Unix Time Conversion

The UNIX Time conversion is developed inside the `unsigned long unixtime(DateTime crtime, char * local, unsigned char DST)` function. The `crtime` variable stores the current date and time.

```
typedef struct
{
int sec,min,hr;
int year,month,date;
}DateTime;
```

The local variable stores the name for the current location and it must have one of the values stored into the timezone[][] array – see below. The DST variable indicates if the current location sees Daylight Savings Time or not. If the DST is set, the time difference between the current location and UTC is the typical time difference + 1 hour.

To calculate the time difference between two locations, the application uses a time zone list.

FIGURE 3: TIME ZONE LIST

Time Zone	Difference Between UTC
CHANDLER	-7 hours/No DST
NEW YORK	-5 hours/DST
UTC	+0 hours/No DST
LONDON	+0 hours/DST
PARIS	+1 hours/DST
BERN	+1 hours/DST
BUCHAREST	+2 hours/DST
MOSCOW	+3 hours/DST
NEW DELHI	+5:30 hours/No DST
BANGKOK	+7 hours/No DST
BEIJING	+8 hours/No DST
TOKYO	+9 hours/No DST

The previous table is defined in firmware like two arrays.

EXAMPLE 1: DEFINE TIME ZONES AND TIME DIFFERENCE INSIDE THE FIRMWARE

```
 char timezone[NMAXZONE][10] = {"Chda","NY","UTC","London","Paris", "Bern","Buch","Moscow",
                                "Delhi","Bang","Beij","Tokyo"};
float timevalue[2][NMAXZONE] = {{-7,-5,0,0,1,1,2,3,5.5,7,8,9},
                                // stores the time difference between   UTC/GMT and another
                                locations
                                {0,1,0,1,1,1,1,1,0,0,0,0}}
                                // the second row stores which locations observe Daylight
                                Saving time
                                // (a 5.5 hours difference = 5 hours and 30 minutes)
```

The timezone[][] array stores the locations from the time zone's map. The timevalue[][] array stores on the first row, the time difference between UTC and another location. The second row indicates which locations see the Daylight Savings Time.

The `unixtime` (`DateTime crtime, char *local, unsigned char DST`) function stores how many seconds passed from 1.1.1970, 00:00:00 AM until now. The following code calculates first, the number of seconds totaled from the number of days in the current month (a day has 86400 seconds). Then, the number of days from January to the current month is multiplied by 86400 seconds. To know how many days are in a month, a global variable is defined:

EXAMPLE 2:

```
unsigned char calendar [] = {31, 28, 31, 30
                              31, 30, 31, 31
                             30, 31, 30, 31}
```

The number of days from 1970 until the current year is multiplied by 86400 seconds. This function makes the necessary corrections for leap years.

The number of seconds from the current day is added to the previous values.

EXAMPLE 3: SOURCE CODE

```
unsigned long unixtime(DateTime crtime, char *local,unsigned char DST)
{
unsigned long s=0                                                                      // stores how many seconds passed from 1.1.1970, 00:00:00
unsigned char localposition=0,foundlocal=0                                             // checks if the local area is defined in the map
static unsigned char k=0;


if ((!(crtime.year%4)) && (crtime.month>2)) s+=86400                                   // if the current year is a leap one -> add one day (86400 sec)
crtime.month--                                                                      ; // dec the current month (find how many months have passed from the current year)

while (crtime.month)                                                                   // sum the days from January to the current month
{
          crtime.month--                                                               // dec the month
          s+=(calendar[crtime.month])*86400                                         ;  // add the number of days from a month * 86400 sec
}
                                                                                       // Next, add to s variable: (the number of days from each year (even leap years)) *
                                                                                             86400 sec,
                                                                                       // the number of days from the current month
                                                                                       // the each hour & minute & second from the current day
s +=((((crtime.year-YEAR0)*365)+((crtime.year-YEAR0)/4))*(unsigned long)86400)+(crtime.date-1)*(unsigned long)86400 +
     (crtime.hr*(unsigned long)3600)+(crtime.min*(unsigned long)60)+(unsigned long)crtime.sec;


while(timezone[localposition])                                                         // search the first locations in the database
{
if (timezone[localposition]==local) {foundlocal=1; break                            ;  // if the locations was found -> break the searching loop
localposition++                                                                     ;  // incr the counter (stores the position of the local city in the array)
}


if (foundlocal)                                                                        // if the local area is found inside the timezone[] array
          {                                                                            // calculate the time difference between localtime and UTC
          if (DST) s-=((timevalue[0][localposition]+timevalue[1][localposition])*3600);// if DST is active (Summer Time) -> subtract the standard time difference + 1 hour
          else s-=(timevalue[0][localposition]*3600)                                ; // else subtract the standard time difference (in seconds: 1 hour=3600 sec)
          }
else s=0                                                                            ; // return 0 if the local area is not foundinside the timezone[] array

return s                                                                            ;  // return the UNIX TIME
}
```

FIRMWARE DESCRIPTION

Drivers

Drivers are divided into 4 classes:

- LCD drivers
- SPI drivers
- RTCC registers access drivers
- Drivers related to the setup menu: keyboard drivers

LCD Drivers

The application is specifically implemented on the PIC18 Explorer demo board. On this board, it was important to reduce the number of GPIO pins used to access the LCD. Accessing the LCD is performed on a SPI bus (included in the MSSP1 module) through an auxiliary chip, the MCP23S17 SPI expander.

The related drivers are:

- Write command to LCD: `wrcmnd_lcd (unsigned char cmnd_lcd)`
- Write data byte/character to LCD: `wrdata_lcd (unsigned char data_lcd)`
- Write to LCD a string stored in the Flash: `wrstr_lcd (const rom unsigned char *str_lcd)`
- Write a long number to LCD: `wrnr_lcd (unsigned long nr_lcd)`

They are defined in the “`lcd_drivers.h`” file.

Drivers to Access RTCC Register

Since the MCP795W20 is an SPI RTCC, it will use the SPI bus of the MCU (the MSSP1 module). Accordingly, the related drivers will be divided into two categories: basic SPI drivers and RTCC drivers. As a control method, they use the SPP1IF bit (flag) in the PIR1 register (interrupt flag of the MSSP1 module). They read through polling and not through interrupts.

FIGURE 4: FLOWCHART FOR A TYPICAL WRITE OPERATION (FOR A RANDOM BYTE ACCESS)

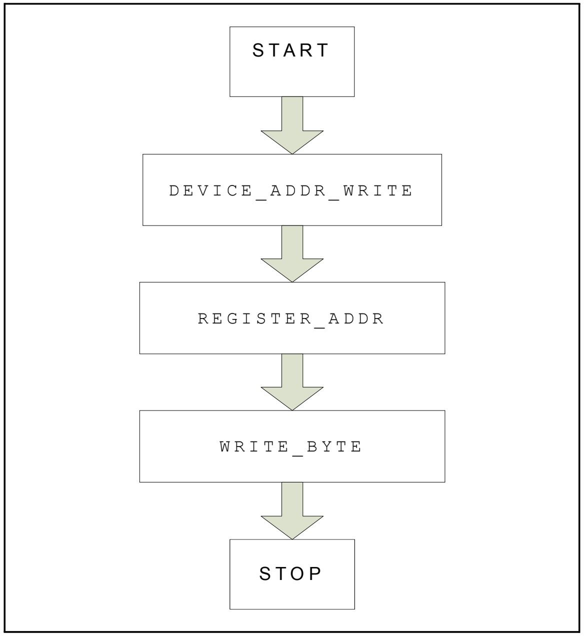
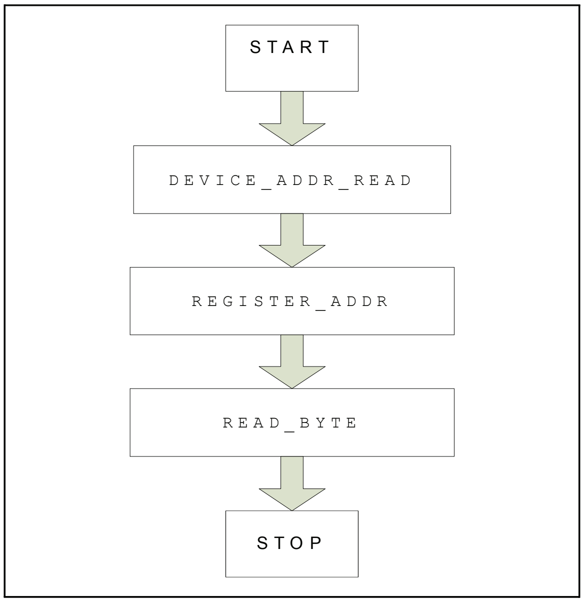


FIGURE 5: FLOWCHART FOR A TYPICAL READ OPERATION



The two related functions are: `void spi_rtcc_wr (unsigned char rtcc_reg, unsigned char time_var); unsigned char spi_rtcc_rd (unsigned char rtcc_reg)`. They are defined in the “`spi_rtcc_drivers.h`” file.

Keyboard Drivers (2 keys O.S.)

The keyboard is read into the keypress() function. The firmware is waiting the selection of one of the two on-board push buttons: S1 (KEY_INC) or S2 (KEY_MENU).

S1 (KEY_INC) can change the displaying mode or can set a time variable (S2 (KEY MENU) dependency). S2 (KEY_MENU) gives the rights to set the date and time.

Each push button has a flag which indicates if it was released or it is still pressed. The firmware reacts when the push button is pressed, not when it is released. This method of reading the keyboard eliminates the unpleasant effect of multiple counting when a key is pressed a long time without releasing it (this firmware reacts as if the button was pressed only once).

ACCESSING THE RTCC REGISTERS

There are two basic functions for accessing the RTCC register: one for writes and one for reads. They can be defined as: `void spi_rtcc_wr (unsigned char rtcc_reg, unsigned char time_var)`, `unsigned char spi_rtcc_rd (unsigned char rtcc_reg)`.

EXAMPLE 4: WRITES TO THE RTCC

```
spi_rtcc_start()        ; //    start SPI communication with the SPI RTCC (CS goes down)
spi_wrbyte(SPI_RTCC_WRITE);//   send the SPI WRITE command (0x12)
spi_wrbyte(rtcc_reg)    ; //    send the register's address
spi_wrbyte(time_var)    ; //    send SPI data
spi_rtcc_stop()         ; //    stop SPI communication
```

EXAMPLE 5: READS FROM THE RTCC

```
spi_rtcc_start()        ; //    start SPI communication with the SPI RTCC (CS goes down)
spi_wrbyte(SPI_RTCC_READ);//    send the SPI READ command (0x13)
spi_wrbyte(rtcc_reg)    ; //    send the register's address
rtcc_buf = spi_rdbyte()();//    read the result and store it
spi_rtcc_stop()         ; //    stop the SPI command with the SPI RTCC
return rtcc_buff        ; //    return the read result
```

As described in the data sheet, the addresses of the RTCC register are shown in Table 1.

TABLE 1: RTCC REGISTER ADDRESSES

Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
Time and Configuration Registers										
00h	Tenth Seconds				Hundredths of Seconds				Hundredths of Seconds	00-99
01h	ST (CT)	10 Seconds			Seconds				Seconds	00-59
02h		10 Minutes			Minutes				Minutes	00-59
03h	CASLGN	12/$\overline{24}$	10 Hour AM/PM	10 Hour	Hour				Hours	1-2 + AM/PM 00-23
04h			OSCON	VBAT	VBATEN		Day		Day	1-7
05h			10 Date		Date				Date	01-31
06h			LP	10 Month	Month				Month	01-12
07h	10 Year				Year				Year	00-99
08h	OUT	SQWE	ALM1	ALM0	EXTOSC	RS2	RS1	RS0	Control Reg.	
09h	CALIBRATION								Calibration	
0Ah	WDTEN	WDTIF	WDDEL	WDTPLS	WD3	WD2	WD1	WD0	Watchdog	
0Bh	EBHIF	EVLIF	EVEN1	EVEN0	EVWDT	EVDLB	EVHS1	EVHS0	Event Detect	

According to these addresses, in the basic read/write functions, only the register's address will differ. Read is used to see if the correct EVDT register's value was written. Writes are used in the initialization function and in the setup sequence (the main function).

CONCLUSION

This application note is a UNIX time-stamp converter. The project is performed on a PIC18 Explorer demo board, using the on-board resources: LCD (accessed through the SPI bus) and push buttons. The code (drivers and main function) is written in C, using the C18 compiler. The target microcontroller is the PIC18F87J11.

APPENDIX A: REVISION HISTORY

Revision A (10/2011)

Original Release.

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.

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.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2009

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC^{32} logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

All other trademarks mentioned herein are property of their respective companies.

© 2011, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

ISBN: 978-1-61341-714-0

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, KEELOQ® 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.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

08/02/11