X11/1998-32

MUMPS Development Committee

Extension to the MDC Standard
Type A Release of the MUMPS Development Committee

Cyclic Redundancy Code functions

September 19, 1998

Produced by the MDC Subcommittee #13
Data Management and Manipulation

Art Smith, Chairman
MUMPS Development Committee

Rod Dorman, Chairman
Subcommittee #13

The reader is hereby notified that the following MDC specification has been approved by the MUMPS Development Committee but that it may be a partial specification that relies on information appearing in many parts of the MDC Standard. This specification is dynamic in nature, and the changes reflected by this approved change may not correspond to the latest specification available.

Because of the evolutionary nature of MDC specifications, the reader is further reminded that changes are likely to occur in the specification released, herein, prior to a complete republication of the MDC Standard.

1. Identification of proposed change

1.1 Title

Cyclic Redundancy Code functions

1.2 MDC Proposer and Sponsor

This proposal originates from David Brown, current sponsor and author are:
SC13/TG2 – String Handling
Chair: David Whitten
Author: Rod Dorman
Polylogics Consulting, Inc. phone: (201) 794-4444
13-15 Broadway fax: (201) 794-4455
Fair Lawn NJ 07410 email: rodd@polylogics.com

1.3 Motion

None, this is the final MDC version which supersedes X11/SC13/TG2/1998-7

1.4 History

October 1998	X11/1998-32	Final version
September 1998	X11/SC13/TG2/1998-7	History and discussion updated, submitted for elevation to MDC Type A. passed 15:0:1
June 1998	X11/SC13/TG2/1998-4	Split into 2 proposals, improved discussion in 3.1, filled in example code, submitted for elevation to SC13 Type A. passed 11:0:4
March 1998	X11/SC13/TG2/1998-2	Proposed as library functions to replace the SCRC type C, taskgroup proposed it for SC13 Type B, passed 15:2:2
September 1997	X11/SC13/93-9	Taskgroup discussion on rescinding. Rod Dorman volunteers to author this proposal.
February 1993	X11/SC13/93-9	Reference to CCITT Recommendation found and entered into document and presented to replace existing type B.
January 1991	X11/SC1/90-109	Edited to conform to proposal format. Presented for elevation to SC#1 Type A. Failed acceptance because reference to correct standard was missing.
January 1988	...	Accepted as type B
June 1987	X11.1/87-16	First submitted, not acted on.

1.5 Dependencies

None

2. Justification of proposed change.

2.1 Needs

While various network protocols take care of a lot of high-end type machine connections (DECnet, TCP/IP, SNA, etc), there remains a need for the M programmer to securely send messages between machines which are not connected via the high-level networks. There is also the need to generate hash values of multi line strings (e.g. name and address) to efficiently check for duplicates. Cyclic Redundancy Code functions can satisfy both these needs.

2.2 Existing practice in Area of Proposed Change

Some vendors have specific $Z... functions for calculating checksums and CRC's. Some users use the $ZCALL facility in their M implementation to achieve this capability. Others simply use a FOR loop to calculate an ASCII checksum which will not catch as many errors as a CRC (e.g. transposition).

3. Description of Proposed Change

3.1 General description of proposed change

Adds the following Cyclic Redundancy Code library functions to the STRING library:

libraryelement	CRC Name	CRC Polynomial
CRC16	CRC-16	$X^{16}+X^{15}+X^{2}+1$
CRCCCITT	CRC-CCITT	$X^{16}+X^{12}+X^{5}+1$
CRC32	CRC-32	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$

Both CRC-16 and CCRC-CCITT are used for 8 bit transmission streams and both result in 16 bit values, they are widely used in the USA and Europe respectively and give adequate protection for most applications. Applications that need extra protection can make use of the CRC-32 which generates 32 bit values. The CRC-32 is used by the local network standards committee (IEEE-802) and in some DOD applications.

The basic idea of CRC algorithms is to treat the message as an enormous binary number and divide it by another fixed binary number. The remainder of this division is the CRC value. However, instead of the divisor, dividend (message), quotient, and remainder being viewed as positive integers, they are viewed as polynomials with binary coefficients and the division is done in "polynomial arithmetic mod 2" which is simply binary arithmetic mod 2 with no carries. Addition and subtraction are equivalent to an exclusive or (XOR). Multiplication and division become XORing (adding & subtracting) while shifting.

Looking at it from a hardware point of view, there's a w-bit wide shift register (where w is the width of the desired CRC) where message bits are shifted into the low order end and whenever a 1-bit is shifted out the high order end (into the link or carry bit) it's XORed into specific bits of the register. The bit positions that are affected are defined by the polynomial.

In addition to the polynomial, a specific CRC algorithm must define the initial value for the register, whether the message bytes are shifted into the register low order bit

first or high order, whether the final register value is shifted out low order bit first or high order, and finally whether the final register value is complemented.

Most software implementations make use of the fact that XOR is associative and commutative and that shifting distributes over XOR in order to speed up the calculation. In particular, it's possible avoid shifting each bit of a message byte by pre-calculating a lookup table.

This description has only scratched the surface of CRC algorithms and polynomial arithmetic. The reader (implementers in particular) is encouraged to read "A Painless Guide to CRC Error Detection Algorithms" as referenced in section 6.

3.2 Annotated examples of use

```
SET (I,C)=0
FOR  SET I=$O(X(I)) QUIT:'I  SET C=$$CRC32^STRING(X(I),C)

SET Data=$E(Msg,1,$L(Msg)-2)
SET MsgCRC=$A(Msg,$L(Msg)-1)*256+$A(Msg,$L(Msg))
IF $$CRC16^STRING(Data)'=MsgCRC DO TransmissionError
```

3.3 Formalization (References are to X11.1 Draft Standard Version 13, X11/TG6/98-1)

In Section 1 Clause 7.1.6.6 (STRING Library Elements), add three new library elements:

3.3.1.1 Library Element Description

CRC-16, a sixteen (16) bit Cyclic Redundancy Code library function

3.3.1.2 Definition

CRC16^STRING : INTEGER (S : STRING, SEED : INTEGER : O)
This function computes a Cyclic Redundancy Code of the 8-bit character string S using $X^{16}+X^{15}+X^2+1$ as the polynomial. The optional SEED parameter supplies an initial value, which allows running CRC calculations on multiple strings. If missing, a default value of zero is used. The message bytes are considered shifted in low order bit first and the return value shifted out low order bit first.

3.3.1.3 Domain

S	STANDARD	
SEED	$0 \le SEED < 2^{16}$	When SEED is outside its domain, an error condition occurs with ecode = "M28"

3.3.1.4 Range

$0 \le result < 2^{16}$

3.3.1.5 Side Effects

None

3.3.1.6 Example of M[UMPS] code to implement

```
CRC16(string,seed)     ; CRC-16
      ; Polynomial = x^16 + x^15 + x^2 + x^0
      NEW I,J,R
      IF '$D(seed) SET R=0
      ELSE  IF seed'<0,seed'>65535 SET R=seed\1
      ELSE  SET $ECODE=",M28,"
      FOR I=1:1:$L(string) DO
      . SET R=$$XOR($A(string,I),R,8)
      . FOR J=0:1:7 DO
      . . IF R#2 SET R=$$XOR(R\2,40961,16)
      . . ELSE  SET R=R\2
      QUIT R
      ;
XOR(a,b,w) ;
      NEW I,M,R
      SET R=b
      SET M=1
      FOR I=1:1:w SET:a\M#2 R=R+$S(R\M#2:-M,1:M) SET M=M+M
      QUIT R
```

3.3.1.7 Note to implementers

There is extensive literature available in the field discussing optimization of CRC calculation via table-driven algorithms.

3.3.2.1 Library Element Description

CRC-CCITT, a sixteen (16) bit Cyclic Redundancy Code library function

3.3.2.2 Definition

CRCCCITT^STRING : INTEGER (S : STRING, SEED : INTEGER : O)
This function computes a Cyclic Redundancy Code of the 8-bit character string S using $X^{16}+X^{12}+X^5+1$ as the polynomial. The optional SEED parameter supplies an initial value, which allows running CRC calculations on multiple strings. If missing, a default value of 65535 (2^{16}-1) is used. The message bytes are considered shifted in high order bit first and the return value shifted out high order bit first.

3.3.2.3 Domain

S	STANDARD	
SEED	$0 \le SEED < 2^{16}$	When SEED is outside its domain, an error condition occurs with ecode = "M28"

3.3.2.4 Range

$0 \le result < 2^{16}$

3.3.2.5 Side Effects

None

3.3.2.6 Example of M[UMPS] code to implement

```
CRCCCITT(string,seed)    ; CRC-CCITT
      ; Polynomial = x^16 + x^12 + x^5 + x^0
      NEW I,J,R
      IF '$D(seed) SET R=65535 ; FFFF = 2^16 -1
      ELSE  IF seed'<0,seed'>65535 SET R=seed\1
      ELSE  SET $ECODE=",M28,"
      FOR I=1:1:$L(string) DO
      . SET R=$$XOR($A(string,I)*256,R,16)
      . FOR J=0:1:7 DO
      . . SET R=R+R
      . . QUIT:R<65536  ; (2^16)
      . . SET R=$$XOR(4129,R-65536,13)
      QUIT R
      ;
XOR(a,b,w) ;
      NEW I,M,R
      SET R=b
      SET M=1
      FOR I=1:1:w SET:a\M#2 R=R+$S(R\M#2:-M,1:M) SET M=M+M
      QUIT R
```

3.3.2.7 Note to implementers

There is extensive literature available in the field discussing optimization of CRC calculation via table-driven algorithms.

3.3.3.1 Library Element Description

CRC-32, a thirty-two (32) bit Cyclic Redundancy Code library function

3.3.3.2 Definition

CRC32^STRING : INTEGER (S : STRING, SEED : INTEGER : 0)
This function computes a Cyclic Redundancy Code of the 8-bit character string S using $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$ as the polynomial. The optional SEED parameter supplies an initial value, which allows running CRC calculations on multiple strings. If missing, a default value of zero is used. The SEED is ones-complemented before being used, the message bytes are considered shifted in low order bit first, the return value is ones-complemented and shifted out low order bit first.

3.3.3.3 Domain

S	STANDARD	
SEED	$0 \le SEED < 2^{32}$	When SEED is outside its domain, an error condition occurs with ecode = "M28"

3.3.3.4 Range

$0 \le result < 2^{32}$

3.3.3.5 Side Effects

None

3.3.3.6 Example of M[UMPS] code to implement

```
CRC32(string,seed)       ; CRC-32
      ; Polynomial = 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+x^0
      NEW I,J,R
      IF '$D(seed) SET R=4294967295 ; 0xFFFFFFFF = 2^32 - 1
      ELSE  IF seed'<0,seed'>4294967295 SET R=4294967295-seed
      ELSE  SET $ECODE=",M28,"
      FOR I=1:1:$L(string) DO
      . SET R=$$XOR($A(string,I),R,8)
      . FOR J=0:1:7 DO
      . . IF R#2 SET R=$$XOR(R\2,3988292384,32)
      . . ELSE  SET R=R\2
      QUIT 4294967295-R ; 32-bit ones complement
      ;
XOR(a,b,w) ;
      NEW I,M,R
      SET R=b
      SET M=1
      FOR I=1:1:w SET:a\M#2 R=R+$S(R\M#2:-M,1:M) SET M=M+M
      QUIT R
```

3.3.3.7 Note to implementers

There is extensive literature available in the field discussing optimization of CRC calculation via table-driven algorithms.

4. Implementation Effects

4.1 Effect on Existing User Practices and Investments

Allows a standard way to do CRC calculations instead of relying on $Z... functions.

4.2 Effect on Existing Vendor Practices and Investments

None beyond implementation it. Note that many vendors already have $Z... CRC functions so all they will have to do is call it in a different way.

4.3 Techniques and Costs for Compliance Verification

Generate CRC values and compare to expected results.

4.4 Legal Considerations

None known

5. Closely Related Standards Activities

5.1 Other X11 Proposals Under Consideration

None known.

5.2 Other Related Standards Efforts

None known.

5.3 Recommendations for Coordinating Liaison

None.

6. Associated Documents

CRC-12 Cyclic Redundancy Code function X11/SC13/TG2/1998-5
A Painless Guide To CRC Error Detection Algorithms,
ftp.adelaide.edu.au/pub/rocksoft/crc_v3.txt

7. Issues, Pros and Cons, and Discussion

March 1998 MDC meeting

Pro	Con
1. Needed functionality	1. Excess baggage
2. More robust	
3. Useful for more than just communications	

Re the only con, this is a matter of personal opinion, the task group considers the functionality useful and apparently most of SC13 thinks so too.

June 1998 MDC meeting

Pro	Con
1. Needed functionality	(none specified)

September 1998 MDC meeting

Pro	Con
1. Needed functionality	(none specified)
2. Useful for more than just communications	