How to Use the DMA CRC Generator on PIC32MX/ PIC32MZ/PIC32MM Devices

Introduction

The Cyclic Redundancy Check (CRC) is a robust error-checking algorithm to ensure the integrity of data before it is processed. The CRC value (checksum) is associated with a message or a particular block of data. Whether it is a data packet for communication or a block of data stored in memory, CRC helps to validate it before processing.

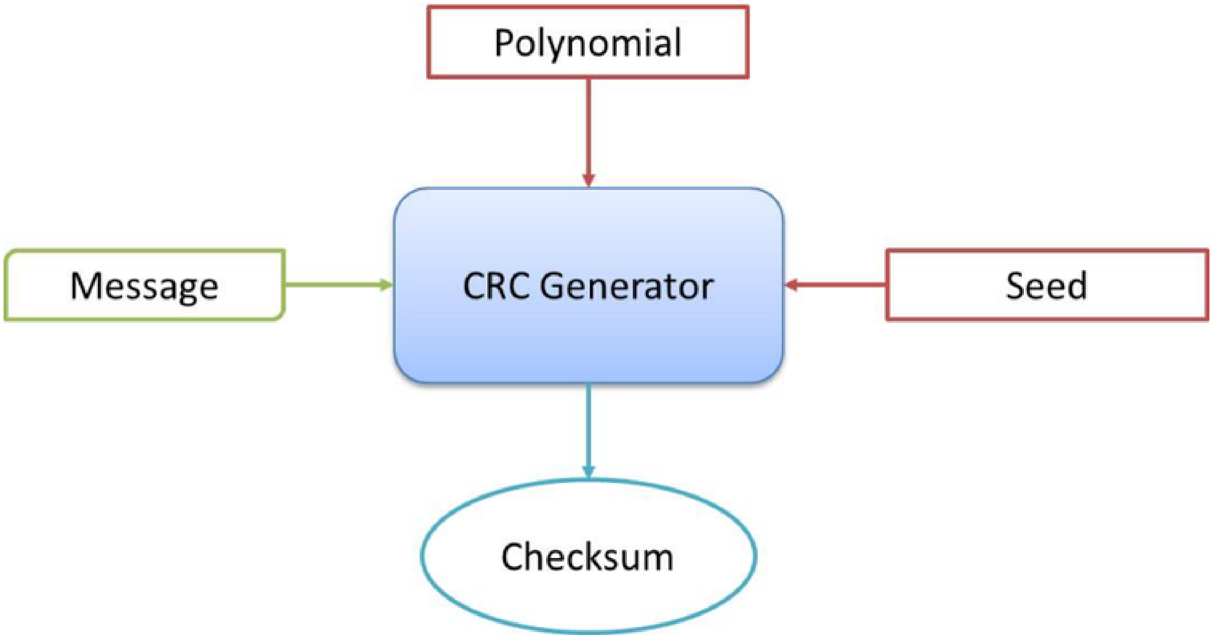
The CRC calculation is an iterative process and it consumes considerable CPU bandwidth when implemented in software. The CRC block in the Special Function Module (SFM) integrated in the DMA module on PIC32MX/PIC32MZ/PIC32MM devices facilitates a fast and efficient CRC checksum calculation with minimal software overhead.

Table of Contents

1. Concept

The block diagram of a CRC generator engine is shown in the following figure.

Figure 1-1. CRC Generator



Polynomial

The CRC calculation is an iterative process of performing the XOR operation. The CRC algorithm uses a *Polynomial* to perform its calculations. The divisor, dividend and remainder that are represented by numbers are termed as *polynomials with binary coefficients*.

For example, the number, 19h (11001b), is represented in the following equation:

(1*x4) + (1*x3) + (0*x2) + (0*x1) + (1*x0) or **x4 + x3 + x0**

The application uses a suitable generator polynomial to detect errors using CRC. Some widely used polynomials are given in the following table.

Table 1-1. Polynomial Functions for Common CRC

Name	Hex Value	Polynomial
CRC-16	0x8005	$x^{16} + x^{15} + x^{2} + 1$
CRC-CCITT	0x1021	$x^{16} + x^{12} + x^{5} + 1$
CRC-XMODEM	0x8408	$x^{16} + x^{15} + x^{10} + x^{3}$
CRC-32	0x04C11DB7	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

Seed

The seed is the desired initial value for the CRC used by the algorithm to start the computation. The seed can be a direct or non-direct value. The direct seed algorithm does not require the *n*-zero bits to be appended at the end of the message for computing the *n*-bit CRC. The non-direct seed algorithm requires the *n* zero bits to be appended at the end of the message for computing the *n*-bit CRC. 16-zero bits need to be appended to the message for a 16-it CRC computation, and 32-zero bits need to be appended to the message for a 32-bit CRC computation.

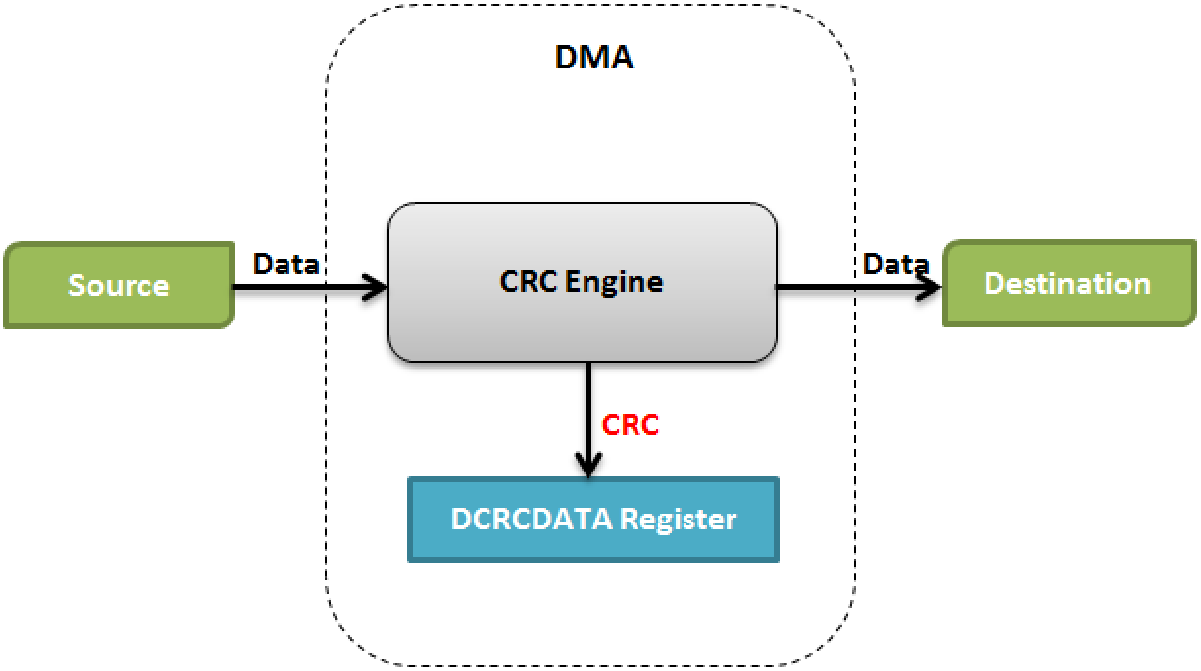
Non-direct seeds can be obtained by the XOR direct seed. The computed CRC is same for the direct or non-direct seeded algorithms. The choice of initial seed being a direct or non-direct value is decided based on the ability of the CRC algorithm to avoid the augmented zeros at the end of the message.

Note: The DMA module on PIC32MX/PIC32MZ/PIC32MM devices has a SFM. The SFM has a Linear Feedback Shift Register (LFSR) CRC engine. The LFSR CRC engine attached to DMA allows parallel CRC computation as data transfers from source to destination over DMA channels. The DMA CRC engine is compatible to direct and non-direct seeded CRC computations. The DMA CRC engine has two modes of operation, Background mode and Append mode.

Background Mode

The CRC engine attached to the DMA on PIC32MX/PIC32MZ/PIC32MM devices has a Background mode. In this mode, the DMA channel associated for CRC computation maintains its default behavior. The DMA reads the data from the source, passes it through the CRC module, and writes it to the destination as shown in the following figure.

Figure 1-2. Background Mode CRC

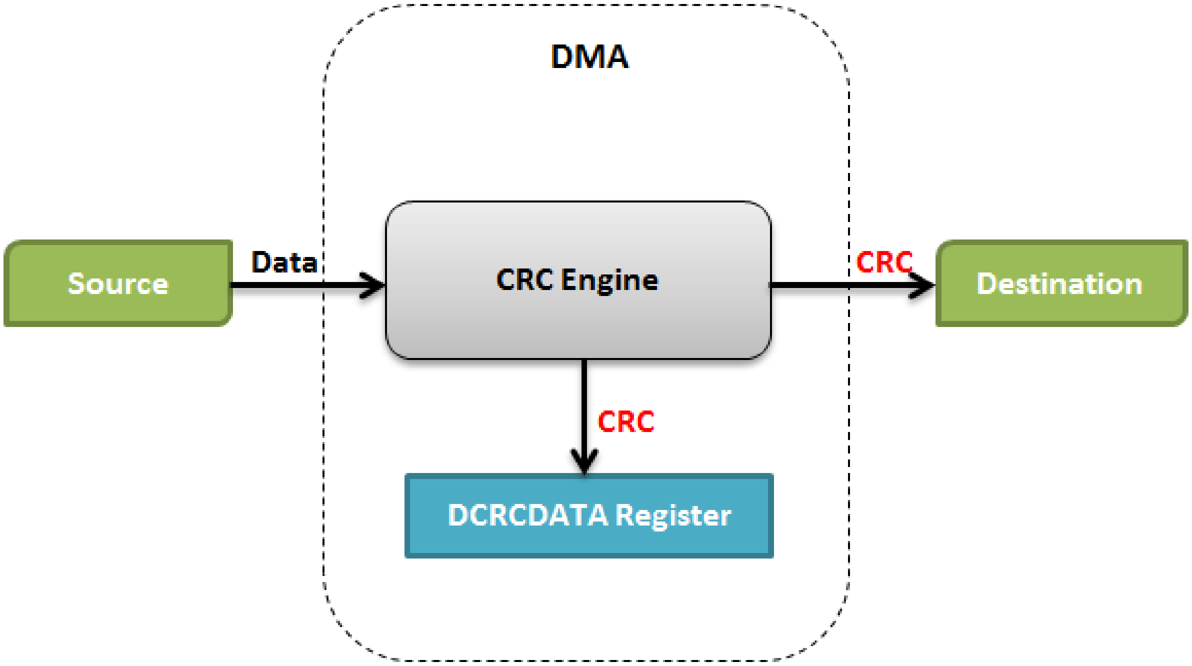


When the data transmission is completed, the calculated CRC is available in the DCRCDATA register.

Append Mode

The CRC engine attached to the DMA on PIC32MX/PIC32MZ/PIC32MM devices has an Append mode. In this mode, the DMA channel associated for CRC computation only feeds the data from the source to the CRC engine. It does not write the data to the destination, instead when a data transmission completes, it writes the computed CRC to the destination and the DCRCDATA register as shown in the following figure.

Figure 1-3. Append Mode CRC



2. Solution

The DMA system service under Microchip's MPLAB Harmony software framework provides the support to implement CRC computation. The following is the sequence of steps used in an MPLAB Harmony-based project to compute the CRC of data using the CRC engine tied to the DMA peripheral.

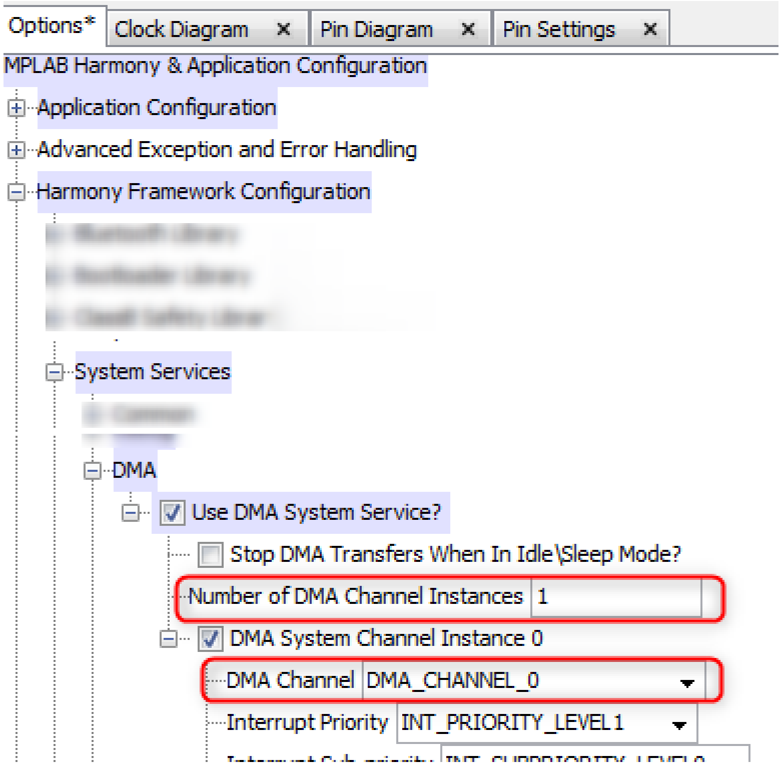
1. Configure a DMA channel using MHC.
2. Open the DMA channel.
3. Register DMA channel event handler.
4. Setup DMA operation characteristics.
5. Setup the CRC computation.
6. Setup and initiate data transfer.
7. Observe the CRC result.

Configure DMA Channel using MHC:

The DMA channel used for CRC computation is configured using the MHC (MPLAB Harmony Configurator) tool as shown in the following figure.

Expand the *MPLAB Harmony Configurator* > *Options* > *System Services* > *DMA* selection tree.

Figure 2-1. DMA Channel Configuration

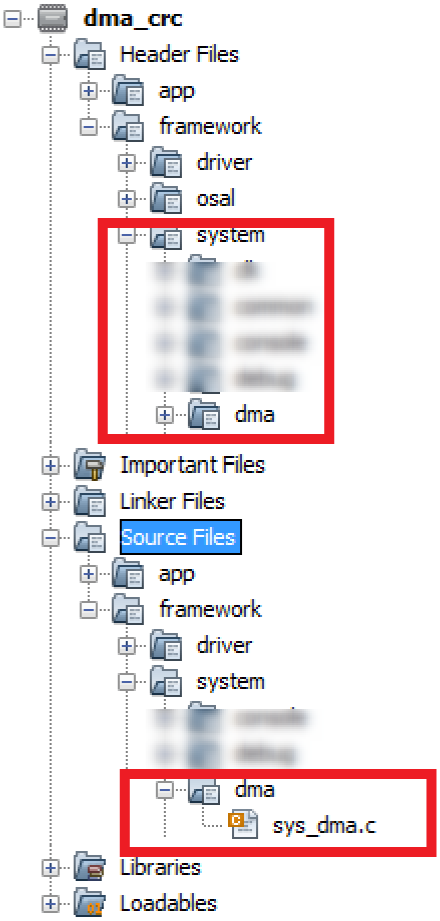


MPLAB Harmony & Application Configuration
- Application Configuration
- Advanced Exception and Error Handling
- Harmony Framework Configuration
  - System Services
    - DMA
      - ☑ Use DMA System Service?
        - □ Stop DMA Transfers When In Idle\Sleep Mode?
        - Number of DMA Channel Instances: 1
        - ☑ DMA System Channel Instance 0
          - DMA Channel: DMA_CHANNEL_0
          - Interrupt Priority: INT_PRIORITY_LEVEL1
          - Interrupt Sub-priority: INT_SUBPRIORITY_LEVEL0

Note:

1. The configured DMA channel could be associated with a Harmony driver (USART, SPI, and so on) if the data transfer is to/from the peripheral associated with the Harmony driver.
2. The configuration shown previously adds the DMA system service library to your Harmony project. The folder structure shown in the following figure is added to the project when you generate the code with the above DMA channel configuration.

Figure 2-2. DMA System Service Folders

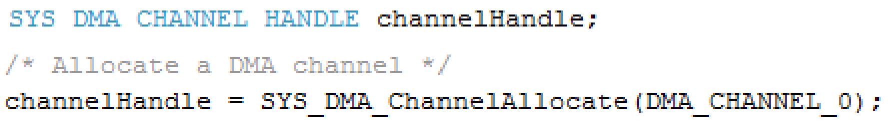


3. The generated code initializes the configured DMA channel by calling the function *SYS_DMA_Initialize* in file `system_init.c`.

Open the DMA Channel:

In the application code, get a handle to the configured DMA channel by calling the DMA system service function *SYS_DMA_ChannelAllocate*, as shown in the following figure.

Figure 2-3.



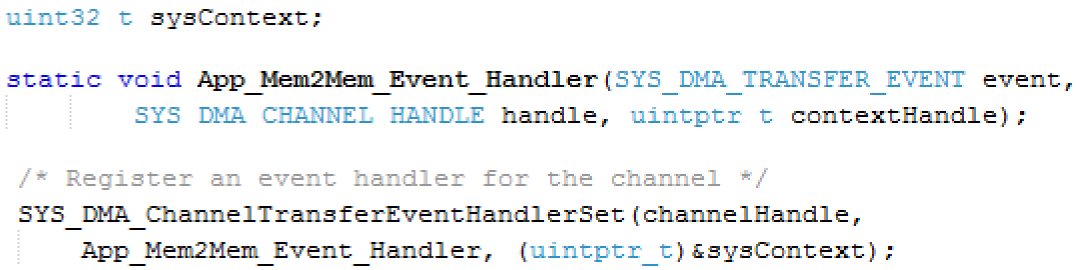
```
SYS_DMA_CHANNEL_HANDLE channelHandle;

/* Allocate a DMA channel */
channelHandle = SYS_DMA_ChannelAllocate(DMA_CHANNEL_0);
```

Note: If the configured DMA channel is associated with a Harmony driver (USART, SPI, and so on), the application need not call the function *SYS_DMA_ChannelAllocate* explicitly as the channel will have been allocated by the driver.

Register DMA channel Event Handler:

In the application code, register an even handler for the opened DMA channel by calling the DMA system service function *SYS_DMA_ChannelTransferEventhandlerSet* as shown in the following figure.

Figure 2-4.



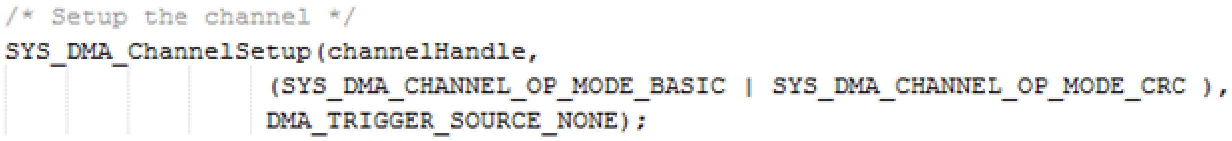
```
uint32_t sysContext;

static void App_Mem2Mem_Event_Handler(SYS_DMA_TRANSFER_EVENT event,
            SYS_DMA_CHANNEL_HANDLE handle, uintptr_t contextHandle);

 /* Register an event handler for the channel */
 SYS_DMA_ChannelTransferEventHandlerSet(channelHandle,
     App_Mem2Mem_Event_Handler, (uintptr_t)&sysContext);
```

Note: If the configured DMA channel is associated with a Harmony driver (USART, SPI, and so on) the application need not call the function *SYS_DMA_ChannelTransferEventhandlerSet* explicitly as the event handler for the channel is registered by the driver.

Setup DMA operation characteristics:

In the application code, setup (operation mode and channel trigger source) the registered DMA channel for CRC enabled operation by calling the DMA system service function *SYS_DMA_ChannelSetup* as shown in the following figure.

Figure 2-5.



```
/* Setup the channel */
SYS_DMA_ChannelSetup(channelHandle,
                    (SYS_DMA_CHANNEL_OP_MODE_BASIC | SYS_DMA_CHANNEL_OP_MODE_CRC ),
                    DMA_TRIGGER_SOURCE_NONE);
```

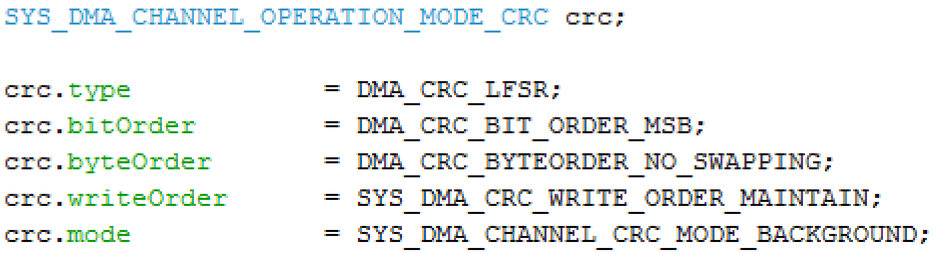
Note: If the configured DMA channel is associated with a Harmony driver (USART, SPI, and so on), enable the CRC operation by calling the DMA Harmony PLIB function *PLIB_DMA_CRCEnable* as shown in the following code.

```
PLIB_DMA_CRCENABLE ( DMA_ID_0) ;
```

Setup the CRC Computation:

The *crc* structure fields, *type*, *bitOrder*, *byteOrder* and *writeOrder* are set according to the desired values as shown in the following figure.

Figure 2-6.



```
SYS_DMA_CHANNEL_OPERATION_MODE_CRC crc;

crc.type          = DMA_CRC_LFSR;
crc.bitOrder      = DMA_CRC_BIT_ORDER_MSB;
crc.byteOrder     = DMA_CRC_BYTEORDER_NO_SWAPPING;
crc.writeOrder    = SYS_DMA_CRC_WRITE_ORDER_MAINTAIN;
crc.mode          = SYS_DMA_CHANNEL_CRC_MODE_BACKGROUND;
```

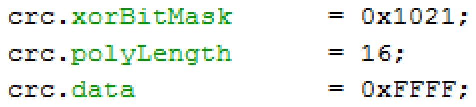
Note:

1. The field mode is set to *SYS_DMA_CHANNEL_CRC_MODE_BACKGROUND* when Background mode is desired.
2. The field mode is set to *SYS_DMA_CHANNEL_CRC_MODE_APPEND* when Append mode is desired.

The fields *xorBitMask* (polynomial value), *polyLength* (polynomial length in bits) and *data* (initial seed value) are inputs to the CRC engine.

To compute the CRC for a 16-bit CRC-CCIT polynomial (0x1021) with an initial seed of 0xFFFF, the structure of the *crc* fields are set as shown in the following figure.

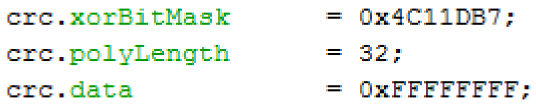
Figure 2-7.



```
crc.xorBitMask      = 0x1021;
crc.polyLength      = 16;
crc.data            = 0xFFFF;
```

To compute the CRC for a 32-bit CRC-32 polynomial (0x4C11DB7) with an initial seed of 0xFFFFFFFF, the structure of the *crc* fields are set as shown in the following figure.

Figure 2-8.



```
crc.xorBitMask      = 0x4C11DB7;
crc.polyLength      = 32;
crc.data            = 0xFFFFFFFF;
```

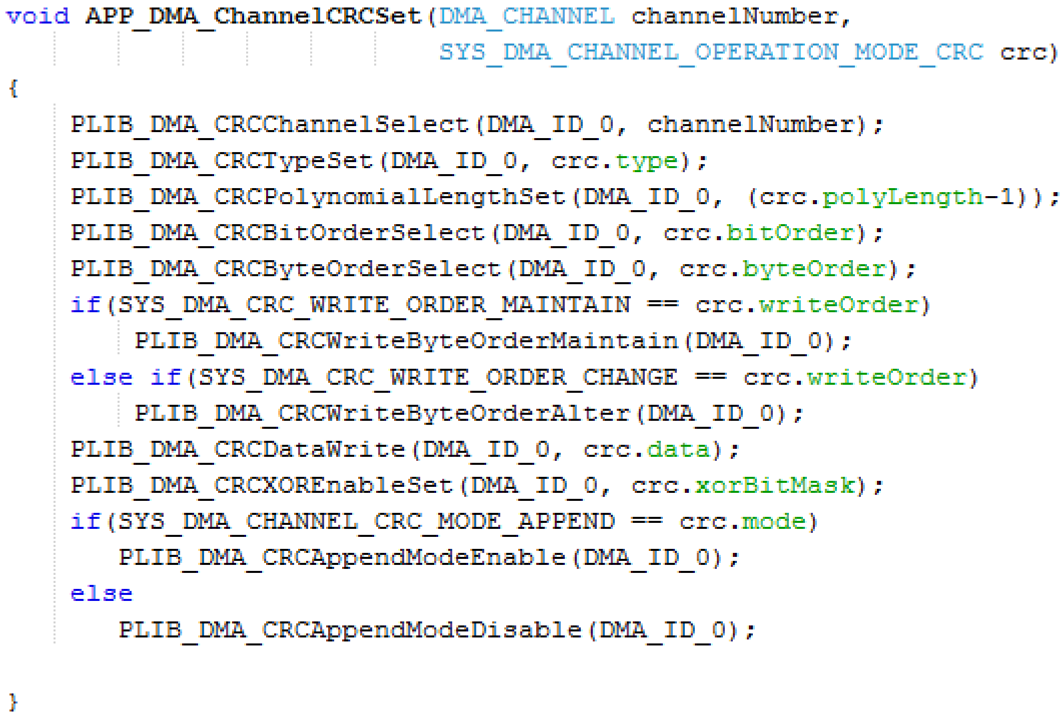
In the application code, setup the CRC computation for the DMA channel setup for the CRC enabled operation by calling the DMA system service function *SYS_DMA_ChannelCRCSet* as shown in the following code.

```
SYS_DMA_ChannelCRCSet (channelHandle, crc) ;
```

Note:

1. If the configured DMA channel is associated with a Harmony driver (USART, SPI, and so on), implement the following function (by using the DMA PLIB functions) to setup the CRC computation for the configured DMA channel.

   Figure 2-9.

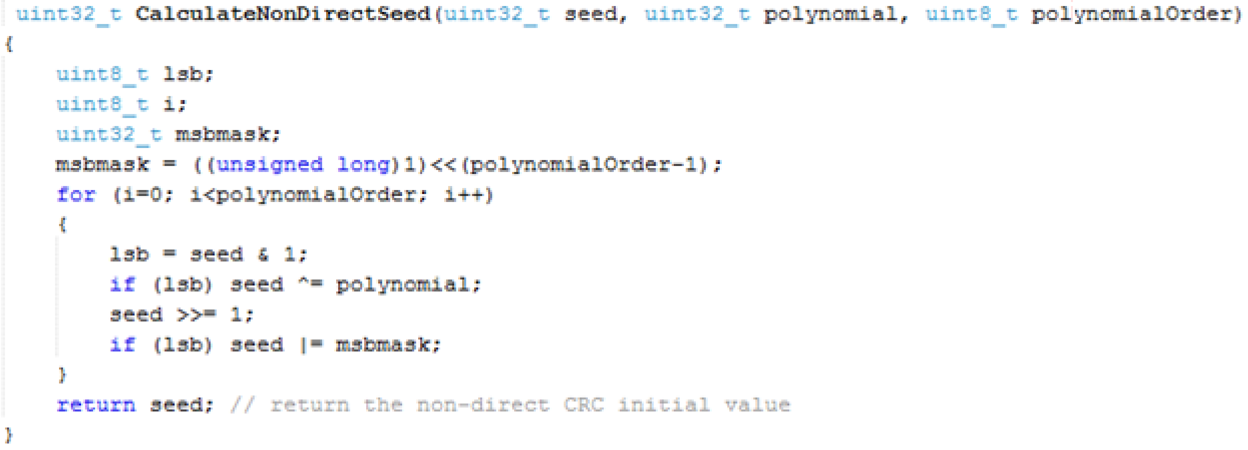
```
void APP_DMA_ChannelCRCSet(DMA_CHANNEL channelNumber,
                           SYS_DMA_CHANNEL_OPERATION_MODE_CRC crc)
{
    PLIB_DMA_CRCChannelSelect(DMA_ID_0, channelNumber);
    PLIB_DMA_CRCTypeSet(DMA_ID_0, crc.type);
    PLIB_DMA_CRCPolynomialLengthSet(DMA_ID_0, (crc.polyLength-1));
    PLIB_DMA_CRCBitOrderSelect(DMA_ID_0, crc.bitOrder);
    PLIB_DMA_CRCByteOrderSelect(DMA_ID_0, crc.byteOrder);
    if(SYS_DMA_CRC_WRITE_ORDER_MAINTAIN == crc.writeOrder)
        PLIB_DMA_CRCWriteByteOrderMaintain(DMA_ID_0);
    else if(SYS_DMA_CRC_WRITE_ORDER_CHANGE == crc.writeOrder)
        PLIB_DMA_CRCWriteByteOrderAlter(DMA_ID_0);
    PLIB_DMA_CRCDataWrite(DMA_ID_0, crc.data);
    PLIB_DMA_CRCXOREnableSet(DMA_ID_0, crc.xorBitMask);
    if(SYS_DMA_CHANNEL_CRC_MODE_APPEND == crc.mode)
       PLIB_DMA_CRCAppendModeEnable(DMA_ID_0);
    else
       PLIB_DMA_CRCAppendModeDisable(DMA_ID_0);

}
```

2. Setup the CRC computation for the configured DMA channel by calling the function *APP_DMA_ChannelCRCSet* as shown in the following code.

```
APP_DMA_ChannelCRCSet (DMA_CHANNEL_0, crc) ;
```

For comparing results, if a non-direct seed is required to be used for CRC computation, convert and use the direct seed to non-direct seed as shown in the following figure.

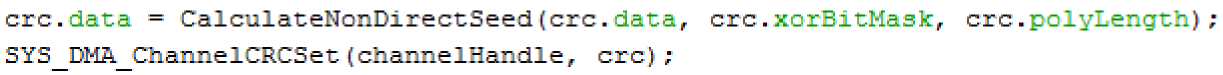
Figure 2-10.



```
uint32_t CalculateNonDirectSeed(uint32_t seed, uint32_t polynomial, uint8_t polynomialOrder)
{
    uint8_t lsb;
    uint8_t i;
    uint32_t msbmask;
    msbmask = ((unsigned long)1)<<(polynomialOrder-1);
    for (i=0; i<polynomialOrder; i++)
    {
        lsb = seed & 1;
        if (lsb) seed ^= polynomial;
        seed >>= 1;
        if (lsb) seed |= msbmask;
    }
    return seed; // return the non-direct CRC initial value
}
```

Before calling the function *SYS_DMA_ChannelCRCSet or APP_DMA_ChannelCRCSet,* convert the direct seed to non-direct by calling the function *CalculateNonDirectSeed* as shown in the following figure.

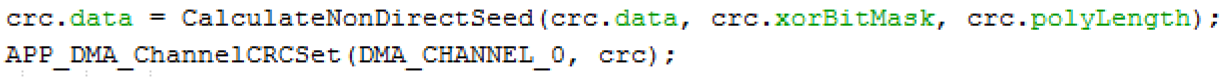
Figure 2-11.



```
crc.data = CalculateNonDirectSeed(crc.data, crc.xorBitMask, crc.polyLength);
SYS_DMA_ChannelCRCSet(channelHandle, crc);
```

Or

Figure 2-12.



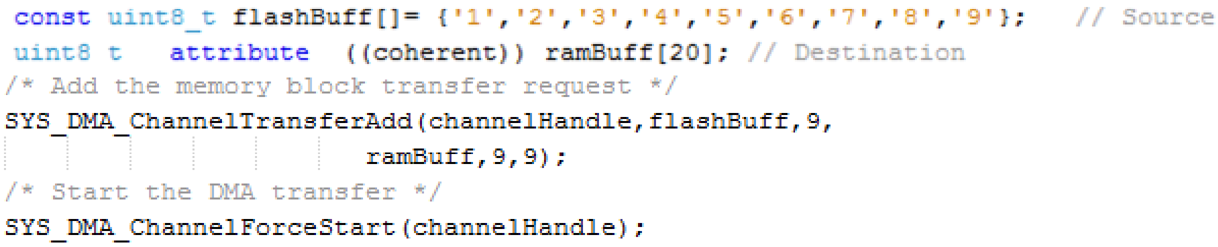
```
crc.data = CalculateNonDirectSeed(crc.data, crc.xorBitMask, crc.polyLength);
APP_DMA_ChannelCRCSet(DMA_CHANNEL_0, crc);
```

Setup and Initiate Data transfer:

Data transfer between memory to memory

Call the *SYS_DMA_ChannelTransferAdd* function to setup the data transfer, and initiate the data transfer by calling the function *SYS_DMA_ChannelForceStart*. To transfer 9 bytes of ASCII data “123456789” from Flash memory to RAM while computing the CRC as shown in the following figure.

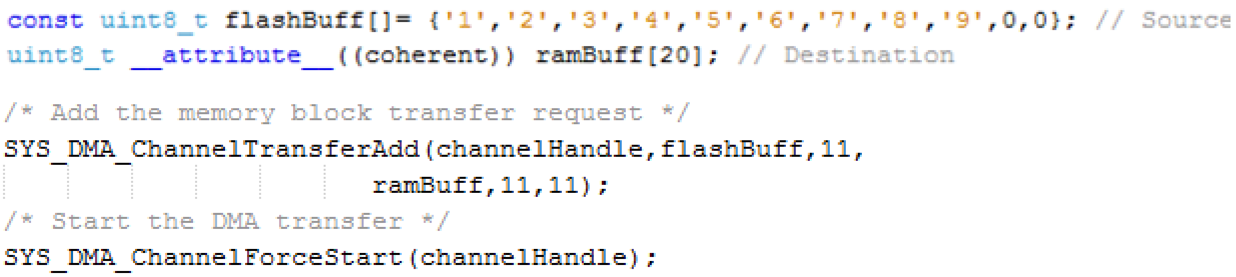
Figure 2-13.



```
const uint8_t flashBuff[]= {'1','2','3','4','5','6','7','8','9'};   // Source
uint8 t   attribute   ((coherent)) ramBuff[20]; // Destination
/* Add the memory block transfer request */
SYS_DMA_ChannelTransferAdd(channelHandle,flashBuff,9,
                           ramBuff,9,9);
/* Start the DMA transfer */
SYS_DMA_ChannelForceStart(channelHandle);
```

Note: If a non-direct seed is applied during the CRC computation (as explained in section ***Setup the CRC Computation*** above), append additional zero bits (16-bit or 32-bit, depending on the polynomial length) as shown in the following figure.

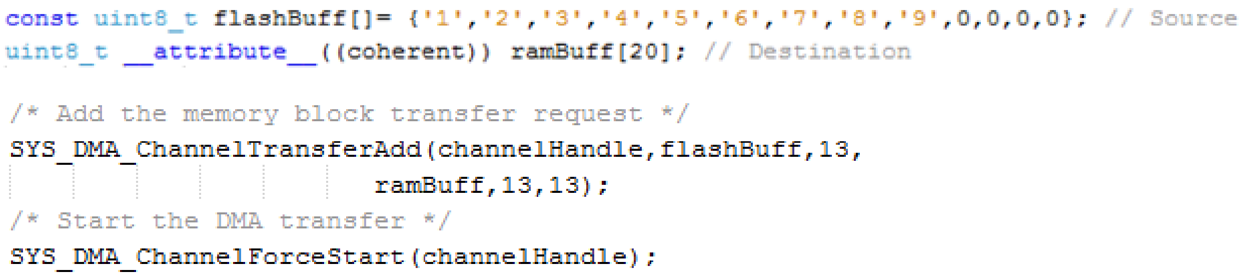
Figure 2-14. For a 16-bit Polynomial



```
const uint8_t flashBuff[]= {'1','2','3','4','5','6','7','8','9',0,0}; // Source
uint8_t __attribute__((coherent)) ramBuff[20]; // Destination

/* Add the memory block transfer request */
SYS_DMA_ChannelTransferAdd(channelHandle,flashBuff,11,
                                ramBuff,11,11);
/* Start the DMA transfer */
SYS_DMA_ChannelForceStart(channelHandle);
```

- The arguments source *(srcAddr)* and destination *(destAddr) to function call SYS_DMA_ChannelTransferAdd* needs to append 2 bytes of zeros (corresponding to the polynomial length).
- The arguments source size (*srcSize*), destination size(*destSize*) and cell size(*cellSize*) *to function call SYS_DMA_ChannelTransferAdd* needs to add value 2 (depending to the polynomial length).

Figure 2-15. For 32-bit Polynomial



```
const uint8_t flashBuff[]= {'1','2','3','4','5','6','7','8','9',0,0,0,0}; // Source
uint8_t __attribute__((coherent)) ramBuff[20]; // Destination

/* Add the memory block transfer request */
SYS_DMA_ChannelTransferAdd(channelHandle,flashBuff,13,
                                ramBuff,13,13);
/* Start the DMA transfer */
SYS_DMA_ChannelForceStart(channelHandle);
```

- The arguments source *(srcAddr)* and destination *(destAddr) to function call SYS_DMA_ChannelTransferAdd* needs to append 4 bytes of zeros (Corresponding to the polynomial length)
- The arguments source size (*srcSize*), destination size(*destSize*) and cell size(*cellSize*) *to function call SYS_DMA_ChannelTransferAdd* needs to add value 4 (depending to the polynomial length)

Data transfer between memory and a peripheral

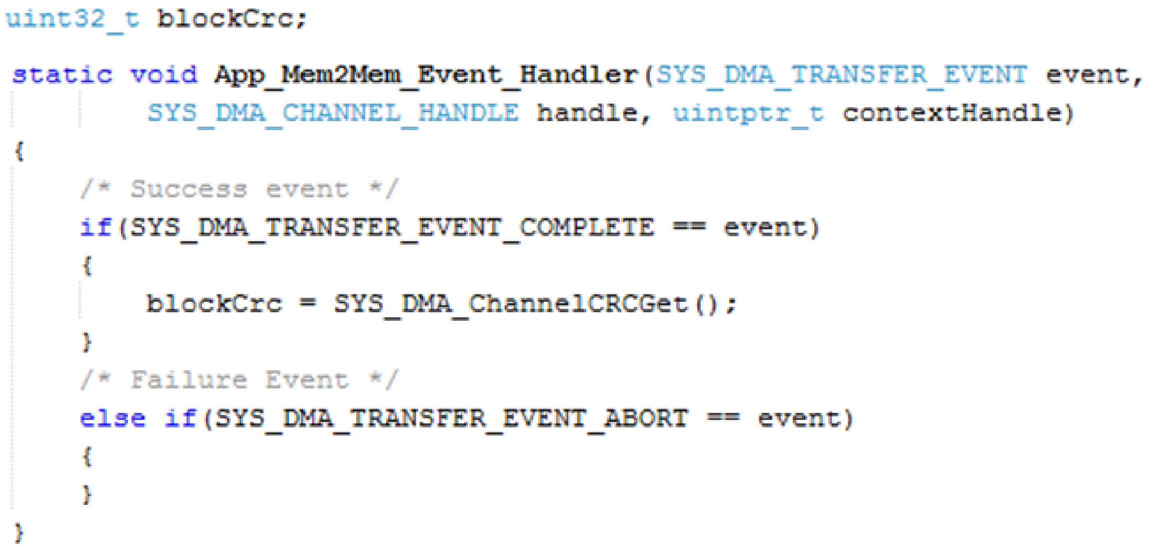
Call the peripheral driver function that sets up and enables the data transfer. If data is transferred from memory to the USART (for transmission) using the DMA in a buffered data model, the function *DRV_USART_BufferAddWrite needs to be called.*

Note: If a non-direct seed is applied during the CRC computation (as explained in section ***Setup the CRC Computation*** above), append additional zeros (2 or 4, depending on polynomial length) for the argument source (*buffer*), and add value (2 or 4, depending on polynomial length) for the argument size (*size*).

Observe the CRC computation result:

- If the data is transferred between memory-to-memory, on data transfer completion, the event handler is registered using the function *SYS_DMA_ChannelTransferEventhandlerSet would be called.* In the event handler, call the function *SYS_DMA_ChannelCRCGet* and observe the generated CRC.

Figure 2-16.



```
uint32_t blockCrc;

static void App_Mem2Mem_Event_Handler(SYS_DMA_TRANSFER_EVENT event,
        SYS_DMA_CHANNEL_HANDLE handle, uintptr_t contextHandle)
{
    /* Success event */
    if(SYS_DMA_TRANSFER_EVENT_COMPLETE == event)
    {
        blockCrc = SYS_DMA_ChannelCRCGet();
    }
    /* Failure Event */
    else if(SYS_DMA_TRANSFER_EVENT_ABORT == event)
    {
    }
}
```

- If the data transfer is between the memory and a peripheral, on data transfer completion, the event handler registered using the peripheral driver *would be called.* In the driver event handler, call the function *SYS_DMA_ChannelCRCGet* and observe the generated CRC.

3. Relevant Resources

For additional information, refer to these docs which are available for download form the following locations:

Cyclic Redundancy Check (CRC)

- http://ww1.microchip.com/downloads/en/AppNotes/01148a.pdf

CRC Generating and Checking

- http://ww1.microchip.com/downloads/en/AppNotes/00730a.pdf

32-Bit Programmable Cyclic Redundancy Check (CRC)

- http://ww1.microchip.com/downloads/en/DeviceDoc/60001336c.pdf

Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks

- http://users.ece.cmu.edu/~koopman/roses/dsn04/koopman04_crc_poly_embedded.pdf

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.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

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

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	[X][(1)]	–X	/XX
Device	Tape and Reel	Temperature Range	Package

Device:	Device A, Feature A, (Package A) Device B, Feature B, (Package B)	
Tape & Reel Option:	Blank	= Tube
	T	= Tape & Reel
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
Package:	AA	= Package AA
	BB	= Package BB

Examples:

- MCPXXXXXAT-E/AA: Tape and Reel, Extended temperature, XAA package
- MCPXXXXXBT-E/BB: Tape and Reel Extended temperature, XBB package

Note:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check http://www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

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

Trademarks

ISBN: 978-1-5224-2933-3

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

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
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

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

China - Beijing
Tel: 86-10-8569-7000

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

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

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

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

China - Hong Kong SAR
Tel: 852-2943-5100

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

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

China - Shanghai
Tel: 86-21-3326-8000

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

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

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

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

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

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

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

India - Pune
Tel: 91-20-4121-0141

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

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

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

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

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

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

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

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

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

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

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

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

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

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

Sweden - Gothenberg
Tel: 46-31-704-60-40

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

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