TECHNICAL DOCUMENT

REAL TIME DATA

CURRENCY DERIVATIVES MARKET
(LEVEL 1)

24 APR 2009

DOTEX INTERNATIONAL LIMITED
EXCHANGE PLAZA,
PLOT NO. C/1, G BLOCK,
BANDRA-KURLA COMPLEX,
BANDRA (E), MUMBAI 400 051.
INDIA.

COPYRIGHT NOTICE

CONTENTS

REAL TIME DATA TECHNICAL SPECIFICATION

CURRENCY DERIVATIVES MARKET

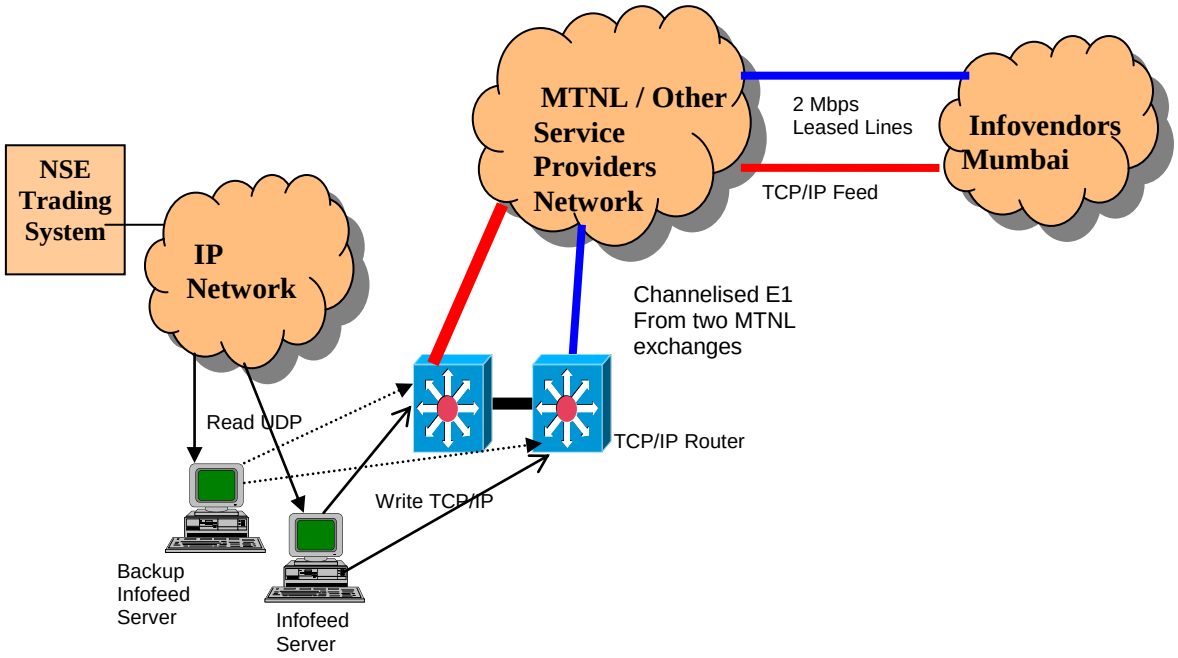
1. INTRODUCTION

DotEx international Ltd. disseminates NSEIL's real time broadcast data to various information agencies. It provides the 3 different types of data to vendors, i.e. Real Time Data, Snapshot Data and End of Day Data. The real time data is a packet broadcast available in TCP/IP format, where as the snapshot data and End of day data is available in the form of files. The Infofeed server provides NSEIL real time broadcast data. The information agencies connect to the Infofeed Server through 2Mbps Leased Lines. These leased lines are terminated on Infofeed Router and their data specific pneumonic calls are forwarded to Infofeed server. The Infofeed server accepts these pneumonic calls and creates a socket connection. The TCP/IP data flows to the information agencies through these socket connections.

2. CONNECTION DETAILS

2.1. STRUCTURAL DIAGRAM

The structural diagram of Infofeed connection has been explained below -



2.2 Online Requirements

a. A Router / Switch or a card with TCP/IP capabilities to connect to 2Mbps transmission lines for receiving NSEIL's Real time information.

b. The Information agency should develop applications that initiate TCP/IP calls through 2Mbps Leased Line.

2.3 Steps for Authentication AND Receiving feed

a) Client applications at vendor end, establish the connection with Infofeed Server application using specified IP address and Port.
b) After establishing the connection, client application sends the login packet to Infofeed server application.

Packet format of Login packet is as follows,

```
struct LOGIN_REQ
{
        struct INFO_HEADER sHeader;
        struct LOGIN_REQ_DATA sData;
        struct INFO_TRAILER sTrailer;
};
struct LOGIN_REQ_DATA {
        char cUserId[10];
        char cPassword[8];
        char cNewPassword[8];
        char cConfmPassword[8];
};
struct INFO_HEADER
{
        short iCode;
        short iLen;
        LONG lSeqNo;
};
struct INFO_TRAILER
{
        short iCheckSum;
        CHAR cEOT;
};
```

c) Password field is case sensitive, password should be minimum 6 characters long, password and user id should not be same, password should start with alphabet and password should be alphanumeric (No wild characters are allowed).
d) If user wants to change his password then the user needs to specify new password & confirm password (Both fields should match) otherwise leave it blank.
e) Based on the above information, user will get Log on response from Infofeed Server.

```
struct LOGIN_RESPONSE
{
	struct COM_HEADER sCOMHeader;
	struct LOGIN_RESPONSE_DETAIL sDetail;
};
struct COM_HEADER
{
	char cCmopOrNot;
	short iPackLen;
	short iNoOfPack;
};
struct LOGIN_RESPONSE_DETAIL
 {
	struct INFO_HEADER sHeader;
	struct LOGIN_RESPONSE_DATA sData;
	struct INFO_TRAILER sTrailer;
};
struct LOGIN_RESPONSE_DATA
{
	long iErrCode;
	char cMesg[50];
};
```

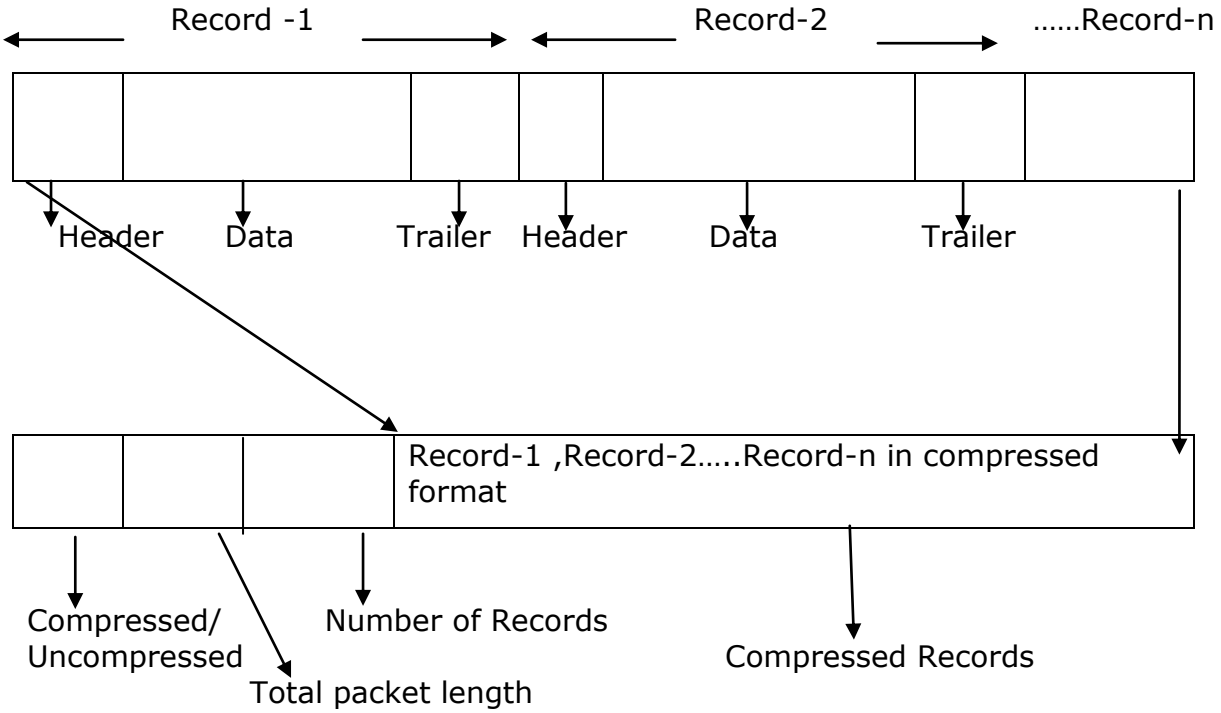
Following Error code will be returned which client needs to interpret as:-

1000- Successful
1001- Password Update Successfully
1002- Wrong UserId-Password Combination
1003- Password is not valid in password change request.
1004- Login request is not correct

Error code other than above - Error in receiving logon response

f) After successful login, Infofeed Application starts sending packets in the below format.

Packet Format:



Compressed/Uncompressed: This field tells whether packet is compressed or not compressed.
If this Field = 0 then Compressed.
Field = 1 then Uncompressed.

Number of Records: This field tells the number of records present in the compressed packets.

Packet length: this field specifies the total packet length.

```
Structure COM_HEADER
{
        char cCmopOrNot;
        short iPackLen;
        short iNoOfPack;
};
```

g) As the data packets are sent in compressed format there is a need to decompress them. The compression algorithm used is LZO.
h) The Decompression algorithm used should be LZO

2.4 Steps for Decompressing feed

- LZO Algorithm Details:

a) LZO is a data compression library which is suitable for data de-/compression in real-time. This means it favors speed over compression ratio.
b) LZO is written in ANSI C. Both the source code and the compressed data format are designed to be portable across platforms.

LZO implements a number of algorithms with the following feature

- Decompression is simple and *very* fast.
- Requires no memory for decompression.
- Compression is pretty fast.
- Requires 64 KB of memory for compression.
- Allows you to dial up extra compression at a speed cost in the Compressor.
- The speed of the decompressor is not reduced.
- Includes compression levels for generating pre-compressed data which achieve a quite competitive compression ratio.
- There is also a compression level which needs only 8 KB for Compression.
- Algorithm is thread safe.
- Algorithm is lossless.

LZO supports overlapping compression and in-place decompression.

c) Files required for LZO algorithm.

- Include files, source files (src) provided by LZO
- LZO.lib

For more information on LZO library and downloads visit:
http://www.nseindia.com/content/press/prs_whatsnew.htm#2

Specifications for utilizing on-line broadcast information

- **Decompression steps**:

a) Receive the packet in the temporary buffer i.e. array of characters.
b) First field will identify whether the packet is compressed or not.
c) If this field is **0** then Decompress it using LZO algorithm else if **1 don't** decompress it and proceed in normal way as it is being done today.
d) The second field is packet length.
e) The third field contains the number of records in the packet.
f) If compressed use following function of LZO to Decompress.

r = lzo1z_decompress ((unsigned char *) cInputBuf, ipLength, (unsigned char*) cOutputBuf, opLength, NULL);

lzo1z_decompress: Function which decompresses the data packet receive

CInputBuf: Input buffer in which compressed data is received

IpLength: The length of the packet which application has received using Receive ().

COutputBuf: The uncompressed output data which is result of decompression.

OpLength: Length of uncompressed data

g) After decompression data will be available in Output Buffer.
h) Map the outputbuf to existing Header structure according to **FN** or **FI.**
i) Look for Record size in the length field and Code (i.e. FN or FI).
j) Steps to recover data from OutputBuf.

Algorithm:

```
Length_of_Record = Header->length;
Sequence_no = Header->Sequence_num;
For I = 0 to Number of records (obtained in step 4)
Begin
  Bytes_to_seek = Length_of_Record * I
  Seek to number of Bytes_to_seek
  Map (Length_of_Record) of bytes to proper structure according
  to iCode (FN or FI) as found in Header part.
  Do the required processing….
   ….

  End
End for Loop.
```

3. DATA DETAILS

A single packet contains Header, the ASCII block of data and trailer. The interpretation of the ASCII data depends on the code field.

3.1 THE HEADER

The header consists of

3.1.1. **Code** – 2 char code field indicating whether the block of data is a Login Request **(DQ)**, Logon Response **(DR)**, Market Open (**DO**), Market Close (**DC**), Market Update (**DN**),Market Message **(DB)**, Open Interest Information (**DI**), EOD Contract Update Addition (**DA**), EOD Contract Update Modification (**DM**), EOD Contract Deletion **(DD)**, EOD Market Information (**DS**), EOD Data End Information (**DE**), Heartbeat Information (**DH**), Spread Order Information **(DP)** and Contract Information **(DT)**

3.1.2. **Length** – 2- byte length field (Hex) indicating the length of the packet that is being sent (including header, trailer and the carriage return).

3.1.3. **Sequence Number** – 4- byte field indicating the sequence number (Hex) of the ASCII text being sent.

3.2 THE DATA

The following information is provided in data block -

a) Market Open
b) Market Close
c) Market Update Information
d) Market Message Information
e) Open Interest Information
f) EOD Contract Update Information
g) EOD Market Information
h) EOD Data End Information
i) Heartbeat Information
j) Spread Order Information
k) Contract Information

3.3 TRAILER

The trailer is a two-byte checksum. A CR will terminate the block of data.

4. DATA STRUCTURE DETAILS

4.1 LOGIN REQUEST (send by client application)

Login Request packet will be send by the client application for login into the Infofeed application. If user wants to change his password he will specify the new password and confirm password field. Password is case sensitive. Format of this packet is as follows.

4.1.1 INFO HEADER

a) Code	'**DQ**'
b) Length	Short Integer (2 Bytes)
c) Sequence Number	00

4.1.2 ASCII DATA

a) User Id	10 Chars
b) Password	8 Chars
c) New Password	8 Chars
d) Confirm Password	8 Chars

4.1.3 INFO TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.2 LOGON RESPONSE (send by Infofeed application)

Logon response packet will be send by the Infofeed server application after receiving the Login Request packet from the client application.

4.2.1 COM HEADER

a) Compressed or Not	1 Char
b) Packet Length	Short Integer (2 Bytes)
c) No of Packets	1

4.2.2 INFO HEADER

a) Code	'**DR**'
b) Length	Short Integer (2 Bytes)
c) Sequence Number	00

4.2.3 ASCII DATA

a) Error Code	Long (4 Bytes)
b) Message	50 Chars

4.2.4 INFO TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.3 MARKET OPEN

The Market open packet is sent when the Currency Derivatives market opens for the day. The message is sent approximately at **9:00 am** on all trading days.

HEADER

a) Code	'**DO**'
b) Length	0x0C
c) Sequence Number	XXXX

DATA

FIELD TYPE	**FIELD WIDTH**
a) Market Type	char 1 Byte ('N'-Normal)

TRAILER

Checksum is not computed.

4.4 MARKET CLOSE

This message is disseminated at the end of the trading hours at approximately **5:00 pm** on each trading day.

HEADER

a) Code	'**DC**'
b) Length	0x0C
c) Sequence Number	XXXX

DATA

FIELD TYPE	**FIELD WIDTH**
a) Market Type	char 1 Byte ('N'-Normal)

TRAILER

Checksum is not computed.

4.5 MARKET UPDATE INFORMATION

These packets are sent during the market hours i.e. between **9:00 am** to **5:00 pm** and also in the market pre open period viz. before the actual market open. It contains the latest order and trade information of Currency Derivatives contracts.

HEADER

a) Code	'**DN**'
b) Length	0xF9
c) Sequence Number	XXXX

DATA

FIELD TYPE	**FIELD WIDTH**
a) Instrument	Char 6 Bytes
b) Symbol	Char 10 Bytes
c) Expiry date	Char 11 Bytes
d) Strike Price	Char 10 Bytes
e) Option Type	Char 2 Bytes
f) Market Type	Char 1 Byte
g) Best Buy-Order Price	Char 17 Bytes
h) Best Buy-Order Quantity	Char 12 Bytes (No. of contracts)
i) Best Sell-Order Price	Char 17 Bytes
j) Best Sell-Order Quantity	Char 12 Bytes (No. of contracts)
k) Last Traded Price	Char 17 Bytes
l) Total Traded Quantity	Char 12 Bytes (No. of contracts)
m) Security Status	Char 1 Byte (If Suspended "S" or Blank)
n) Opening Price	Char 17 Bytes
o) High Price	Char 17 Bytes
p) Low Price	Char 17 Bytes
q) Close Price	Char 17 Bytes
r) Average Traded Price	Char 17 Bytes
s) Total Turnover	Char 25 Bytes (Precision upto 2 decimal places)

TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.6 MARKET MESSAGE INFORMATION

This packet contains the messages broadcast in the market relating to corporate news or some important announcements in NSEIL's Currency Derivatives market segment. They are sent during the market time and message length is variable depending on their content.

HEADER

a) Code	'**DB**'
b) Length	0xXX (Varies as per message length)
c) Sequence Number	XXXX

DATA

FIELD TYPE	**FIELD WIDTH**
a) Message Code	Char 3 Bytes (NSE)
b) Message Length	Char 3 Bytes
c) Message String	Char varies

(Variable as per message length. Maximum Length is 240 bytes)

TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.7 OPEN INTEREST INFORMATION

This packet is sent during the trading hours and it indicates the Open Interest of the various contracts traded.

HEADER

a) Code	'**DI**'
b) Length	0x3D
c) Sequence Number	XXXX

DATA

FIELD TYPE	**FIELD WIDTH**
a) Instrument	Char 6 Bytes
b) Symbol	Char 10 Bytes
c) Expiry Date	Char 11 Bytes
d) Strike Price	Char 10 Bytes
e) Option Type	Char 2 Bytes
f) Open Interest	Char 10 Bytes (No. of contracts)
g) Market Type	Char 1 Byte

TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.8 EOD CONTRACT UPDATE INFORMATION

These packets are sent as the End of the Day feed on each trading day by approximately **06:00 PM** and this feed contains the information about the new contracts added to the F&O Market for trading.

HEADER

a) Code	'**DA**' or '**DM**' or '**DD**'
b) Length	0x7E
c) Sequence Number	XXXX

DATA

FIELD TYPE	**FIELD WIDTH**
a) Instrument	Char 6 Bytes
b) Symbol	Char 10 Bytes
c) Expiry Date	Char 11 Bytes
d) Strike Price	Char 10 Bytes
e) Option Type	Char 2 Bytes
f) Contract Name	Char 30 Bytes
g) Regular Lot	Char 5 Bytes
h) Market Type	Char 1 Byte ('N'-Normal)
i) Tick Size	Char 9 Bytes (Maximum precision upto 7 decimal places)
j) Maturity Date	Char 11 Bytes (Format: DD-MON-YYYY)
k) Last Update Date & Time	Char 20 Bytes (Format: DD-MON-YYYY HH: MM: SS)

TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.9 EOD MARKET INFORMATION

These packets contain the End of the Day values of the contracts traded during the Trading day. This information is disseminated on every Trading day by **06:00 PM**

HEADER

a) Code	'**DS**'
b) Length	0xE3
c) Sequence Number	XXXX

DATA

FIELD TYPE	FIELD WIDTH
a) Instrument	Char 6 Bytes
b) Symbol	Char 10 Bytes
c) Expiry Date	Char 11 Bytes
d) Strike Price	Char 10 Bytes
e) Option Type	Char 2 Bytes
f) Market Type	Char 1 Byte ('N'-Normal)
g) Opening Price	Char 17 Bytes
h) Trade High Price	Char 17 Bytes
i) Trade Low Price	Char 17 Bytes
j) Closing Price	Char 17 Bytes
k) Last Traded Price	Char 17 Bytes
l) Previous Close Price	Char 17 Bytes
m) Settlement Price	Char 17 Bytes
n) Total Traded Quantity	Char 12 Bytes (No. of contracts)
o) Total Traded Value	Char 25 Bytes (Precision up to 2 decimal places)
p) Open Interest	Char 10 Bytes (No. of contracts)
q) Change in Open Interest	Char 10 Bytes (No. of contracts)

TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.10 EOD DATA END INFORMATION

This packet is sent after the EOD MARKET INFORMATION feed and the EOD CONTRACT UPDATE INFORMATION feed is successfully sent to the Vendors. This packet intimates the End of EOD data.

HEADER

a) Code	'**DE**'
b) Length	0x0B
c) Sequence Number	XXXX

DATA

No data is associated with data packet.

TRAILER
Checksum is not computed.

4.11 HEARTBEAT INFORMATION

The Heartbeat packets are sent throughout the day. This packet indicates to the Info-Vendors that data packets are received from the Infofeed server (Implies that connection to Infofeed server is alive).

HEADER:

a) Code	'**DH**'
b) Length	0x0B
c) Sequence Number	XXXX

ASCII DATA:
Not associated with any ASCII data.

TRAILER:
Checksum is not computed.

4.12 SPREAD ORDER UPDATE INFORMATION

These packets are sent during the market hours i.e. between **9:00 am** to **5:00 pm** and also in the market pre open period viz. before the actual market open. It contains the latest order and trade information of Currency Derivatives spread contracts.

HEADER

a) Code	'**DP**'
b) Length	0xE3
c) Sequence Number	XXXX

DATA

FIELD TYPE	FIELD WIDTH
a) Instrument_1	Char 6 Bytes
b) Symbol_1	Char 10 Bytes
c) Expiry date_1	Char 11 Bytes
d) Strike Price_1	Char 10 Bytes
e) Option Type_1	Char 2 Bytes
f) Instrument_2	Char 6 Bytes
g) Symbol_2	Char 10 Bytes
h) Expiry date_2	Char 11 Bytes
i) Strike Price_2	Char 10 Bytes
j) Option Type_2	Char 2 Bytes
k) Best Buy-Order Price	Char 17 Bytes
l) Best Buy-Order Quantity	Char 12 Bytes (No. of contracts)
m) Best Sell-Order Price	Char 17 Bytes
n) Best Sell-Order Quantity	Char 12 Bytes (No. of contracts)
o) Last Traded Price Difference	Char 17 Bytes
p) Total Traded Quantity	Char 12 Bytes (No. of contracts)

q)	Opening Price Difference	Char 17 Bytes
r)	Day High Price Difference	Char 17 Bytes
s)	Day Low Price Difference	Char 17 Bytes

TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.13 CONTRACT INFORMATION

These packets are sent at the beginning of the each trading day by approximately **08:15 AM.** This feed contains the information about the contracts valid in the Currency Derivatives Market for trading.

HEADER

a)	Code	'**DT**'
b)	Length	0x3D
c)	Sequence Number	XXXX

DATA

FIELD TYPE		**FIELD WIDTH**
a)	Token Number	Char 10 Bytes
b)	Instrument	Char 6 Bytes
c)	Symbol	Char 10 Bytes
d)	Expiry Date	Char 11 Bytes
e)	Strike Price	Char 10 Bytes
f)	Option Type	Char 2 Bytes
g)	Delete Flag	Char 1 Bytes

TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

5. CONTACT INFO

Name	Email	Contact Number
DOTEX Business	**dotex@nse.co.in**	91-22-26598385
Technical Support	**infofeed_support@nse.co.in** .	-

6. NOTE

- In the Pre-Open phase of the market, the Order Price can be 0.00. This indicates that the order placed is an At Open Order (ATO) and the actual Order Price will be calculated by the system when the market opens. Even though no trade takes place in the Pre-Open phase the Last Traded Price and the Open Price will change as the orders are placed in the system. The final open price for the security is calculated after Pre-Open period has ended. The Total Traded Quantity will be zero in the Pre-Open phase. In the Market Open phase also the Total Traded Quantity will be zero when orders are placed but no trades take place. The Last Traded Price will never be zero in the Normal Market.

- Prices field will have the precision up to 7^{th} decimal places.

7. CHECKSUM ALGORITHM

The **Checksum** routine followed for Info Vendor Feed is as follows:

```
// Following are the defines for checksum calculation
#define DC1           17
#define DC3           19
#define CR            13
#define LF            10
#define POLY    0x1021
// End of defines

unsigned check_sum (cData, iLength)
char *cData;
int iLength;
{
unsigned uAccum = 0;
unsigned uData;
unsigned char ucChk [2];
int i, j;

        for (i=0;i<iLength;i++){
                        uData = *(cData+i);
                        uData <<= 8;
                        for (j=8; j>0; j--){
                                if ((uData^uAccum)&0x8000)
                                        uAccum=(uAccum<<1)^POLY;
                                /* SHIFT AND SUBTRACT POLY */
                                else
                                        uAccum<<=1;
                                uData<<=1;
                        }
        }

        ucChk[0] = uAccum>>8;
        if (ucChk [0] == DC1 || ucChk [0] == DC3 || ucChk [0] == CR ||
ucChk[0] == LF)
                        UcChk [0] -= 1;
        UcChk [1] = uAccum&0xFF;
        if (ucChk [1] == DC1 || ucChk [1] == DC3 || ucChk[1] == CR || ucChk
[1] == LF )
                        UcChk [1] -= 1;
        uAccum = ucChk [1];
        uAccum = (uAccum<<8) + ucChk[0];

        return(uAccum);
}
```

8. EXAMPLE: FUNCTION FOR DECOMPRESSION

```
lzo_decomp (char cInputBuf [], unsigned int ipLength, char cOutputBuf [],
unsigned *opLength, unsigned short * lzo_errorcode)
{
        int r;
        Char mess [50];
        r = lzo1z_decompress ((unsigned char *) cInputBuf, ipLength,
                                (unsigned char *) cOutputBuf, opLength,
                                 NULL);

        If ( r == LZO_E_OK)
        {
         Print (mess," Decompressed %lu bytes back into %lu bytes\n",
                                    (long) ipLength, (long) *opLength);

         Return true;
        }

        Else
        {
        OutputDebug ("Internal error - decompression failed");
        Return false;
        }
}
```