TB3128

CRC and Memory Scan on 8-Bit Microcontrollers Technical Brief

Author: *Vinaya Skanda*
Microchip Technology Inc.

INTRODUCTION

In most applications today, data integrity is necessary for both storage and transmission. A popular error-detection technique, the Cyclic Redundancy Check (CRC) can be used to preserve the integrity of data in memory and communications.

This technical brief provides information about the Cyclic Redundancy Check (CRC) peripheral along with the Memory Scan functionality available on a range of PIC® microcontrollers. The CRC calculation and Memory Scan methods along with different standards and commonly used CRC polynomials are also explained in this technical brief. The CRC and Memory Scan peripheral are useful while performing the memory tests required in the IEC 60730 standard to support the Class B certification.

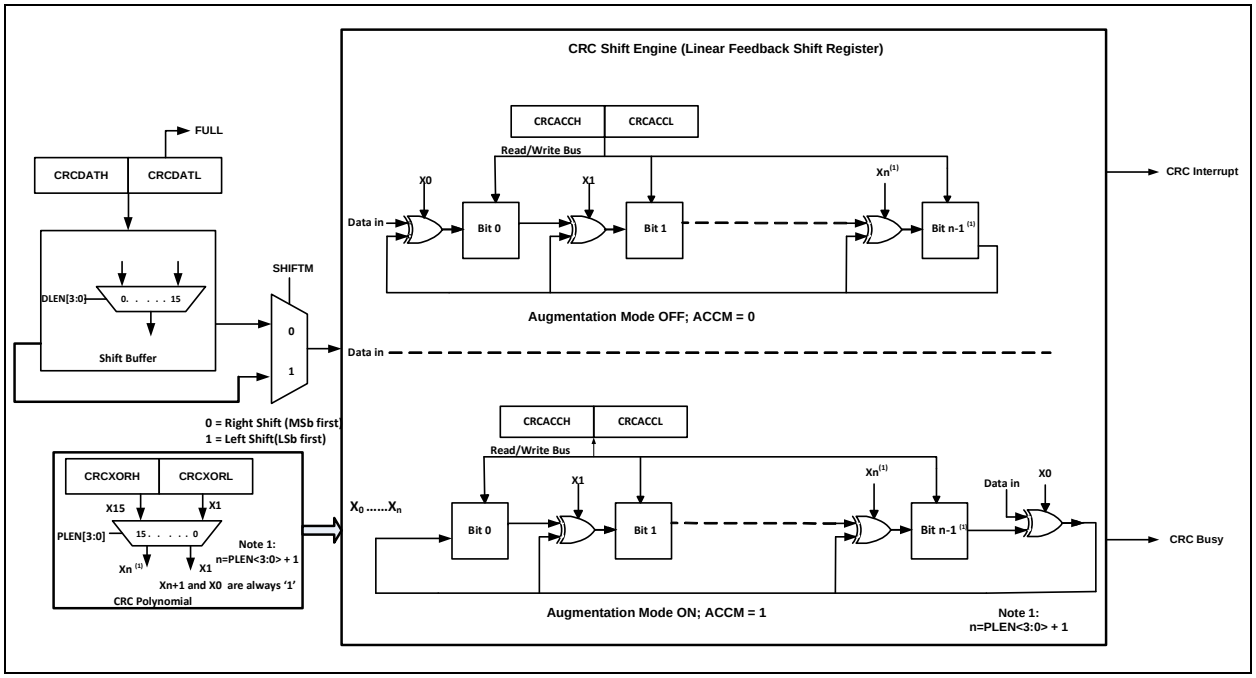
CRC PERIPHERAL ON PIC MICROCONTROLLER

The Cyclic Redundancy Check (CRC) module provides a software-configurable hardware-implemented CRC checksum generator. The CRC algorithm treats the data to be a binary number. This number is divided by another binary number called the polynomial. The remainder of this division is the CRC checksum, which is appended to the data. Every integer can be expressed uniquely as a polynomial in base 2 with coefficients that are either '0' or '1'. It is convenient to interpret the polynomial in terms of a dummy variable X.

For example, a message 110101 is represented by the polynomial: $X^5 + X^4 + X^2 + 1$ $(1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0)$

In communication systems, when messages are transmitted, the CRC checksum is appended to the message. The receiver then divides the message (including the calculated CRC) by the same polynomial that the transmitter used (in order to implement a CRC based on a polynomial, the transmitter and receiver must use the same polynomial). If the result of this division yields a zero, then the transmission is considered to be successful. However, if the result of this division is not equal to zero, then an error is considered to have occurred during the transmission and corrective action needs to be taken against the presumed data corruption. A CRC is called an n-bit CRC when its checksum value is n-bits long.

FIGURE 1: BLOCK DIAGRAM OF CRC



The CRC module uses a Linear Feedback Shift Register (LFSR) implementation to perform the polynomial division required for the CRC calculation, as shown in Figure 1. The CRC module can accept up to a 17-bit (n+1) generator polynomial. In any (n+1) bit polynomial, X^n and X^0 are always '1'. While loading the (n+1) bit polynomial into the CRCXOR<15:1> register, only X^{n-1} to X^0 should be entered. The PLEN<3:0> bits are used to specify the length of the polynomial. While loading the length, the MSB and LSB of the (n+1) bit polynomial should be excluded. Hence, n-1 should be loaded.

For example, in CRC-16-ANSI, a 17-bit polynomial $X^{16}+X^{15}+ X^2 + 1$, n = 16.

So, PLEN<3:0> = n – 1 = 15

CRCXOR<15:1> = 0b100000000000010

(CRCXOR <0> is unimplemented since CRC hardware considers X^{16} and X^0 to be '1')

So, CRCXOR can be either 0x8005 or 0x8004. The CRC engine reads it as 0x8005, but if CPU reads it using software, it is read as 0x8004.

The data needs to be loaded into the CRCDATH and CRCDATL registers in the same sequence. Once the data is loaded into the CRCDATL register, the entire 16-bit data is moved into the Shift buffer, this is a readable register. The DLEN bits determine the data width (the valid bits in the CRCDATH and CRCDATL registers). Only the data as determined by the DLEN bits are moved to the shift buffer. If the SHIFTM bit is set to '1', the data will be shifted with LSb first, else data will be shifted with MSb first.

Augmentation of Zeros

For the CRC calculation to be completed, all the data bits need to be shifted through all the XOR gates, and this requires adding zeros equal to the length of the polynomial at the end. This is known as augmentation of zeros. Previously, this had to be taken care of in software. Now, Augmentation mode is available to take care of this in hardware, thereby reducing the software overhead. By setting ACCM bit to '1', one can enable Augmentation mode.

Initial CRC Value (Seed Value)

This is the initial value that provides a fixed starting point from which the data bits can progress. Many CRC algorithms initialize their seed value to a non-zero value. As in real applications, many data strings are likely to begin with a long series of '0's and the computation of CRC ignores any number of '0's ahead of the first '1' bit.

To avoid this problem, before computing the n-bit CRC we will always begin by exclusive ORing the leading n bits with an n-bit string of non-zero value.

The seed value can be used in two methods for the CRC calculation, for example:

Direct Method

The seed value is loaded in to the CRCACC <15:0> registers.

Non-Direct Method

The seed value is considered to be the initial data value and prefixed with the actual data stream and the CRCACC<15:0> register is loaded with 0x0000.

The BUSY bit indicates if the CRC has completed calculations. Once the BUSY bit is '0', the check value can be read from the CRCACC<15:0> registers. The CRCIF flag will also get set to '1'.

MEMORY SCAN MODULE ON PIC MICROCONTROLLER

The Program Memory Scan module can be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory addresses. The CRC module is coupled with the Memory Scan module for faster CRC calculations. The Memory Scan module can automatically provide data to the CRC module. This memory test will be helpful for the IEC 60730 standard to support the Class B certification.

Four modes of memory scan operation are possible:

Burst: CPU operation is stalled until the whole memory scan is completed. It gives highest memory scan throughput.

Concurrent: Scan starts immediately after setting of SCANGO bit. CPU operation is stalled during the memory access.

CPU resumes execution following each access.

Triggered: Waits to begin the scan until the rising edge from separate trigger clock. CPU operation is stalled during the memory access.

CPU resumes execution following each access.

Peek: Waits for an instruction cycle in which the CPU does not need to access the NVM (such as a branch instruction) and uses that cycle to do its own NVM access. It gives the lowest throughput. CPU operation is unaffected.

The Memory Scanner can automatically provide data to the CRC module when data is placed in the program memory. When Memory Scan is used in Peek mode, it has the advantage of freeing the CPU for other tasks, while the data is being checked for errors using CRC.

Interrupt interaction during memory scan: If the INTM bit is set, the interrupt will take precedence there by delaying the memory scan. If INTM is not set, the scanner will take precedence over the interrupt, resulting in increased interrupt response latency.

Configuration of the CRC and Memory Scan Peripheral

Some commonly used CRC generator polynomials are as follows:

TABLE 1: COMMONLY USED CRC GENERATOR POLYNOMIALS

Name	Polynomial	Polynomial Presentation	Seed Value
SDLC (CRC-16-CCITT)	$X^{16} + X^{12} + X^5 + 1$	0x1021	0xFFFF
CRC-16-ANSI	$X^{16} + X^{15} + X^2 + 1$	0x8005	0
CRC12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$	0x80F	0

Configuring the CRC

This section provides the configuration sequence for the CRC in the form of a code snippet.

EXAMPLE 1: CODE SNIPPET FOR CRC-CCITT STANDARD FOR DIRECT MODE CRC

```
unsigned char data_buffer[] = {0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39};
CRCCON0bits.EN = 1; // enable the CRC module
CRCACC = 0xFFFF; //seed a starting CRC value
CRCXOR = 0x1021; //CRC generator polynomial for CRC-CCITT
CRCCON1bits.DLEN = 0b0111; //length of the data word = 8 bits
CRCCON1bits.PLEN = 0b1111; //order of the polynomial = 16
CRCCON0bits.ACCM = 1; //Direct CRC algorithm, zero augmentation not required
CRCCON0bits.SHIFTM = 0; // MSb of the data will be shifted first
CRCCON0bits.CRCGO = 1; //start the CRC module
data_size = sizeof(data_buffer); // data_buffer is the array containing data
for (data_index=0; data_index < data_size; data_index++  )
{
    while (CRCCON0bits.CRCFULL); // wait until the data registers are full
    if (CRCCON1bits.DLEN <8){ // data has < 8 bits; load data in CRCDATL
        CRCDATL = data_buffer[data_index];}
    else {// data has> 8 bits; load CRCDATAH register first and then CRCDATL
        CRCDATH = (data_buffer[data_index] & 0xFF00)>>8;
        CRCDATL = (data_buffer[data_index] & 0xFF);    }
}
while(CRCCON0bits.BUSY);// check if CRC calculation is completed
crc = CRCACC; // read the CRC check value ( = 0x29B1 for given data sequence)
```

Configuring the Memory Scan

According to the IEC 60730 standard to support the Class B certification, the memory tests are performed periodically during system run time. At the system start-up, the CRC checksum of the data stored in the program memory (Flash) is calculated and can be used as a reference checksum. It can be stored in the Flash memory. During run time, the CRC and Memory Scan functions can be called periodically. The calculated checksum during run time can be compared with the reference checksum. If both the checksums are equal, then the memory test is considered to have passed.

EXAMPLE 2: CODE SNIPPET FOR MEMORY SCAN USING CRC-CCITT STANDARD FOR NON-DIRECT MODE CRC

```
SCANCON0bits.EN = 1; // enable the Memory Scan module
SCANCON0bits.MODE = 0; // Concurrent mode
SCANCON0bits.INTM = 0; //Interrupts do not prevent NVM access
SCANLADR = STARTADR; // load beginning location of the memory to be scanned
SCANHADR = STOPADR; //load the end location of the memory to be scanned

CRCCON0bits.EN = 1; // enable the CRC module
CRCACC = 0xFFFF; //seed a starting CRC value
CRCXOR = 0x1021; //CRC generator polynomial for CRC-CCITT
CRCCON1bits.DLEN = 0b0111; //length of the data word = 8 bits
CRCCON1bits.PLEN = 0b1111; //order of the polynomial = 16
CRCCON0bits.ACCM = 1; //zeroes equal to order of polynomial will be augmented
CRCCON0bits.SHIFTM = 0; // MSb of the data will be shifted first
CRCCON0bits.CRCGO = 1; //start the CRC module
SCANCON0bits.SCANGO = 1; //Begin the scan
while(SCANCON0bits.SCANGO); // wait for scanner to finish pushing data in to
//the CRC data registers
while(CRCCON0bits.BUSY); //check if CRC calculation is completed
crc = CRCACC;// read the CRC check value
```

Using MPLAB® Code Configurator (MCC) for CRC and Memory Scan Modules

The MPLAB® Code Configurator, which is a plug-in tool for MPLAB X IDE, generates the drivers for controlling and driving the CRC and Memory Scan modules based on the settings and selections made in the GUI, as shown in Figure 2.

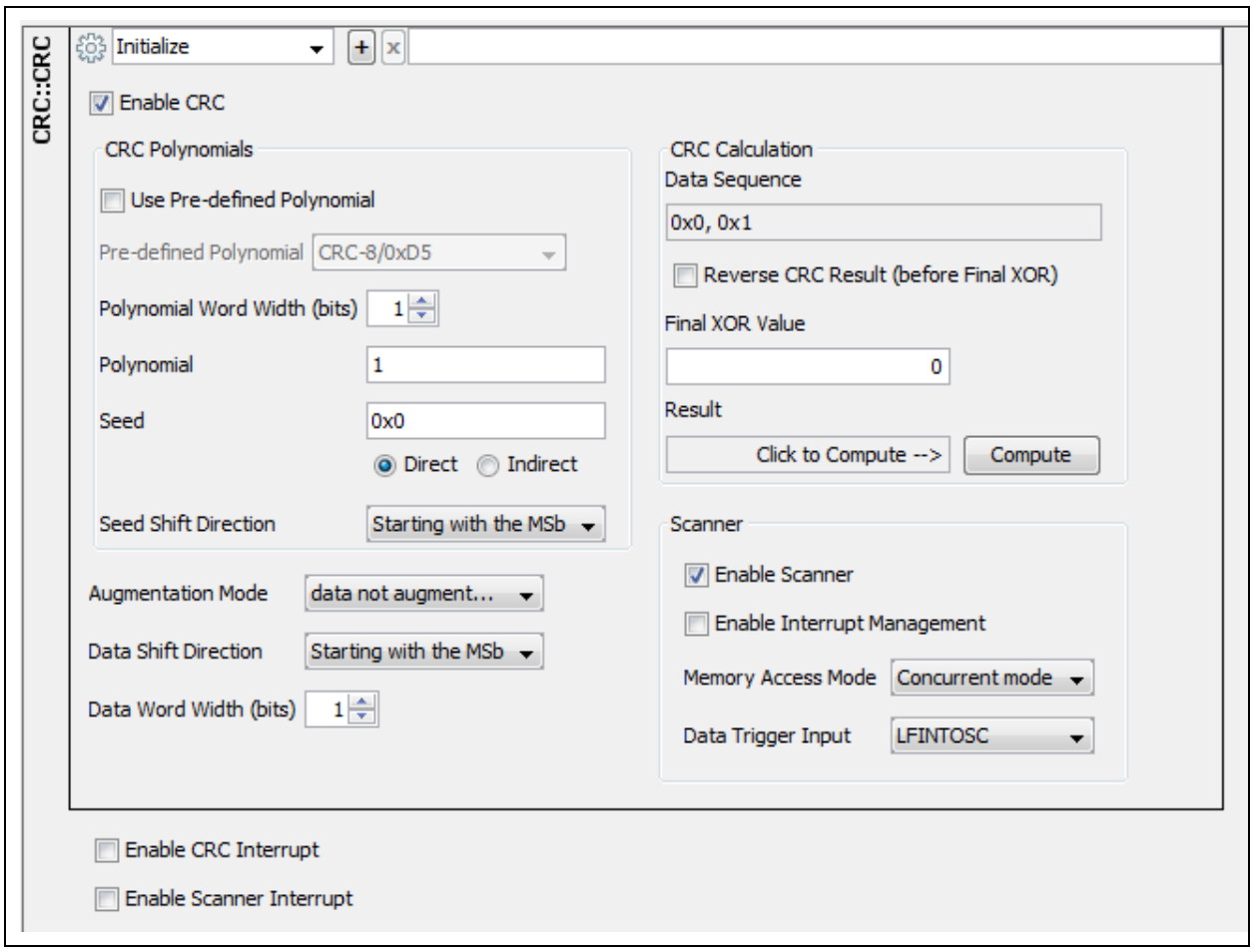
After clicking the Generate Code button, the project in MPLAB X generates all the APIs for CRC and memory scan as listed in the below Table 2:

TABLE 2: LIST OF MCC APIs FOR CRC AND MEMORY SCAN

Function	Description
void CRC_Initialize(void)	Initializes the CRC as per settings done in the MCC.
void CRC_Start(void)	Sets the CRCGO bit of the CRCCON0 register to begin the shifting process.
bool CRC_8BitDataWrite(uint8_t data)	Writes data into CRCDATHL register pair.
uint16_t CRC_CalculatedResultGet(bool reverse, uint16_t xorValue)	Reads and returns the normal or reverse value.
bool CRC_IsBusy(void)	Returns the status of the BUSY bit of the CRCCON0 register to check CRC calculation is over or not.
void CRC_SCAN_StartScanner(void)	Starts the scanning process.
void CRC_SCAN_StopScanner(void)	Stops the scanning process.
void CRC_SCAN_SetAddressLimit(uint16_t startAddr,uint16_t endAddr)	Loads the address limits into the SCANLADRH/L and SCANHADRH/L register pairs.
bool CRC_SCAN_HasInvalidAddressOccured(void)	Checks the occurrence of invalid address in scanning process.
bool CRC_SCAN_IsScannerBusy(void)	Returns the status of BUSY bit of SCANCON0 register.
bool CRC_SCAN_HasScanCompleted(void)	Returns the status of the SCANIF interrupt flag.

Note 1: For the latest version of the MCC GUI for CRC, visit Microchip's web site.

FIGURE 2: MPLAB® CODE CONFIGURATOR (MCC) FOR CRC AND MEMORY SCAN MODULES



Conclusion

The CRC can be used to detect the errors in the data transmission in various digital communication systems. Most of the popular communication protocols like USB, Modbus, CAN and Ethernet employ CRC for error detection. The CRC along with the Memory Scan peripheral is useful while performing the memory tests required in the IEC 60730 standard to support the Class B certification. This technical brief covers the CRC peripheral on 8-bit PIC microcontrollers. It also provides the usage of the Memory Scan module in conjunction with CRC and its configuration.

APPENDIX A: EXAMPLE REGISTER SUMMARY FOR CRC AND MEMORY SCAN

TABLE A-1: SUMMARY OF REGISTERS ASSOCIATED WITH CRC AND MEMORY SCAN MODULES

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CRCACCH	ACC<15:8>							
CRCACCL	ACC<7:0>							
CRCCON0	$\overline{EN}$	CARGO	BUSY	ACCM	—	—	SHIFTM	FULL
CRCCON1	DLEN<3:0>				PLEN<3:0>			
CRCDATH	DATA<15:8>							
CRCDATL	DATA<7:0>							
CRCSHIFTH	SHIFT<15:8>							
CRCSHIFTL	SHIFT<7:0>							
CRCXORH	X<15:8>							
CRCXORL	X<7:1>							—
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF
PIR4	SCANIF	CRCIF	SMT2PWAIF	SMT2PRAIF	SMT2IF	SMT1PWAIF	SMT1PRAIF	SMT1IF
PIE4	SCANIE	CRCIE	SMT2PWAIE	SMT2PRAIE	SMT2IE	SMT1PWAIE	SMT1PRAIE	SMT1IE
SCANCON0	EN	SCANGO	BUSY	INVALID	INTM	—	MODE<1:0>	
SCANHADRH	HADR<15:8>							
SCANHADRL	HADR<7:0>							
SCANLADRH	LADR<15:8>							
SCANLADRL	LADR<7:0>							
SCANTRIG	—	—	—	—	—	—	TSEL<1:0>	

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by the Timer0 module.

* Page provides register information.

APPENDIX B: CONVERSION OF SEED IN NON-DIRECT MODE OF CRC

If Non-Direct mode for CRC seed is used, the user can pre-convert the seed value and directly load it into the CRCACC<15:0> registers to reduce the time taken for calculations.

For example, in SDLC (CCITT) standard, the initial seed value used in direct algorithm is 0xFFFF. This can be pre-converted to a 0x84CF using the code snippet:

Code snippet for converting from direct algorithm seed values to non-direct algorithm seed values, as shown in Example B-1.

EXAMPLE B-1: CONVERTING DIRECT ALGORITHM SEED VALUES TO NON-DIRECT ALGORITHM SEED VALUES

```
crcinit_direct = crcinit;
crc = crcinit;
for (i=0; i<order; i++) //order of CRC polynomial e.g. for CCITT order = 16
{
    bit = crc & 1;
    if (bit) crc^= polynom;
    crc >>= 1;
    if (bit) crc|= crchighbit;//crchighbit for 16 bit CRC polynomial= 0x8000
}
crcinit_nondirect = crc;
```

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.

Trademarks

ISBN: 978-1-63277-052-3

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, 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

Austin, TX
Tel: 512-257-3370

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
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983
Indianapolis

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

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

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
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: 852-2943-5100
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 - Dongguan
Tel: 86-769-8702-9880

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
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: 86-755-8864-2200
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

ASIA/PACIFIC

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

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

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: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

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: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
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 - Dusseldorf
Tel: 49-2129-3766400

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

Germany - Pforzheim
Tel: 49-7231-424750

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

Italy - Venice
Tel: 39-049-7625286

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

Poland - Warsaw
Tel: 48-22-3325737

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

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

01/27/15