CRC Training Outline

1. Module Overview
2. On-chip interconnects and inter-module dependencies
3. Software configuration
4. Typical use cases
5. Demo code explanation
6. Frequently asked question list
7. Reference material

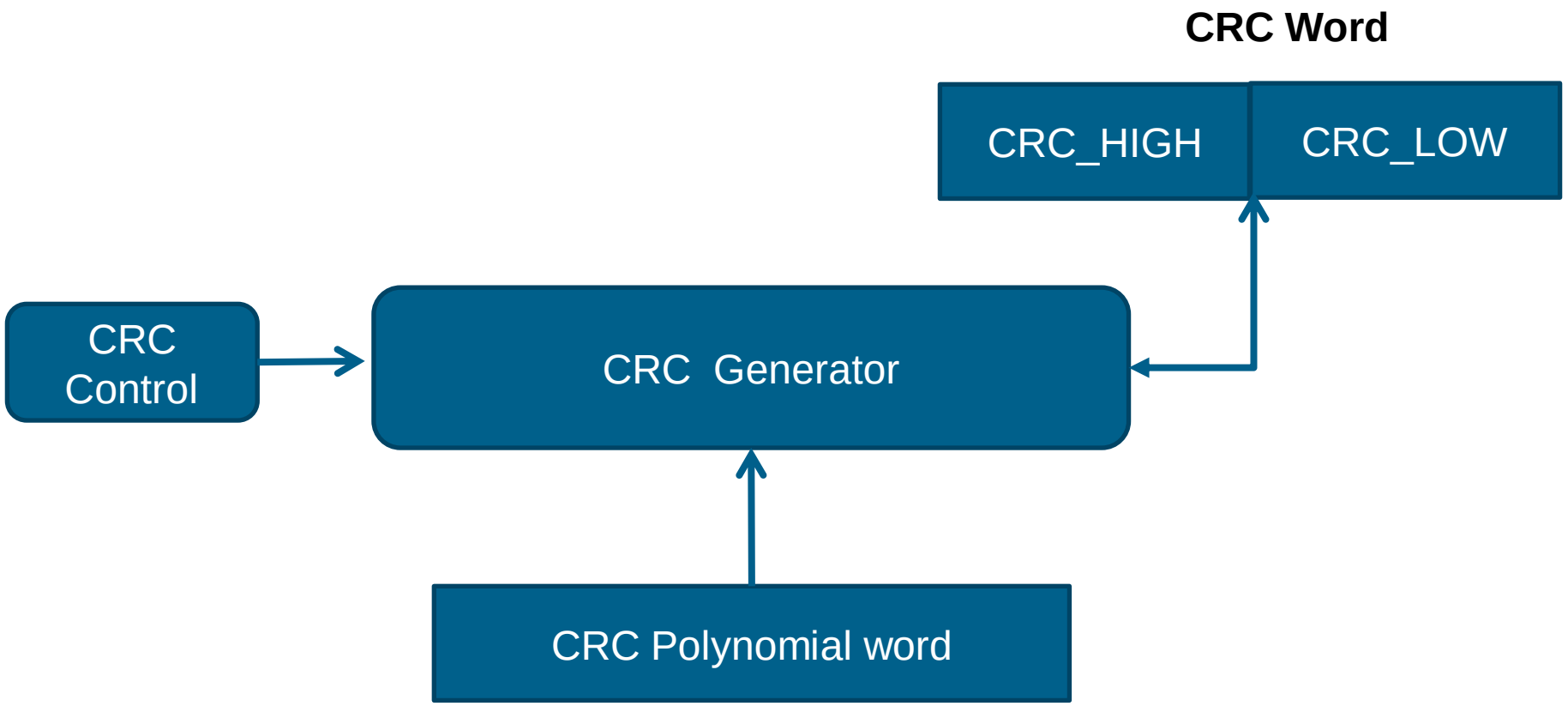
freescale™ semiconductor

CRC Training Outline

1. **Module Overview**
2. On-chip interconnects and inter-module dependencies
3. Software configuration
4. Typical use cases
5. Demo code explanation
6. Frequently asked question list
7. Reference material

freescale semiconductor

Block Diagram



X = a*b+c; // input data
X mod poly = c mod poly
where c = CRC of X

freescale semiconductor

Feature list

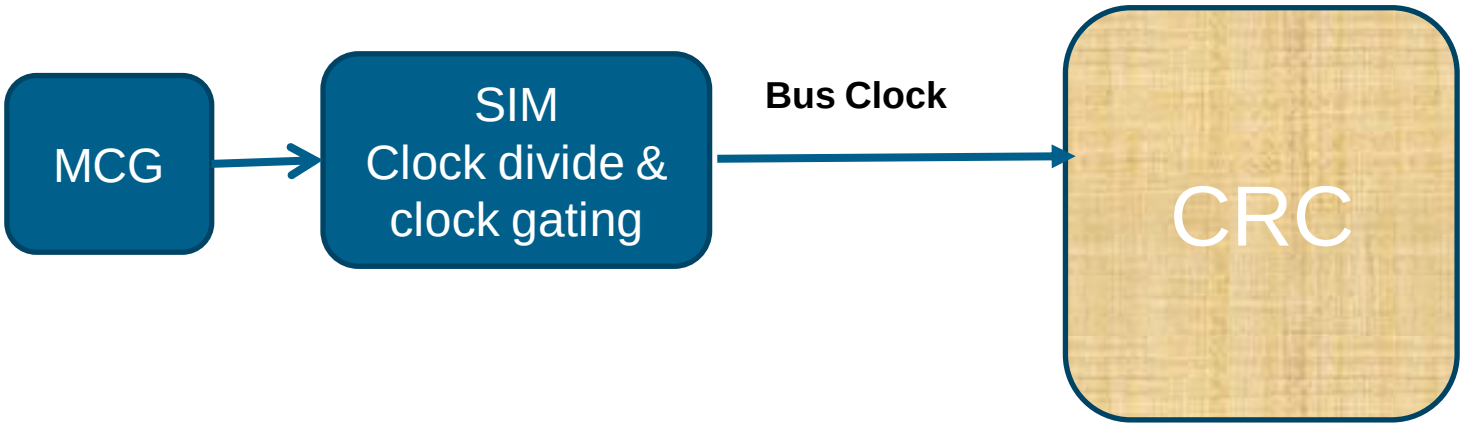
- Cyclic redundancy check (CRC) generates 16/32-bit CRC code for error detection
- Hardware CRC generator circuit using 16-bit or 32-bit (programmable) shift register
- Programmable initial seed value and Polynomial.
- Transpose input data and CRC result via transpose register, required for certain CRC standards
- Final XOR of the output. Some CRCs have final XOR of their CRC checksum with 0xFFFFFFFF or 0xFFFF in their protocol

CRC Training Outline

1. Module Overview
2. **On-chip interconnects and inter-module dependencies**
3. Software configuration
4. Typical use cases
5. Demo code explanation
6. Frequently asked question list
7. Reference material

freescale semiconductor

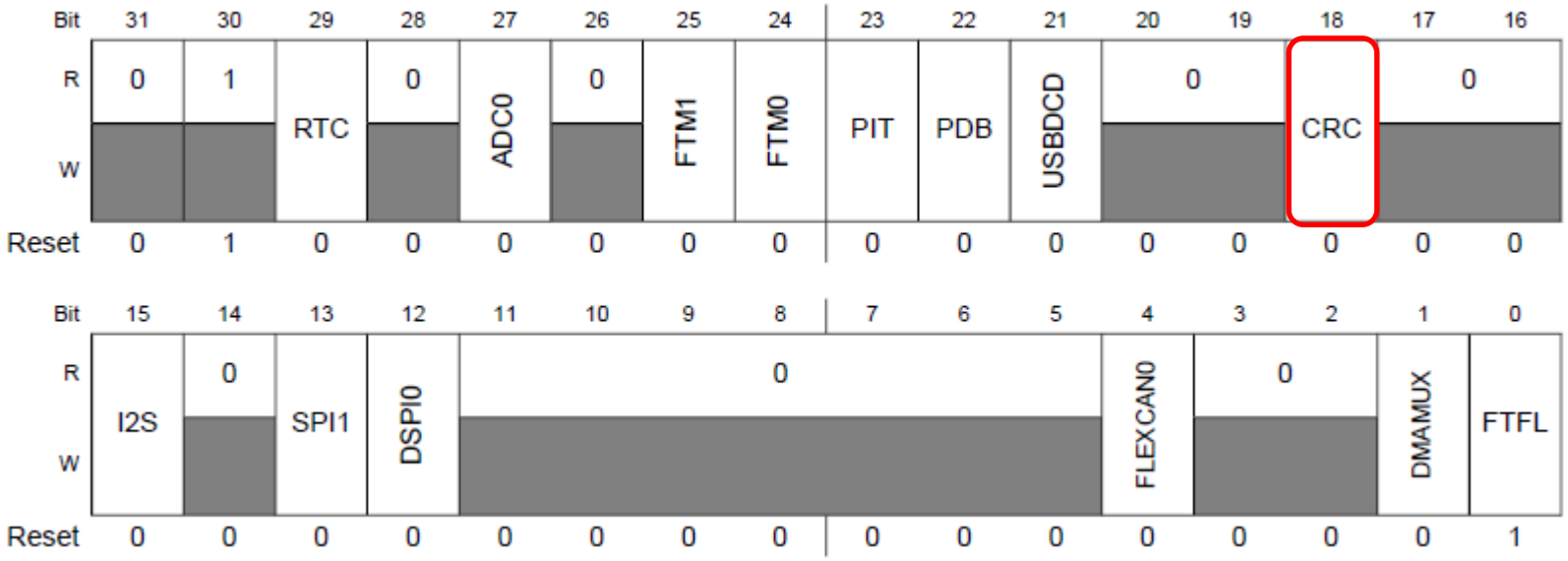
SoC interconnect diagram



freescale semiconductor

Module dependencies

- *Clock gating*
  - Prior to using CRC, SIM_SCGC6[CRC] bit must be set



freescale semiconductor

CRC Training Outline

1. Module Overview
2. On-chip interconnects and inter-module dependencies
3. **Software configuration**
4. Typical use cases
5. Demo code explanation
6. Frequently asked question list
7. Reference material

freescale semiconductor

Memory Map for Registers

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
4003_2000	CRC Word Register (CRC_CRC)	32	R/W	FFFF_FFFFh
4003_2004	Polynomial Word Register (CRC_GPOLY)	32	R/W	0000_1021h
4003_2008	Control Register (CRC_CTRL)	32	R/W	0000_0000h

```
#define CRC_BASEADDR  0x40032000  // CRC Starting Address
#define CRC_CRC                   (*(vuint32_t*)(CRC_BASEADDR + 0x0000))
#define CRC_CRC_HIGH              (*(vuint16_t*)(CRC_BASEADDR + 0x0002))
#define CRC_CRC_LOW               (*(vuint16_t*)(CRC_BASEADDR + 0x0000))
#define CRC_CRC_HU                (*(vuint8_t*)(CRC_BASEADDR + 0x0003))   // high word upper byte
#define CRC_CRC_HL                (*(vuint8_t*)(CRC_BASEADDR + 0x0002))   // high word low byte
#define CRC_CRC_LU                (*(vuint8_t*)(CRC_BASEADDR + 0x0001))   // low word upper byte
#define CRC_CRC_LL                (*(vuint8_t*)(CRC_BASEADDR + 0x0000))   // low word low byte
#define CRC_GPOLY                 (*(vuint32_t*)(CRC_BASEADDR + 0x0004))
#define CRC_CTRL                  (*(vuint32_t*)(CRC_BASEADDR + 0x0008))
```

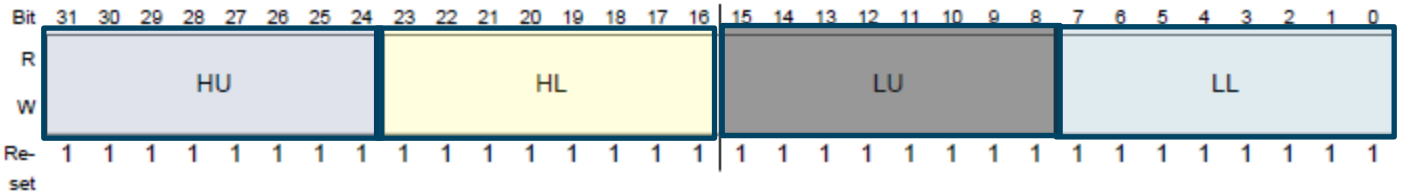
Sequence of register setup

Step1	• Program the CTRL register according to the configuration required
Step2	• Program the CTRL[WAS] with 1 to indicate the state machine that the write to the CRC register is a SEED
Step3	• Program the GPOLY register to specify poly
Step4	• Program the 32-bit of CRC register
Step5	• Disable the CTRL[WAS] with 0
Step6	• Write data that will be used in the CRC calculation
Step7	• Read the CRC from the CRC register

CRC_CRC Access Requirement

- Data bytes written into CRC_CRC register must be contiguous
- Valid access to CRC_CRC will be
  - Single byte data
    - Can be written to any CRC_CRC byte

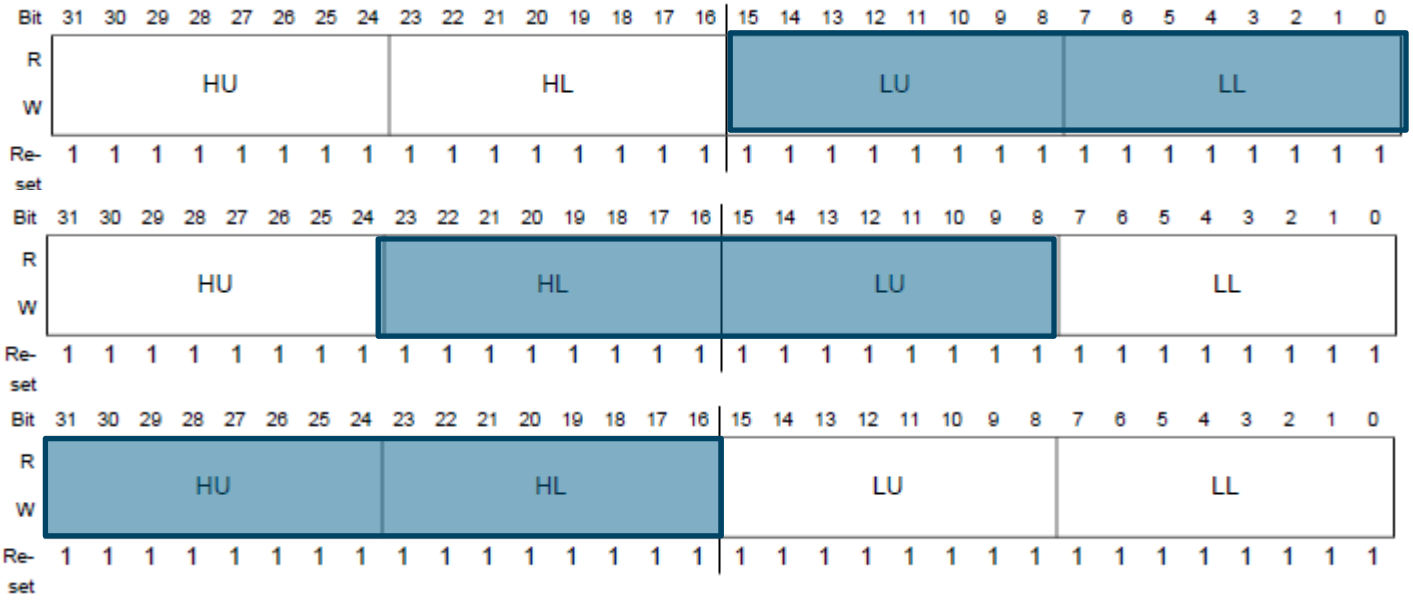
  Register



freescale semiconductor

CRC_CRC Access Requirement

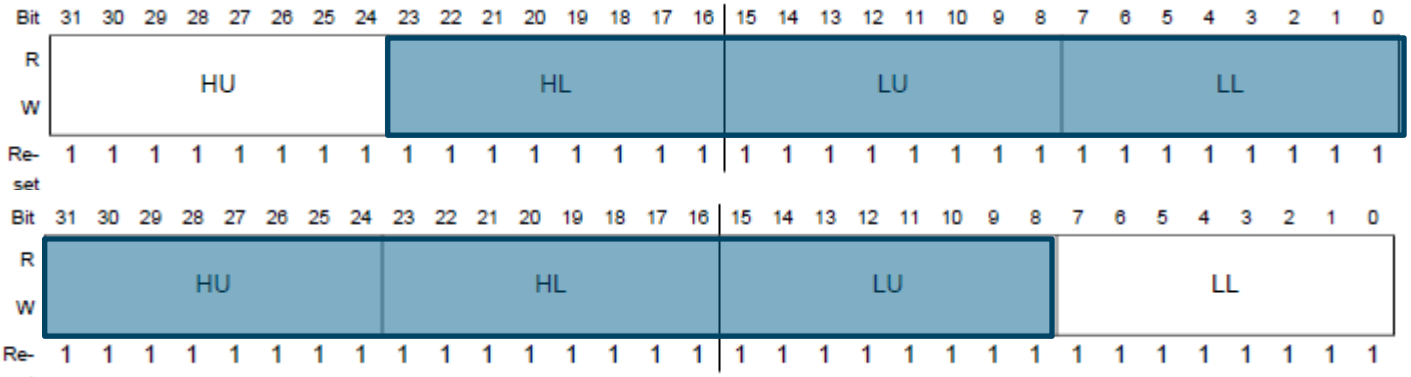
- Data bytes written into CRC_CRC register must be contiguous
- Valid access to CRC_CRC will be
  - Two-byte data
    - Can only be written to contiguous CRC_CRC byte registers with the following combinations:
      - CRC_CRC_LL & CRC_CRC_LU (or CRC_CRC_LOW)
      - CRC_CRC_LU & CRC_CRC_HL
      - CRC_CRC_HL & CRC_CRC_HU (or CRC_CRC_HIGH)



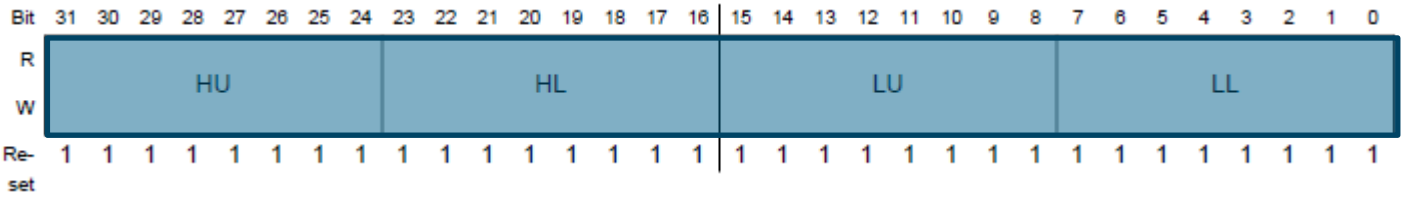
freescale semiconductor

CRC_CRC Access Requirement

- Data bytes written into CRC_CRC register must be contiguous
- Valid access to CRC_CRC will be
  - Three-byte data
    - Can only be written to contiguous CRC_CRC byte registers
      - CRC_CRC_HL & CRC_CRC_LU & CRC_CRC_LL (or CRC_CRC_HL & CRC_CRC_LOW)
      - CRC_CRC_HU & CRC_CRC_HL & CRC_CRC_LU (or CRC_CRC_HIGH & CRC_CRC_LU)



  - Four-
    - Can be written to the whole CRC_CRC register



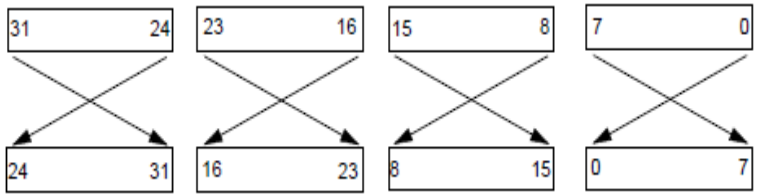
freescale semiconductor

Transpose

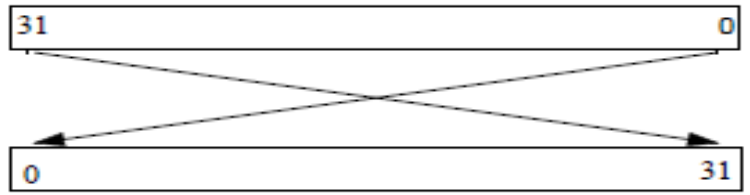
- Used for transforming data format in between Big Endian and Little Endian
- Transposition can happen while writing input data and/or reading CRC register
- CTRL[TOT] bit controls transposition while writing input data
- CTRL[TOTR] bit controls transposition while reading CRC
- CTRL[TOT] = 0, no transpose
- CTRL[TOTR] = 0, no transpose

Transpose Types

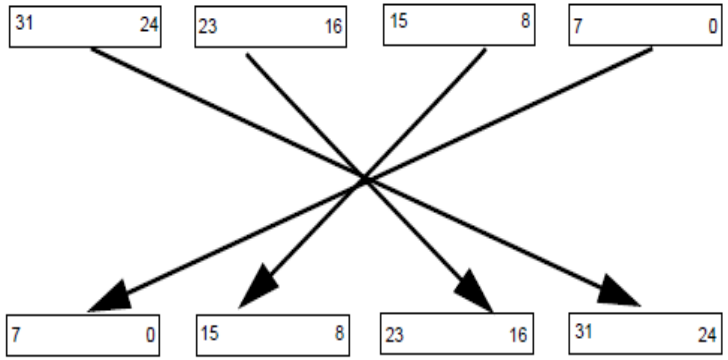
Transpose only bits in each byte
CTRL[TOT/TOTR] =1



Transpose both bits & bytes
CTRL[TOT/TOTR] =2



Transpose bytes only
CTRL[TOT/TOTR] =3



freescale™ semiconductor

Typical Use Cases

Name	Poly	Seed	Final XOR?	Type of Transpose for Input	Type of Transpose for CRC Read	Standards
CRC-16	0x1021	0xFFFF (ITU-T V.41) 0x0000 (ITU-T T.30, X.25)	No	No transpose	No transpose	CRC-CCITT, ADCCP, SDLC/HDLC
CRC-16	0x1021	0	No	Transpose only bits in a byte	Transpose only bits in a byte	CRC-CCITT (Kermit)
XMODEM	0x8408	0x0000	No	Transpose only bits in a byte	Transpose only bits in a byte	XMODEM
ARC	0x8005	0x0000	No	Transpose only bits in a byte	Transpose only bits in a byte	ARC (zip file)
CRC-32	0x04C11DB7	0xFFFFFFFF	Yes	Transpose only bits in a byte	Transpose both bits and bytes	PKZIP, AUTODIN II, Ethernet, FDDI

CRC Training Outline

1. Module Overview
2. On-chip interconnects and inter-module dependencies
3. Software configuration
4. Typical use cases
5. **Demo code explanation**
6. Frequently asked question list
7. Reference material

freescale semiconductor

Hands on Demo

- *CRC_Demo lab will guide you how to use CRC*
- *It uses Terminal to show and enter different settings*
- *Terminal is configured as 115200bps,N-8-1 format, non- XON/XOFF protocol*
- *The first option is shown below*

Please select CRC width (16-bit/32-bit):
1. CRC16
2. CRC32
select: 2

- *After selecting one option, the following messages appear:*

Please select CRC polynomial:
1. poly = 0x1021 (CRC-CCITT)
2. poly = 0x8408 (XMODEM)
3. poly = 0x8005 (ARC)
4. poly = 0x04C11DB7 (CRC32)
5. others
select: 4

Hands on Demo

- *Then ask you to select one of transpose type for input:*

Please select type of Transpose for input:
1. No Transposition
2. Only transpose bits in a byte
3. Transpose both bits and bytes
4. Only transpose bytes
select:2

- *After selecting one option, the following messages appear:*

Please select type of Transpose for Read:
1. No Transposition
2. Only transpose bits in a byte
3. Transpose both bits and bytes
4. Only transpose bytes
select:3

Hands on Demo

- *When asked with “XOR final checksum(y/n)?”, key ‘y’/’n’ in little case*
- *Then you’ve to enter seed in hex format, either low case or upper case.*
- *Now, it asks you to enter input data in ASCII format.*
- *Then CRC result will be printed*
- *Press any key to continue or ‘q’ to quit the demo*
- *The following screen shot is an example*

```
XOR final checksum (y/n)?y
Please enter seed in hex:ffffffff
Please enter an ASCII Message:123456789
 CRC result = 0xCBF43926
Press any key to continue...,'q' to quit!
```

CRC Training Outline

1. Module Overview
2. On-chip interconnects and inter-module dependencies
3. Software configuration
4. Typical Use Cases
5. Demo code explanation
6. **Frequently asked question list**
7. Reference material

freescale semiconductor

CRC Preliminary FAQ

- **Q: Can CRC support bit reflect for input data?**
  A: Yes, it supports bit reflect for both input data and output CRC. It is controlled by CTRL[TOT] or CTRL[TOTR] bits.

- **Q: Can CRC support little endian to big endian byte format?**
  A: Yes, it supports little endian to big endian byte format. It is controlled by CTRL[TOT] or CTRL[TOTR] bits.

- **Q: Can CRC support currently known CRC standards?**
  A: yes, it supports both CRC-16 and CRC-32 with different seeds and reflection types as well as final XORing 0xFFFF (FFFF) feature for result.

CRC Training Outline

1. Module Overview
2. On-chip interconnects and inter-module dependencies
3. Software configuration
4. Hardware configuration/considerations
5. Demo code explanation
6. Frequently asked question list
7. **Reference material**

freescale semiconductor

CRC References

http://en.wikipedia.org/wiki/Cyclic_redundancy_check


freescale™
semiconductor