rexroth
A Bosch Company

06/08/2021

Protocol Description For Zoomlion

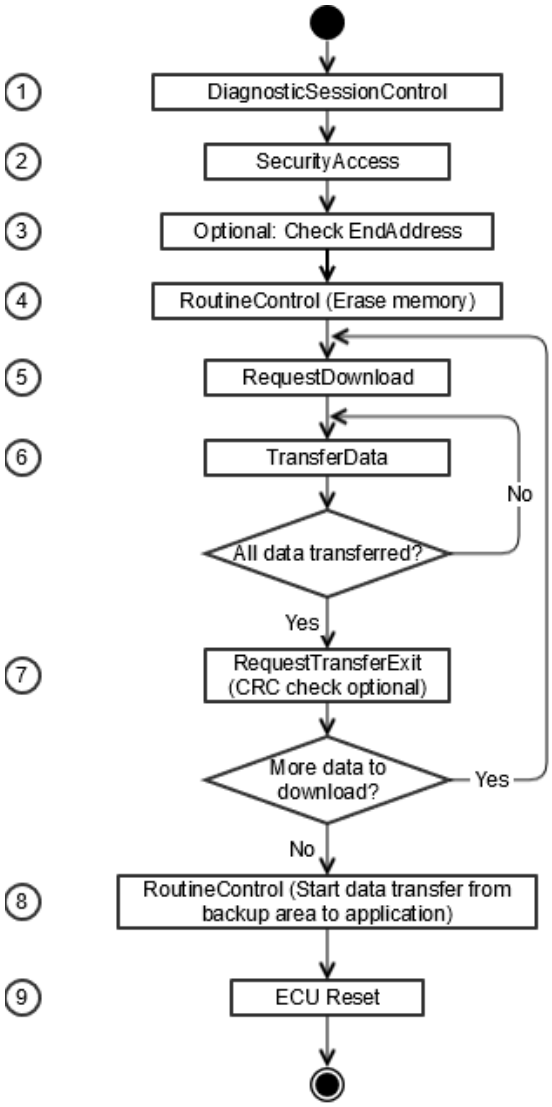
1 修改历史

v0.1	Unknown User	Initial	03 Apr 2020
v0.2	Unknown User	Update flashing sequence diagram	10 Jul 2020
v0.3	Unknown User	Update to standard UDS protocol	22 Jul 2020
v0.4	Unknown User	Update routine control identifier for backup mechanism	17 Aug 2020
v1.0	Jing Shuman (DC-MH/ENG52-CN)	Added detailed CAN messages	09 Sep 2020
v1.1	Jing Shuman (DC-MH/ENG52-CN)	Corrected CAN ID	14 Sep 2020
v2.0	Unknown User	Added error handling	14 Oct 2020
v3.0	Unknown User	Added protocol breakpoint transmission Added protocol controller specifier	26 Jan 2021
v3.1	JING Shuman (DC-MH/EMF-CN)	Added $31 CheckEndAddress	08 Jun 2021

2 CAN ID

	ID
T-box → RC	0x18DA3218
RC → T-box	0x18DA1832

3 刷写流程



步骤	UDS服务名称	T-box发送信息														RC返回信息					
		0	1	2	3	4	5	6	7	8	9	10	11	12	...	0	1	2	3	4	5
1	会话管理	10	02													50	02				
2	安全访问:请求种子	27	07													67	07	种子			

	安全访问:发送密钥	27	08	密钥										67	08					
3	程序控制:检查程序结束地址	31	01	FF	02	04	结束地址							71	01	FF	02	00	(刷写备份区)	
														71	01	FF	02	01	(不刷写备份区)	
4	程序控制:擦除存储器	31	01	FF	00	44	起始地址	数据长度						7F	31	78	(擦除中)			
														71	01	FF	00	(擦除完毕)		
5	请求下载	34	00	44	存储器地址			存储器长度						74	20	数据块最大长度				
6	传输数据	36	块序号	待传输数据 (每次传输最长长度为1024字节)										76	块序号					
7	请求退出传输	37	CRC模式(00/01)	CRC校验值(当模式为01)										7F	37	78	(传输中)			
														77	CRC模式(00/01)	00	(传输完毕)			
8	程序控制:拷贝备份	31	01	FF	01									7F	31	78	(拷贝中)			
														71	01	FF	01	(拷贝完毕)		

9	重启ECU	11	01													无					

注：使用T-box进行刷写时，无需用到显示屏。

> **注意**
>
> 当结束地址超出应用软件区域，控制器将不刷写到备份区，也就无需步骤8拷贝备份。此时若刷写失败，控制器中无有效的应用软件，直到正确刷写之后应用软件才能正常运行。
>
> 如果不执行步骤3，控制器将会刷写到备份区。

3.1 其他服务

UDS服务名称	**T-box发送信息**														**RC返回信息**							**备注**
	0	1	2	3	4	5	6	7	8	9	10	11	12	...	0	1	2	3	4	5	6	
根据标识符读数据：读取控制器SN号	22	F2	01												62	F2	01	SN号				用于终端与控制器唯一标识符绑定
根据标识符读数据：读取有效程序尾地址	22	F2	02												62	F2	02	有效程序尾端地址				用于断点续传

3.2 断点续传

相较于标准刷写流程断点续传需要在程序擦写下载前向控制器询问当前程序有效尾地址，从有效末地址开始后续刷写流程。如果未询问有效尾地址，则控制器默认从起始进行刷写。即：在步骤2之后通过根据标识符读数据读取程序有效尾地址。

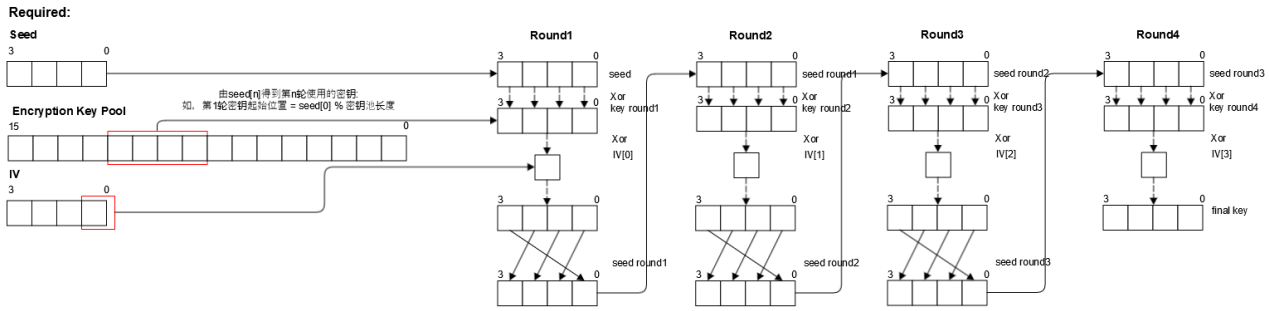
> 为减少内存过度擦写消耗，有效地址存储只发生在TransferExit之后。

> 如无刷数据传输发生或者整个软件下载拷贝完毕，则询问返回为0。

4 安全访问

4.1 安全访问算法

$27安全访问服务使用的算法如下，示例程序见附录8.2



示例：

```
==========================================================================
==========================START ENCRYPTION==============================
==========================================================================
seed = 0F B6 F7 47

Key pool: { 1B, 66, 79, 88, 45, 6D, 31, D9,
            21, 22, 5D, 7A, C1, FB, B3, 1A,  }

IV: { E5, ED, 89, C1 }

Key pool start index = seed[0] % key_pool_size = 0F % 10 = 0F
seed_round_0 = 0F | key_pool_index = 0F | key_pool_value = 1A | Xor -> 15 | iv = E5 | round_1 = F0
seed_round_0 = B6 | key_pool_index = 00 | key_pool_value = 1B | Xor -> AD | iv = E5 | round_1 = 48
seed_round_0 = F7 | key_pool_index = 01 | key_pool_value = 66 | Xor -> 91 | iv = E5 | round_1 = 74
seed_round_0 = 47 | key_pool_index = 02 | key_pool_value = 79 | Xor -> 3E | iv = E5 | round_1 = DB

Key pool start index = seed[1] % key_pool_size = B6 % 10 = 06
seed_round_1 = DB | key_pool_index = 06 | key_pool_value = 31 | Xor -> EA | iv = ED | round_2 = 07
seed_round_1 = F0 | key_pool_index = 07 | key_pool_value = D9 | Xor -> 29 | iv = ED | round_2 = C4
seed_round_1 = 48 | key_pool_index = 08 | key_pool_value = 21 | Xor -> 69 | iv = ED | round_2 = 84
seed_round_1 = 74 | key_pool_index = 09 | key_pool_value = 22 | Xor -> 56 | iv = ED | round_2 = BB

Key pool start index = seed[2] % key_pool_size = F7 % 10 = 07
seed_round_2 = BB | key_pool_index = 07 | key_pool_value = D9 | Xor -> 62 | iv = 89 | round_3 = EB
seed_round_2 = 07 | key_pool_index = 08 | key_pool_value = 21 | Xor -> 26 | iv = 89 | round_3 = AF
seed_round_2 = C4 | key_pool_index = 09 | key_pool_value = 22 | Xor -> E6 | iv = 89 | round_3 = 6F
seed_round_2 = 84 | key_pool_index = 0A | key_pool_value = 5D | Xor -> D9 | iv = 89 | round_3 = 50

Key pool start index = seed[3] % key_pool_size = 47 % 10 = 07
seed_round_3 = 50 | key_pool_index = 07 | key_pool_value = D9 | Xor -> 89 | iv = C1 | round_4 = 48
seed_round_3 = EB | key_pool_index = 08 | key_pool_value = 21 | Xor -> CA | iv = C1 | round_4 = 0B
seed_round_3 = AF | key_pool_index = 09 | key_pool_value = 22 | Xor -> 8D | iv = C1 | round_4 = 4C
seed_round_3 = 6F | key_pool_index = 0A | key_pool_value = 5D | Xor -> 32 | iv = C1 | round_4 = F3

Final key = 48 0B 4C F3
```

4.2 密钥及初始向量

Encryption key pool 密钥池：

```
sec_encKey_au8[16] = {
    0xb0, 0xad, 0x2a, 0x70, 0x5c, 0xd1, 0x33, 0xc6,
    0x3a, 0x4f, 0xf7, 0x89, 0xb6, 0x28, 0x6f, 0x0d
}
```

Initialization vector (IV) 初始向量：

```
sec_IV_au8[4] = { 0xf6, 0x7b, 0x17, 0x4d }
```

5 CRC校验算法

在$37请求退出传输服务这一步里，RC会计算该数据块的校验值，使用到CRC16 (CCITT)算法，涉及到的参数如下：

CRC Width CRC结果宽度	16 bits
Truncated Polynomial 多项式	0x1021
Initial Value 初始值	0xFFFF
Input Data Reflection 输入数据反射	No
Output CRC Reflection 结果数据反射	No
XOR Value 异或值	0x0000
Check Value 校验	0x29B1

CRC16计算示例程序见附录8.1。

6 Hex文件解析

6.1 Hex文件格式

Hex文件以行为单位，每行以冒号开头，内容全部为16进制码（以ASCII码形式显示）。Hex文件可以按照如下的方式进行拆分来分析其中的内容：

例如下图第一行 :02000004800278 , 把它看做 0x02 0x00 0x00 0x04 0x80 0x02 0x78

第一个 0x02 为该行数据长度；

紧跟着后面的0x00 0x00 为地址；

再后面的0x04为数据类型，类型共分以下几类：

'00' 数据记录
'01' 文件结束记录
'02' 扩展段地址记录
'03' 开始段地址记录
'04' 扩展线性地址记录
'05' 开始线性地址记录

然后，接着0x04后面的两个 0x80 0x02就是数据。最后一个0x78是校验码。

```
1  :02000004800278
2  :100000000000000A0000000080 76D40A004A16809C
3  :10001000000000000000000000000432D4150492076
4  :10002000202020200000000000000000010000004F
5  :10003000000000000FFFFFFFFFFFFFFFFFFFFFFFCC
```

6.2 RC应用程序文件格式

第一行为存储器起始地址，应用软件的起始地址为0x80020000。

余下各行为数据记录。如上图第4行：

0020: 地址偏移量，既该行数据地址为0x80020000 + 0x0020 = 0x80020020
2020**20**20**00**00**00**00**00**00**00**01**00**00**00：待传输数据

$36传输数据服务每次最多可发送1024字节数据，Hex文件每行数据长度为16，也就是说可将64行的地址连续的数据拼接起来发送。

7 诊断服务

7.1 支持的UDS服务

SID	Name	Meaning
0x10	DiagnosticSessionControl	Managing the diagnosis sessions
0x11	ECUReset	Release of a software reset
0x22	ReadDataByIdentifier	Load information from ECU
0x27	SecurityAccess	Verification of login to the unit
0x31	RoutineControl	Controlling of ECU function remotely
0x34	RequestDownload	Preparation of a data download
0x36	TransferData	Transfer of data packages
0x37	RequestTransferExit	Exit the transfer of data packages

7.2 错误代码 (DTC)

DTC	Name	Meaning
0x10	General Reject	Unit does not process the request
0x11	Service Not Supported	SID is unknown
0x12	Subfunction Not Supported	LID is unknown
0x13	Incorrect Message Length	Length of the message is incorrect or format is unknown
0x21	Busy Repeat Request	Unit cannot complete the service at the moment
0x22	Conditions Not Correct	Server (RC) prerequisite conditions are not met
0x24	Request Sequence Error	Server expects a different sequence of message
0x31	Request Out Of Range	Server detects that message contains parameter out of range limit
0x33	Security Access Denied	Server's security strategy has not been satisfied by client
0x35	Invalid Key	Security access not given by server because key sent by client doesn't match
0x71	Transfer Data Suspended	Data transfer is halted due to transfer fault
0x72	General Programming Failure	Server detects an error when erasing or programming a memory location
0x73	Wrong Block Sequence Counter	Server detects an error in the sequence of counter values

DTC	Name	Meaning
0x78	Request Correctly received-Response Pending	Request message was received correctly. Action to be performed is not yet completed
0x7F	Service Not Supported In Active Session	The service is not supported in the currently opened session, however it can be operated in other sessions.

7.3 DiagnosticSessionControl (0x10)

Request:

SID: 0x10 (DiagnosticSessionControl)

LID: Refer to the following table

LID	Name	Meaning
0x01	defaultSession	This session is always open, no data access is allowed.
0x02	programmingSession	This session supports the memory programming of server.

Positive Response:

SID: 0x50 (DiagnosticSessionControlPositiveResponse)

LID: 0x01 | 0x02 (according to the request)

Negative Response:

SID: 0x7F (NegativeResponse)

LID: 0x10 (Error on: DiagnosticSessionControl)

DTC: General

7.4 ECUReset (0x11)

Request:

SID: 0x11 (ECUReset)

LID	Name	Meaning
0x01	HardReset	The unit should stimulate a PowerOn Reset.

Positive Response:

SID: 0x51 (ECUResetPositiveResponse)

LID: Reset type (mirror value of the request)

Negative Response:

SID: 0x7F (NegativeResponse)

LID: 0x11 (Error on: ECUReset)

DTC: Refer to the below table

DTC	Name	Meaning
0x13	Incorrect Message Length	Length of the message is incorrect
0x22	Conditions Not Correct	The prerequisites for this service are not fulfilled
0x25	Flash Process Error	Flash process should end before reset

7.5 ReadDataByIdentifier (0x22)

Request:

Format:

0x22 (SID)	LID1	LID2	...

LID: Refer to the below table

LID	Name	Meaning
0xF201	Read SN	Request controller SN number
0xF202	Read Valid End Address	Request end address for breakpoint transmission

Positive Response:

SID: 0x62 (ReadDataByIdentifierPositiveResponse)

LID: RequestLID

Data: Feedback data

Negative Response:

SID: 0x7F (NegativeResponse)

LID: 0x27 (Error on: SecurityAccess)

DTC: Refer to the below table

DTC	Name	Meaning
0x13	Incorrect Message Length	Length of the message is incorrect
0x14	Response Too Long	Response exceeds the maximum number of bytes available
0x31	Request Out Of Range	Request out ff Range
0x33	Security Access Denied	Security access level not reached

7.6 SecurityAccess (0x27)

7.6.1 RequestSeed

Request:

SID: 0x27 (SecurityAccess)

LID: Refer to the below table

LID	Name	Meaning
0x07	RequestSeed	Request a random number for diagnositic session 'Flash'

PositiveResponse for RequestSeed:

SID: 0x67 (SecurityAccessPositiveResponse)

LID: RequestLID

Data: Random number of ECU (128-Bit)

NegativeResponse for RequestSeed:

SID: 0x7F

LID: 0x27 (Error on: SecurityAccess)

DTC: 0x22 ConditionsNotCorrect

7.6.2 SendKey

The key (128 bit value) is sent by the tester to ECU for comparison.

Request:

SID: 0x27 (SecurityAccess)

LID: Refer to the below

LID	Name	Meaning
0x08	SendKey	Send key for access level 'Flash'

Key: 128 bit AES result

Positive Response for SendKey:

SID: 0x67 (SecurityAccessPositiveResponse)

LID: RequestLID

Data: 0x34 (SecurityAccessAllowed)

Negative Response:

SID: 0x7F (NegativeResponse)

LID: 0x27 (Error on: SecurityAccess)

DTC: Refer to the below table

DTC	Name	Meaning
0x22	Conditions Not Correct	The prerequisites for this service are not fulfilled
0x24	Request Sequence Error	No RequestSeed before SendKey
0x35	Invalid Key	Key does not match with the internally calculated value

7.7 RoutineControl (0x31)

Request:

Format:

0x31 (SID)	LID	RID	OptionRecord

SID: 0x31

LID: Refer to the below table

LID	Name	Meaning
0x01	Start Routine	Start routine specified by RID

RID: Routine Identifier, refer to the table below

RID	Name	Meaning	Option Record Format	Example
0xFF00	Erase Block	Erase the assigned block	<AddressAndLengthFormatIdentifier (1 byte)> (see below), <MemoryAddress (4 bytes)>, <MemorySize (4 bytes)>	31 01 FF 00 44 80 02 00 00 00 00 80 00
0xFF01	Transfer Data From Application Area To Backup	Flashing application area after finishing all data transfer		31 01 FF 01
0xFF02	CheckEndAddress	Check whether program end address reaches backup area. Data will be flashed to application area directly if end address exceeds 0x80200000.	<AddressFormatIdentifier (1 byte)>, <EndAddress (4 bytes)>	31 01 FF 02 04 80 10 FF FF

Address and length format identifier:

Bit	Value	Description
7-4	4	4 Bytes for memory length
3-0	4	4 Bytes for memory address

Positive Response:

SID: 0x71 (SecurityAccessPositiveResponse)

LID: RequestLID, <Result (optional)>

Negative Response:

SID: 0x7F (NegativeResponse)

LID: 0x31 (Error on: RoutineControl)

DTC: Refer to the below table

DTC	Name	Meaning
0x12	Subfunction Not Supported	LID is unknown
0x13	Incorrect Message Length	Length of the message is incorrect
0x22	Conditions Not Correct	The prerequisites for this service are not fulfilled

DTC	Name	Meaning
0x24	Request Sequence Error	Request sequence error
0x25	Flash Process Error	Flash process should end before reset
0x31	Request Out Range	Request out of range
0x33	Security Access Denied	Security access level not reached
0x78	Request Correctly received-Response Pending	Request message was received correctly. Action to be performed is not yet completed

7.8 RequestDownload (0x34)

The RequestDownload service is used by the client to initiate a data transfer from the client to the server(download).

Request:

0x34 (SID)	DataFormatIdentifier (1 Byte)	AddressAndLengthFormatIdentifier (1 Byte)	MemoryAddress (4 Bytes)	MemorySize (4 Bytes)

Data format identifier:

Value	Description
0x00	No encryption required

Address and length format identifier:

Bit	Value	Description
7-4	4	4 Bytes for memory length
3-0	4	4 Bytes for memory address

Response format:

0x34 (SID)	LengthFormatIdentifier (1 Byte)	MaxNumberOfBlockLength (2 Byte)

Length format identifier:

Bit	Value	Description
7-4	2	2 Bytes for max block length
3-0	0	0 (Reserved)

Positive Response:

0x74	0x20	0x04	0x00

Negative Response:

0x7F	0x34	DTC

DTC:

DTC	Name	Meaning
0x13	Incorrect Message Length	Length of the message is incorrect
0x22	Conditions Not Correct	The prerequisites for this service are not fulfilled
0x31	Request Out Range	Request out of range
0x33	Security Access Denied	Security access level not reached
0x72	General Programming Failure	Error when programming a memory location
0x7F	Service Not Supported In Active Session	The service is not supported in the currently opened session

7.9 TransferData (0x36)

The TransferData service is used by the client to transfer data to the server

Request:

0x36 (SID)	BlockSequenceCounter (1 Byte)	Data (max length refer to MaxNumberOfBlockLength from RequestDownload(0x34) response)

Positive Response:

0x76	BlockSequenceCounter (1 Byte)

请求下载服务($34)之后第一个传输数据请求报文中的块序号参数的值为0x01，之后每次递增1。当值达到0xFF后，该参数归0 重新开始递增。

Negative Response:

0x7F	0x36	DTC

DTC:

DTC	Name	Meaning
0x13	Incorrect Message Length	Length of the message is incorrect
0x22	Conditions Not Correct	The prerequisites for this service are not fulfilled
0x24	Request Sequence Error	Request sequence error
0x31	Request Out Range	Request out of range
0x71	Transfer Data Suspended	Data transfer is halted due to transfer fault

DTC	Name	Meaning
0x72	General Programming Failure	Error when programming a memory location
0x73	Wrong Block Sequence Counter	Error in the sequence of counter values
0x7F	Service Not Supported In Active Session	The service is not supported in the currently opened session

7.10 RequestTransferExit (0x37)

Request:

0x37 (SID)	CRCMode (1 Byte)	BlockCRCValue(2 Bytes)

CRC mode:

Value	Description
0x00	No CRC check required
0x01	CRC check required

Positive Response:

0x77	CRCMode (1 Byte)	CRC check result (1 Byte)(if CRC mode)

CRC Check Result

Value	Description
0x00	CRC check passed

Negative Response:

0x7F	0x37	DTC

DTC:

DTC	Name	Meaning
0x13	Incorrect Message Length	Length of the message is incorrect
0x21	Busy Repeat Request	Unit cannot complete the service at the moment
0x22	Conditions Not Correct	The prerequisites for this service are not fulfilled
0x24	Request Sequence Error	Request sequence error
0x77	Invalid CRC value	CRC check failed
0x78	Request Correctly received-Response Pending	Request message was received correctly. Action to be performed is not yet completed

DTC	Name	Meaning
0x7F	Service Not Supported In Active Session	The service is not supported in the currently opened session

8 异常处理

针对于可能出现的刷写流程中异常情况（尤其是在总线负载率较高的情况下），例如丢帧或其他传输层错误，这时RC控制器会将错误代码发送到总线上，TBOX应基于RC控制器给定错误代码进行异常处理。具体示例详见附录9.3.

错误信息CAN ID：0x18DA5C5C

错误信息帧定义

0x02	0x0A	错误代码

错误代码定义

错误代码	内容	原因	异常处理机制
0x80	流控帧超时	客户端未在定义时限内（1000ms）发送流控帧	N.A.
0x81	流控帧超时	客户端未在定义时限内（1000ms）发送连续帧	客户端应重发当前包并保证连续帧在限定时间内
0x82	流控帧溢出	客户端发送流控帧溢出 (0x32)	N.A.
0x83	流控帧保留	客户端发送流控帧保留 (0x33 - 0x3F)	N.A.
0x84	连续帧丢失	连续帧丢失（连续帧序号不连续）	客户端在收到错误信息后立即重发当前包
0x85	错误帧类型	在发送连续帧的条件下，服务端收到其他帧类型	客户端需要立即停止当前包发送并回滚到前一包
0x92	单帧长度错误	单帧长度小于0或大于7	客户端应根据实际长度适配单帧多帧原则
0x93	多帧首帧长度错误	多帧首帧长度小于8	客户端应根据实际长度适配单帧多帧原则
0x94	多帧首帧溢出	多帧首帧长度大于给定限定长度	客户端应对当前包进行拆分以保证包长在限定范围内

注：表中客户端即TBOX，服务端即控制器

建议：为防止在刷写过程中出现不可预料的中断，建议TBOX在总线传输非活跃状态下5s后对控制器进行重启，此次刷写流程判定为失败。

9 附录

9.1 crc16计算程序示例

```
typedef unsigned char uint8;
typedef unsigned short uint16;
typedef unsigned int uint32;

uint16 crc16CCITT(uint8 *data_pu8, uint32 length_u32)
{
    uint16 crc_u16 = 0xFFFF;
    uint8 index_u8;
    uint16 dataOp_u16;

    do
    {
        for (index_u8 = 0, dataOp_u16 = ((uint32)0xFF & *data_pu8++) << 8; index_u8 <
8; index_u8++, dataOp_u16 <<= 1)
        {
            if ((crc_u16 & 0x8000) ^ (dataOp_u16 & 0x8000))
            {
                crc_u16 = (crc_u16 << 1) ^ 0x1021;
            }
            else
            {
                crc_u16 <<= 1;
            }
        }

    } while(--length_u32);

    return crc_u16 & 0xFFFF;
}
```

9.2 安全访问算法程序示例

```
int main() {

  unsigned char seed_au8[4] = { 0x0F, 0xB6, 0xF7, 0x47 };
  unsigned char key_pool_pau8[16] = { 0x1B, 0x66, 0x79, 0x88, 0x45, 0x6D, 0x31, 0xD9,
                                      0x21, 0x22, 0x5D, 0x7A, 0xC1, 0xFB, 0xB3, 0x1A};
  unsigned char index, indexj;
  unsigned char round1[4][4];
  unsigned char iv_au8[4] = {0xE5, 0xED, 0x89, 0xC1};

  memcpy(round1[0], seed_au8, sizeof(seed_au8));
  printf("seed = %02X %02X %02X %02X\n\n", round1[0][0], round1[0][1], round1[0][2],
round1[0][3]);
  for (indexj = 0; indexj < 4; indexj ++)
  {
    printf("Key pool start index = seed[%d] %% key_pool_size = %02X %% 10 = %02X\n",
indexj, seed_au8[indexj], seed_au8[indexj] % 16);
    for (index = 0; index < 4; index++)
    {
       printf("seed_round_%d = %02X ", indexj, round1[indexj][index]);
       printf("| key_pool_index = %02X ", (seed_au8[indexj] % 16 + index) % 16);
       printf("| key_pool_value = %02X ", key_pool_pau8[(seed_au8[indexj] % 16 +
index) % 16]);
       round1[indexj][index] ^= key_pool_pau8[(seed_au8[indexj] % 16 + index) % 16];
       printf("| Xor -> %02X ",round1[indexj][index]);
       printf("| iv = %02X ", iv_au8[indexj]);
       round1[indexj][index] ^= iv_au8[indexj];
       printf("| round_%d = %02X\n", indexj+1, round1[indexj][index]);
    }
    printf("\n");
    if (indexj < 3) {
       round1[indexj + 1][0] = round1[indexj][3];
       round1[indexj + 1][1] = round1[indexj][0];
       round1[indexj + 1][2] = round1[indexj][1];
       round1[indexj + 1][3] = round1[indexj][2];
    }
  }

  printf("Final key = %02X %02X %02X %02X\n", round1[3][0], round1[3][1], round1[3]
[2], round1[3][3]);
}
```

9.3 异常处理示例

9.3.1 连续帧丢失（0x84）

```
18DA3218x      Tx   d 8 14 02 36 06 99 32 7C 54
18DA1832x      Rx   d 3 30 81 00  Length = 36800
18DA3218x      Tx   d 8 21 60 D3 09 25 B4 18 0C
18DA3218x      Tx   d 8 23 30 02 F4 6D 00 16 1B
18DA5C5Cx      Rx   d 8 02 0A 84 FF FF FF FF FF
18DA3218x      Tx   d 8 14 02 36 06 99 32 7C 54
18DA1832x      Rx   d 3 30 81 00  Length = 36800
18DA3218x      Tx   d 8 21 60 D3 09 25 B4 18 0C
18DA3218x      Tx   连续帧丢失，收到错误信息后停 B
18DA3218x      Tx   止原包发送，立即进行连续帧重 06
18DA3218x      Tx   发 8 24 60 D9 F3 06 90 89 C2
18DA3218x      Tx   d 8 25 40 09 4C C1 59 F8 3C
18DA3218x      Tx   d 8 26 80 59 FF 00 90 8C C0
18DA3218x      Tx   d 8 27 AC 20 D9 F2 0A 90 8C
18DA3218x      Tx   d 8 28 C1 B4 29 AC 30 D9 F3
18DA3218x      Tx   d 8 29 08 90 0C 30 96 20 8F
18DA3218x      Tx   d 8 2A 3F C4 F1 2C 30 D9 F3
```

9.3.2 连续帧超时 （0x81）

```
175.429331 1   18DA3218x      Tx   d 8 14 02 36 06 99 32 7C 54
175.430183 1   18DA1832x      Rx   d 3 30 81 00  Length = 36800
175.580725 1   18DA5C5Cx      Rx   d 8 02 0A 81 FF FF FF FF FF
175.583482 1   18DA3218x      Tx   d 8 14 02 36 06 99 32 7C 54
175.584245 1   18DA1832x      Rx   d 3 30 81 00  Length = 36800
175.585326 1   18DA3218x      Tx   d 8 21 60 D3 09 25 B4 18 0C
175.585898 1   18D[illegible]D 00 16 1B
175.586474 1   18D[illegible]3 23 D9 F2 04 90 3B 40 06
175.587038 1   18DA3218x      Tx   d 8 24 60 D9 F3 06 90 89 C2
```

超时未检测到连续帧，收到错误信息
后重新进行原包发送