Discover What's Possible™ Anritsu

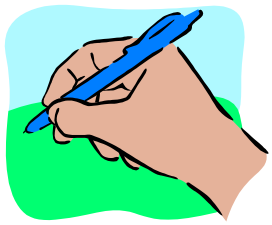
TECHNICAL NOTE

MD8470A
Signalling Tester

Scenario Modification

ANRITSU CORPORATION

Detail Explanation of Scenario Modification



Contents 1

Part 1: Inputting a value from a user

- General C language
- Scenario's layer3 messages are 'Char' type variable
- useful libraries
- global variable over a scenario

Part 2: Modifying wireless protocol messages

- RRC message
- NAS message (CC/MM/SM/GMM/SMS/SS)
- SIB message
- message modification trial

Contents 2

Part 3: Lower layer parameter modification
- lower layer parameter setting
- lower layer configure (executing)

Part 4: TE(terminal equipment) part
- Voice
- Video telephony (loopback)
- Packet
- Video telephony (ISDN)

Part 5: Protocol Sequence Modification

Part 1: Inputting a value from a user

General 1

1. If we need to change some parameters in the layer3 message to some fixed value, modifying the layer3 message is enough. (please refer the part2 of this.)
In some cases, user would like to change some parameters to the various kinds of values.

2. In above cases, we need to modify a scenario that has a 'User-input' interface and replace a layer3 message based on a 'User-input'.

3. This is general C program language area, so there are various kinds of way.

4. Usually I modify a scenario based on the following way.
   - A - Using 'SequenceStr' library (User can input any letters)
   - B - Input value conversion to the 'Integer' variable (for easier to operate, such as incrementing…)
   - C - Conversion from 'Integer' to 'MSB(layer3 message format)'
   - D - Replacing(overwriting) a layer3 message

General 2

B - Input value conversion to the 'Integer' variable

```
// example : using 'atol' library
// this example is for inputting 'Integer' value
INT rac;
CHAR str[256];
SequenceStr("RAC?", str, NO_TIMEOUT );
rac = atol(str);  // converting to INT variable

// another example : using 'ConvCharToArray' and 'Msb2IntIE' library
// this example for inputting 'Hex' value
INT lac;
CHAR str[256];
CHAR buff[256];
SequenceStr("LAC('0x0000' to '0xFFFF')", str, NO_TIMEOUT );
ConvCharToArray(str, buff, sizeof( buff ));        // allocate enough area for buff
LAC = Msb2IntIE(buff, 16);              // 16 is the bit-length of input value(LAC)
```

General 3

C - Conversion from ‘Integer’ to ‘MSB(layer3 message format)’

// example
Int2MsbIE(LAC, buff, 16);
// first parameter is ‘Integer’ type variable that is converted from
// second parameter is ‘CHAR’ array type variable that is converted to
// third parameter is the bit-length for converting

D - Replacing(overwriting) a layer3 message

// example
ReplaceIE(SndData, buff, 38, 16);
// first parameter is ‘CHAR’ array type variable, this is layer3 message
// second parameter is ‘CHAR’ array type variable that replace
// third parameter is ‘Start-bit-point’ for replacing
// fourth parameter is ‘bit-length’ for replacing

Example 1

```
UCHAR SndData[] = {
        0x00, 0x0e, 0x01, 0x74, 0xc4, 0x02, 0x01, 0x88, 0x00, 0x04, 0x10, 0x00, 0x18, 0x02, 0x01, 0x9c,
        0x03, 0x56, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
UCHAR buff[256], str[256];
INT LAC;

SequenceStr("LAC('0x0000' to '0xFFFF')", str, NO_TIMEOUT );  // A part
ConvCharToArray(str, buff, sizeof( buff )); // B part
LAC = Msb2IntIE(buff, 16); // B part
// set LAC value
Int2MsbIE( LAC, buff, 16 ); // C part
ReplaceIE( SndData, buff, 38, 16 ); // D part
SIB_POS = 2;
SIB_REP = 32;
SndMessage( UNIT_BTS1, RLC_TR_DATA_REQ, D_BCCH, 0, SndData, sizeof(SndData));
SequenceDisp( " send 'SIB1'" );
```

Appendix (Global variable over a scenario)

1. When we use a scenario that is divided each part such as '0001.c' for idle, '0021.c' for mobile terminated call, etc…, in some case, we would like to use a global variable that we can use more than 1 scenario.

2. We can see such global variables.
(please refer the 'c:¥MX847010¥scenario¥include¥scenario.h file)
you can see the following parameter; (the following is a little example)

```
extern  DLLIMPORT   INT     UserVar0;
extern  DLLIMPORT   INT     UserVar1;
extern  DLLIMPORT   UCHAR   UserMem0[256];
extern  DLLIMPORT   UCHAR   UserMem1[256];
```

we can use these variables for more than 1 scenario.

Part 2: Modifying wireless protocol messages

RRC Layer message modification 1

When we modify RRC layer message, we can modify this by using the 'Message Coder' tool. The following is a example for 'RRC Connection Release'.

```
/* Send Message: RRC Connection Release */
{
        UCHAR SndData[] = {
                0x3c, 0x82, 0x00
        };

        SndMessageIntegrity( UNIT_BTS1, RLC_UM_DATA_REQ, D_DCCH, 0, SndData, 18);
        SequenceDisp( " send 'RRC Connection Release'" );
};
```

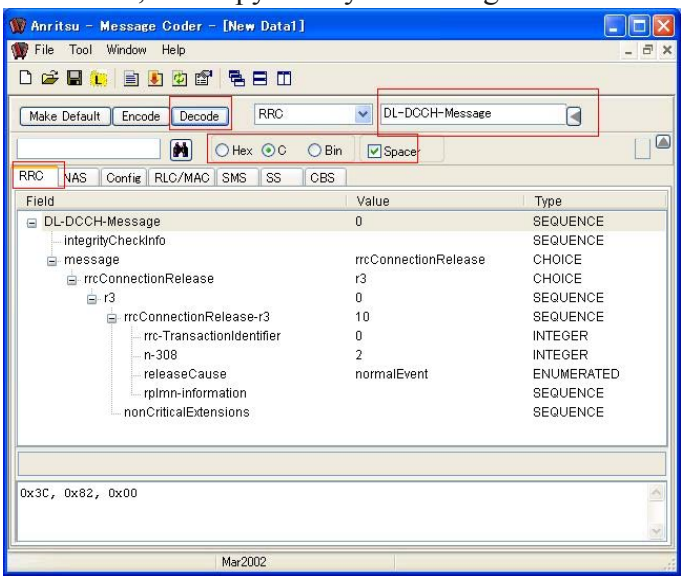
D_DCCH : DL-DCCH message

18 : length of this message

RRC Layer message modification 2

Please execute the 'Message Coder' tool, and copy the layer3 message to this tool.

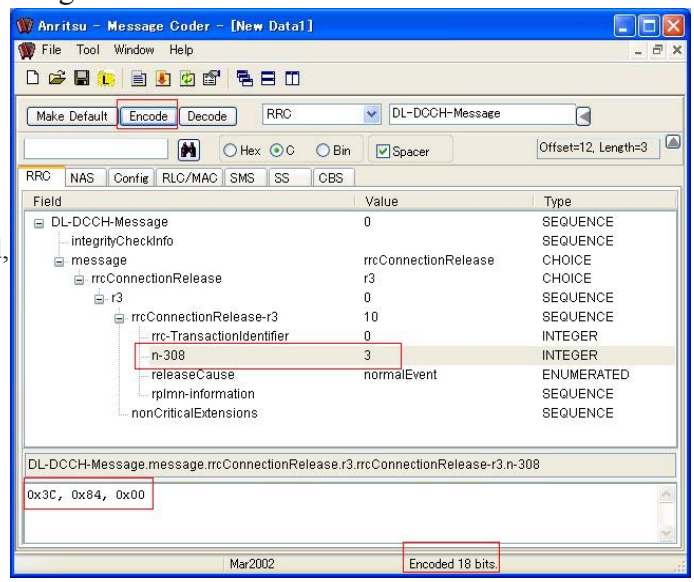
- Select 'RRC'
- match the format (in this case 'C' and 'Spacer-ON')
- select 'DL-DCCH-msg'
- push 'Decode' button
- Then we can see the decoded data



RRC Layer message modification 3

Please modify the layer3 message in this tool.

- modify some parameters
- push 'Encode' button
- copy the modified data to scenario (bottom of this)
- if the bit-length is changed, please modify the length in the scenario



RRC Layer message modification 4

Please copy the modified RRC layer message, if the modified data length is different from original one, please change the data length too.

```
/* Send Message: RRC Connection Release */
{
        UCHAR SndData[] = {
                0x3c, 0x84, 0x00
        };

        SndMessageIntegrity( UNIT_BTS1, RLC_UM_DATA_REQ, D_DCCH, 0, SndData, 18);
        SequenceDisp( " send 'RRC Connection Release'" );
};
```

RRC Layer message replacement 1

When we replace some parameters in RRC layer message, we can use the 'Message Coder' tool. The following is a example for PSC(Primary Scrambling Code) replacement in 'RRC Connection Setup'.

```
UCHAR SndData[] = {                    // RRC Connection Setup message
        0x30, 0xe7, 0x12, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x01, 0x00, 0x00, 0x20, 0xd6, 0x06, 0x93, 0xa7, 0x81,
        0x29, 0xe0, 0xd4, 0x20, 0x00, 0x28, 0x42, 0xcf, 0x2b, 0x41,
                              :
        0xf5, 0xea, 0xd3, 0x41, 0xc1, 0x00, 0x00, 0x00, 0x9f, 0xc8,
        0x05, 0x0a, 0x00, 0x01, 0x00, 0x48, 0x00, 0x01, 0x4f, 0x00 };
UCHAR buff[16];


Int2MsbIE( PrimaryScramblingCode, buff, 9 );
ReplaceIE( SndData, buff, 836, 9 );
```

836 : Why 836th data?

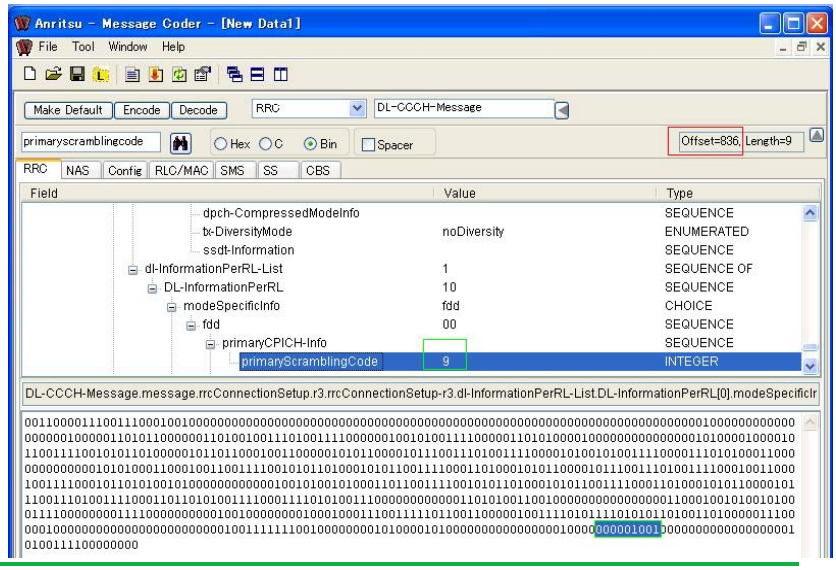
RRC Layer message replacement 2

When we decode the RRC layer message, then please find the parameter that you need to replace. You can see the 'Offset'. This 'Offset' is the 'Start-bit' for replacement

In this case, the 'Offset' is 836.

We can confirm 'How many bit for replace'.

In this case, 9bit length.



NAS Layer message modification 1

When we modify NAS layer message, we can modify this by using the 'Message Coder' tool. The following is a example for 'Identity Request'.

```
/* Send Message: Identity Request */
{
        UCHAR SndData[] = {
                0x14, 0x00, 0x04, 0x0a, 0x30, 0x02
        };

        SndMessageIntegrity(UNIT_BTS1,RLC_AM_DATA_REQ,D_DCCH,2,SndData,47);
        SequenceDisp( " send 'Identity Request'" );
};
```

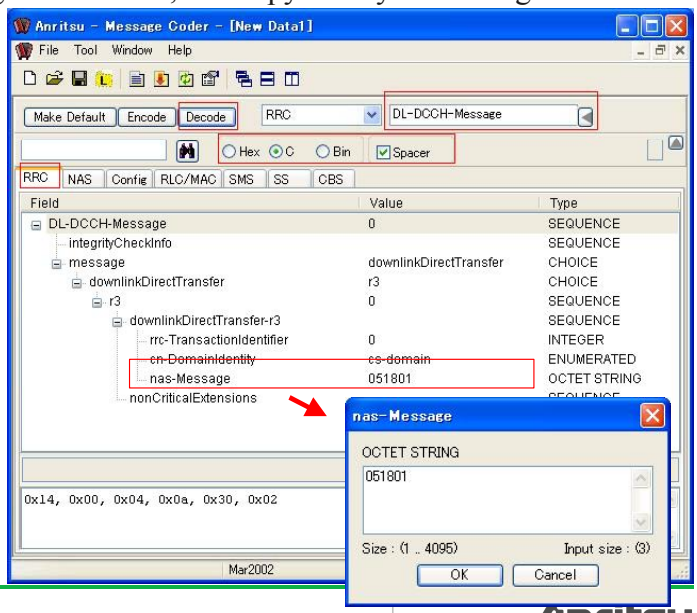
D_DCCH : DL-DCCH message

47 : length of this message

NAS Layer message modification 2

Please execute the 'Message Coder' tool, and copy the layer3 message to this tool.

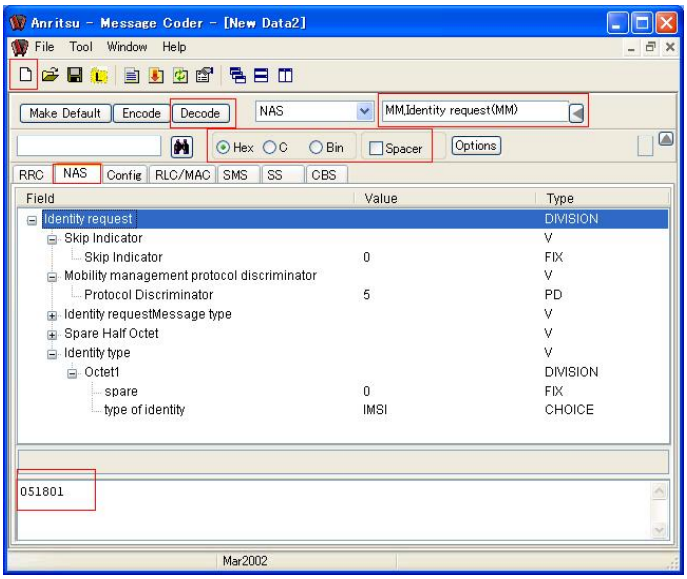
- Select 'RRC'
- match the format (in this case 'C' and 'Spacer-ON')
- select 'DL-DCCH-msg'
- push 'Decode' button
- Copy the nas-message part



NAS Layer message modification 3

Please decode nas-message part.

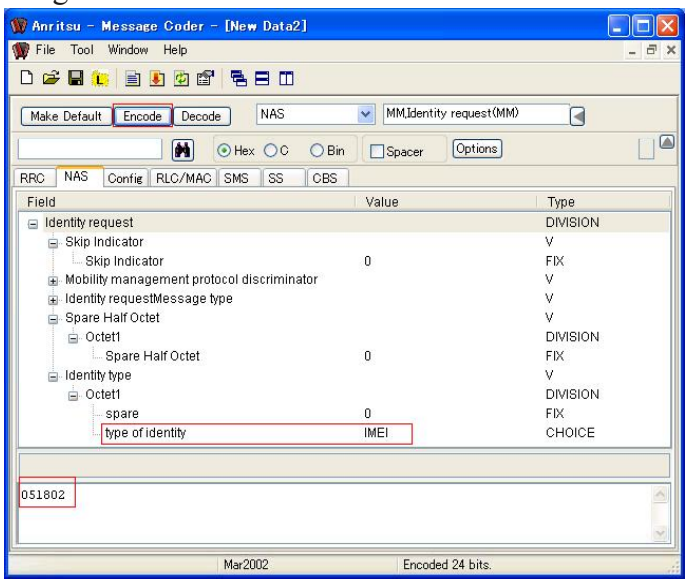
- Open new screen
- Select 'NAS'
- match the format (in this case 'Hex' and 'Spacer-Off')
- Paste nas-message part
- Select message type (in this case MM : Identity request)
- push 'Decode' button
- Then we can see the decoded data in the nas-message part



NAS Layer message modification 4

Please modify the layer3 message in this tool.

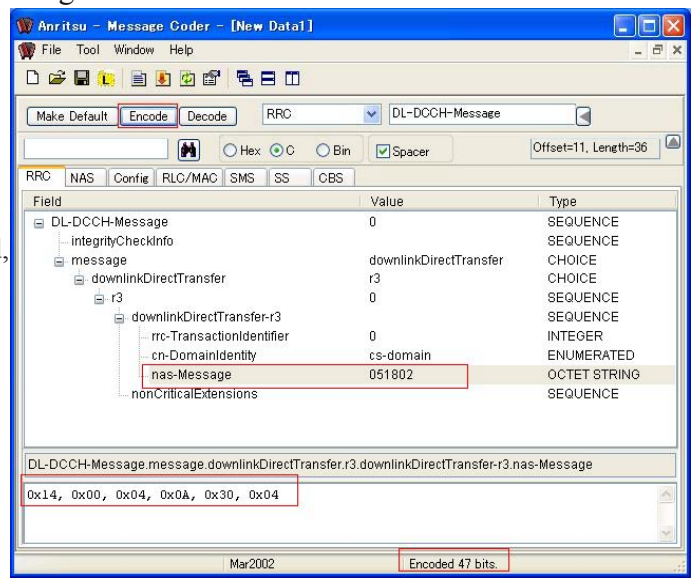
- modify some parameters
- push 'Encode' button
- copy the modified data to RRC part (bottom of this)



NAS Layer message modification 5

Please modify the layer3 message in this tool.

- paste nas-message part
- push 'Encode' button
- copy the modified data to scenario (bottom of this)
- if the bit-length is changed, please modify the length in the scenario



NAS Layer message modification 6

Please copy the modified NAS layer message, if the modified data length is different from original one, please change the data length too.

```
/* Send Message: Identity Request */
{
        UCHAR SndData[] = {
                0x14, 0x00, 0x04, 0x0a, 0x30, 0x04
        };

        SndMessageIntegrity(UNIT_BTS1,RLC_AM_DATA_REQ,D_DCCH,2,SndData,47);
        SequenceDisp( " send 'Identity Request'" );
};
```

NAS Layer message replacement 1

When we replace some parameters in NAS layer message, we can use the 'Message Coder' tool. The following is a example for LAC(Location Area Code) replacement in 'Location Updating Accept'.

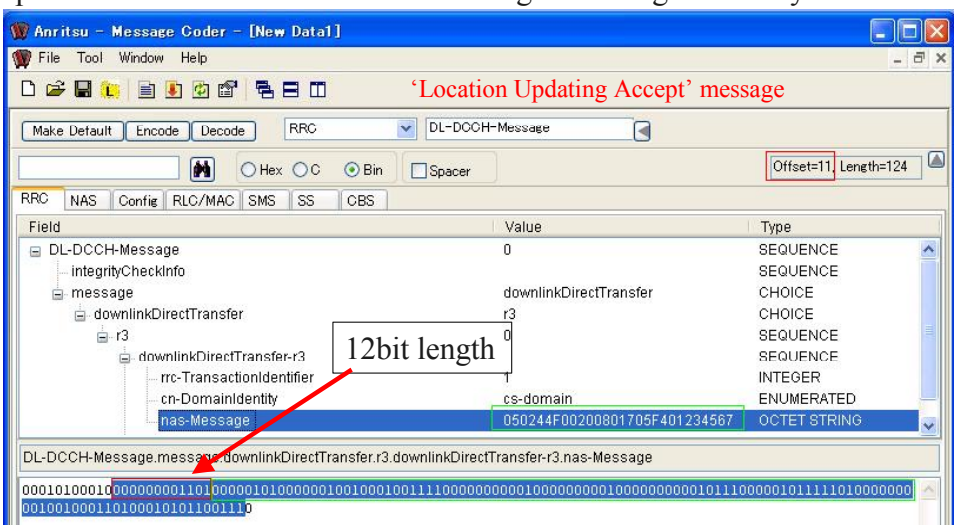
```
UCHAR SndData[] = {
        0x14, 0x40, 0x1a, 0x0a, 0x04, 0x89, 0xe0, 0x04,
        0x01, 0x00, 0x2e, 0x0b, 0xe8, 0x02, 0x46, 0x8a,
        0xce
};

UCHAR buff[256];
// set LAC value
Int2MsbIE( LAC, buff, 16 );
ReplaceIE( SndData, buff, 63, 16 );
```

63 : Why 63th data?

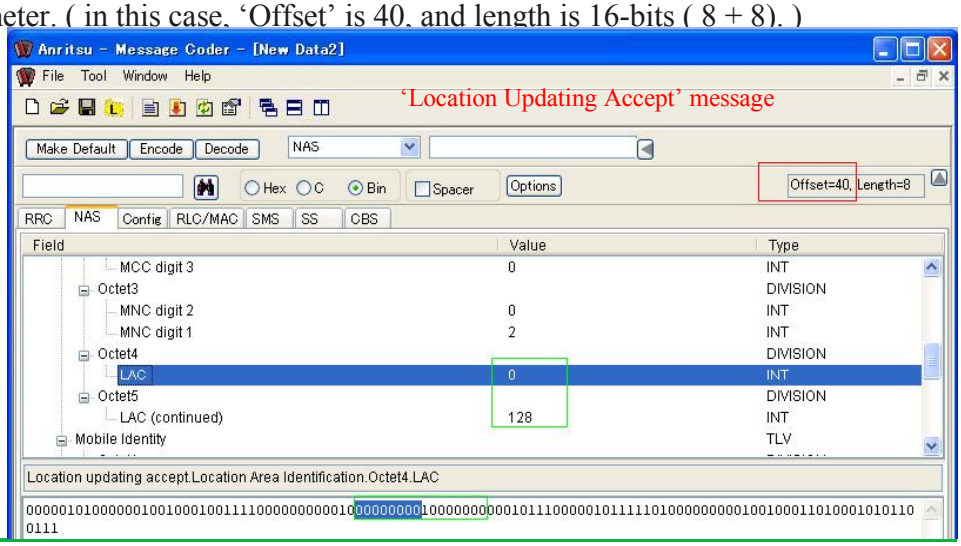
NAS Layer message replacement 2

When we decode the NAS layer message, then please find the nas-message parameter. Then you can see the 'Offset'. This 'Offset' is the 'Start-bit' for nas-message part. And next is the field for nas-message area length. Usually this field is 12bits



NAS Layer message replacement 3

And then when we decode the NAS layer message, then please find the parameter that you replace. Then you can see the 'Offset'. This 'Offset' is the 'Start-bit' for replacing parameter part. Then we can confirm the length of this parameter. (in this case, 'Offset' is 40, and length is 16-bits (8 + 8).)



NAS Layer message replacement 4

So the start bit of the replacement parameter is the following;

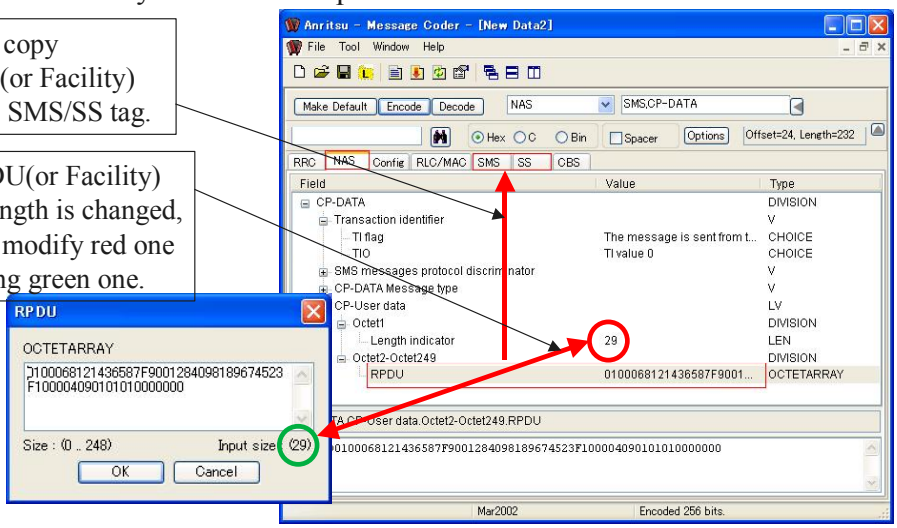
nas-message 'Offset' (in this case 11)
+ length of nas-message length field (in this case 12, usually 12bits)
+ 'Offset' in the NAS message part (in this case 40)
So in this case, the start bit of LAC is 63 (11 + 12 + 40)

Appendix (SMS/SS Layer message) 1

SMS/SS layer messages are contained in the NAS message, so in the modification and replacement case, please go into more higher layer. But the operation is basically same as NAS part.

Please copy RPDU(or Facility) data to SMS/SS tag.

If RPDU(or Facility) data length is changed, please modify red one by using green one.



Appendix (SMS/SS Layer message) 2

The start-bit of the SMS/SS layer messages are following;

'Offset' of nas-message
+ length of nas-message length field (in this case 12, usually 12bits)
+ 'Offset' in the NAS message tag
+ 'Offset' in the SMS/SS message tag

SIB (System Information Message) modification 1

When we modify SIB message, we can modify this by using the 'Message Coder' tool. The following is a example for 'SIB Type1'.

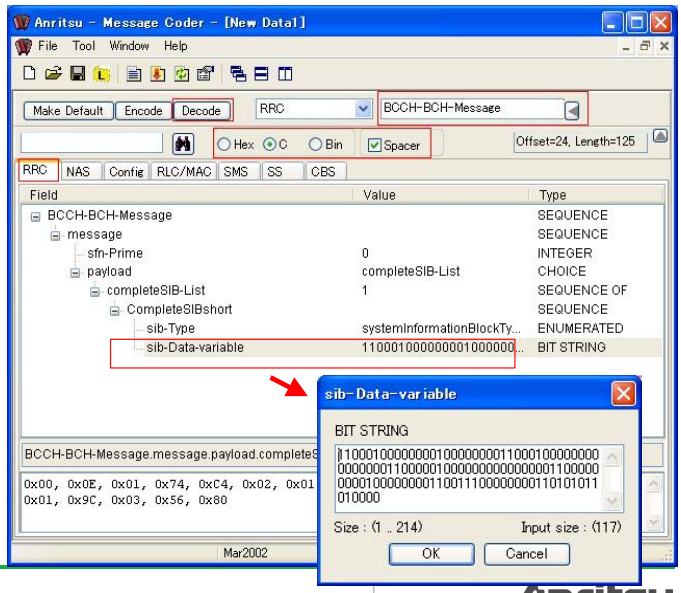
```
UCHAR SndData[] = {    // SIB Type1
            0x00, 0x0e, 0x01, 0x74, 0xc4, 0x02, 0x01, 0x88,
            0x00, 0x04, 0x10, 0x00, 0x18, 0x02, 0x01, 0x9c,
            0x03, 0x56, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
SIB_POS = 2;
SIB_REP = 32;
SndMessage( UNIT_BTS1, RLC_TR_DATA_REQ, D_BCCH, 0, SndData, sizeof(SndData));
SequenceDisp( " send 'SIB1'" );
```

D_BCCH : BCCH-BCH-Message

SIB (System Information Message) modification 2

Please execute the 'Message Coder' tool, and copy the layer3 message to this tool.

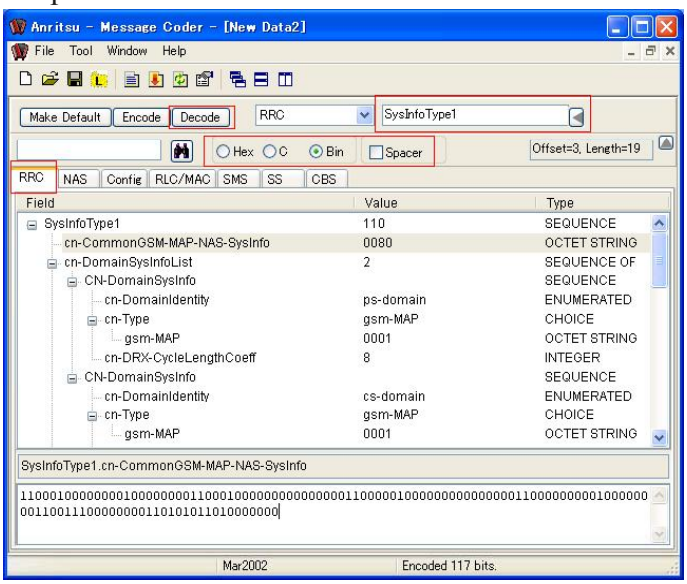
- Select 'RRC'
- match the format (in this case 'C' and 'Spacer-ON')
- select 'BCCH-BCH-msg'
- push 'Decode' button
- Copy the sib-data-variable part



SIB (System Information Message) modification 3

Please decode sib-data-variable part.

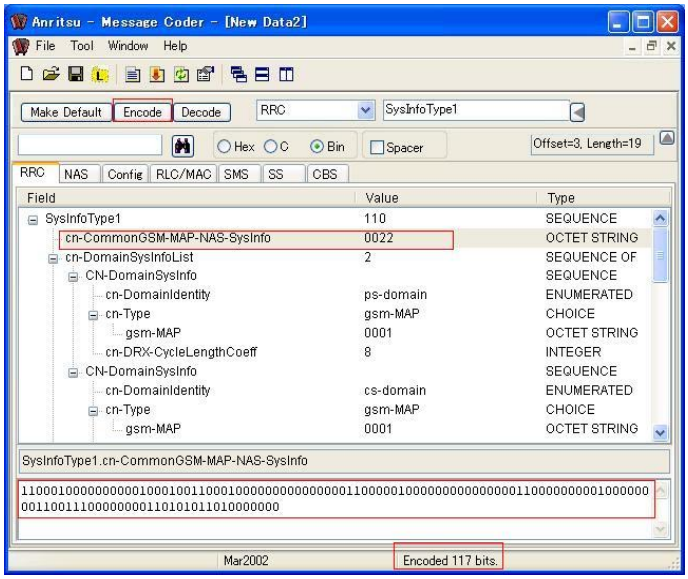
- Open new screen
- Select 'RRC'
- Select 'SysInfoType1'
- match the format (in this case 'Bin' and 'Spacer-Off')
- Paste sib-data-variable part
- push 'Decode' button
- Then we can see the decoded data in the SIB message part



SIB (System Information Message) modification 4

Please modify some parameters in this tool.

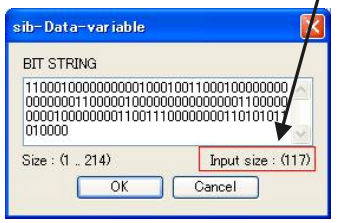
- modify some parameters
- push 'Encode' button
- copy the modified data to RRC part (bottom of this)
- memorize the length

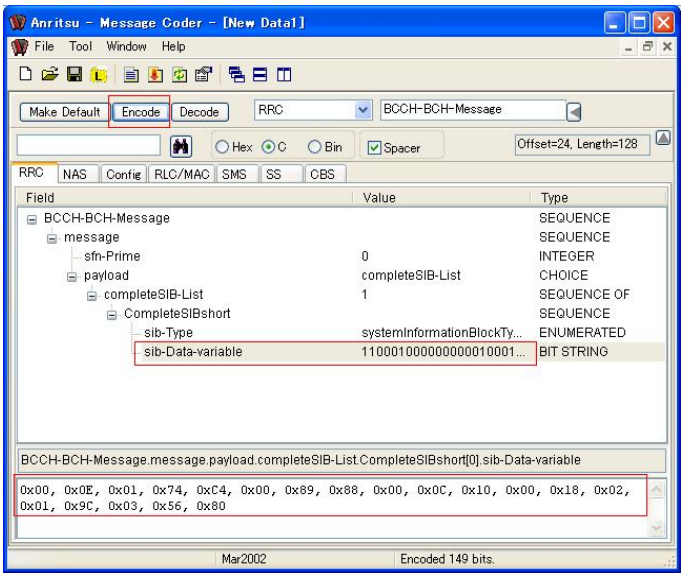


SIB (System Information Message) modification 5

Please modify the layer3 message in this tool.

- paste sib-data-variable part
- match the length in the previous slide
- push 'Encode' button
- copy the modified data to scenario (bottom of this)





SIB (System Information Message) modification 6

Please copy the modified SIB message. And please fill with '0' in the rest of the data. (SIB message is always same length.)

```
UCHAR SndData[] = {     // SIB Type1
            0x00, 0x0e, 0x01, 0x74, 0xc4, 0x00, 0x89, 0x88,
            0x00, 0x04, 0x10, 0x00, 0x18, 0x02, 0x01, 0x9c,
            0x03, 0x56, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
SIB_POS = 2;
SIB_REP = 32;
SndMessage( UNIT_BTS1, RLC_TR_DATA_REQ, D_BCCH, 0, SndData, sizeof(SndData));
SequenceDisp( " send 'SIB1'" );
```

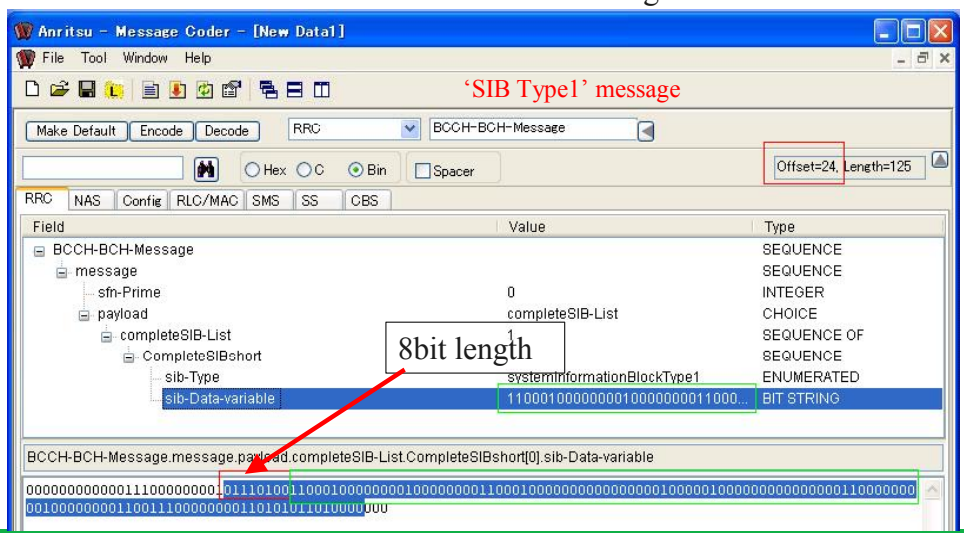
SIB (System Information Message) replacement 1

When we replace some parameters in SIB message, we can use the 'Message Coder' tool. The following is a example for LAC replacement in 'SIB Type1'. WHY 38th data?

```
UCHAR SndData[] = {
        0x00, 0x0e, 0x01, 0x74, 0xc4, 0x02, 0x01, 0x88,
        0x00, 0x04, 0x10, 0x00, 0x18, 0x02, 0x01, 0x9c,
        0x03, 0x56, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
UCHAR buff[256];
// set LAC value
Int2MsbIE( LAC, buff, 16 );
ReplaceIE( SndData, buff, 38, 16 );
SIB_POS = 2;
SIB_REP = 32;
SndMessage( UNIT_BTS1, RLC_TR_DATA_REQ, D_BCCH, 0, SndData, sizeof(SndData));
SequenceDisp( " send 'SIB1'" );
```

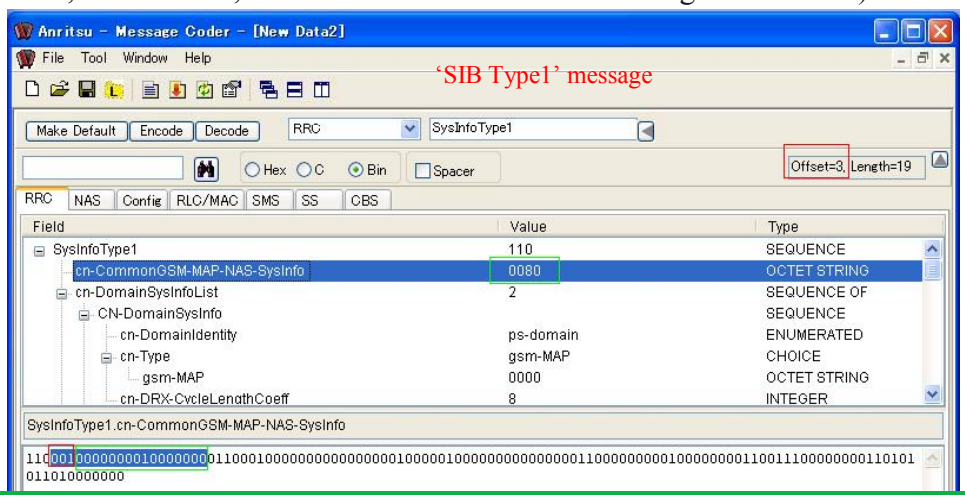

SIB (System Information Message) replacement 2

When we decode the SIB message, then please find the sib-data-variable part. Then you can see the 'Offset'. This 'Offset' is the 'Start-bit' for sib-data-variable part. And next is the field for sib-data-variable area length. This field is 8bits



SIB (System Information Message) replacement 3

And then when we decode the SIB message, then please find the parameter that you replace. Then you can see the 'Offset'. This 'Offset' is the 'Start-bit' for replacing parameter part. Then we can confirm the length of this parameter. (in this case, 'Offset' is 3, and next 3bit'001' is fixed and length is 16-bits.)



SIB (System Information Message) replacement 4

So the start bit of the replacement parameter is the following;

nas-message 'Offset' (in this case 24)
+ length of sib-data-variable length field (in this case 8)
+ 'Offset' in the sib-data-variable part (in this case 3,)
+ more 3bit fixed data '001' (usually nothing)
So in this case, the start bit of LAC is 38 (24 + 8 + 3 + 3)

Appendix (Bigger SIB message)

There are bigger SIB message such as SIB type5 and SIB type11. In these cases, at first please get the sib-data-variable part from all BCCH-BCH-Message. Then please concatenate all sib-data-variable part and paste on the SysInfoType5 or SysInfoType11. After modifying and encoding a SysInfoType5 or 11 message, and then please past sib-data-variable part from first block. The length of these sib-data-variable part is always 222. But only last block case, the length is not 222. (in last block case, please past rest of SysInforType5 or 11 data.)

Message modification trial

Please prepare the scenario and header file (0001.c, 0001.h, 0081.c and 0081.h from W-CDAM sample scenairo). 0081 is SMS MT scenario. And this scenario ask UE to reply 'acknowledgement'. Please modify this scenairo for 'no-request acknowledgement SMS'. (only 0081.h modification is enough by using Message Coder tool.) The key parameter for this is 'TP-RP'.

Part 3: Lower layer parameter modification

General

When we modify the layer3 message that has a relation with lower layer (such as RRC Connection Setup, Radio Bearer Setup), we need to modify not only layer3 message, but also lower layer parameter setting in the MD8470A. About the lower layer parameter setting, there are 2 key point as follows;

1. Parameter Setting : we modify the lower layer parameter.

2. Executing : previous step1 is only modifying some lower layer parameters, for being effected this step1 modification, we need to execute(or re-execute) lower layer.

Example for parameter setting

```
INT InitParam_PhyRlSetup_D_DPCH_AMR()
{
        CPHY_RL_SETUP_PAR *CphyRlSetupPar;

        CphyRlSetupPar = &CphyRlSetup_D_DPCH_AMR;
        memset( CphyRlSetupPar, 0, sizeof(CPHY_RL_SETUP_PAR) );
        CphyRlSetupPar->Offset          = 0;
        CphyRlSetupPar->ScrCode         = 0x00000090;
        CphyRlSetupPar->SlotFormat      = SLOT_FORMAT_8;      /* for DPCCH */
        CphyRlSetupPar->SymbolRate      = SYMRATE30K;         /* for DPDCH */
        CphyRlSetupPar->ChCode          = 5;
        CphyRlSetupPar->Power           = POWER_STEP_01DB(-60);       /* Power = -6.0dB */
                                :
        CphyRlSetupPar->TxDiversity = DIVERSITY_OFF;
        return 0;
}
```

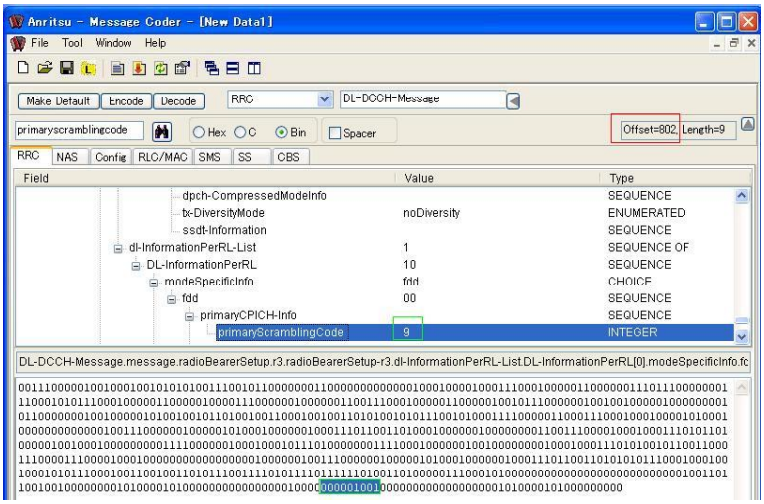

Example for executing

```
CalcRMParameter( D_DPCH, &CphyRlSetup_D_DPCH_AMR, &CphyTrchConfig_D_DPCH_AMR );
CphyRlSetup( UNIT_BTS1, D_DPCH, 0, &CphyRlSetup_D_DPCH_AMR, CFN, NO_TIMEOUT );
CphyTrchConfig( UNIT_BTS1, D_DPCH, 0, &CphyTrchConfig_D_DPCH_AMR, CFN, NO_TIMEOUT );
CmacConfig( UNIT_BTS1, D_DPCH, 0, &CmacConfig_D_DPCH_AMR, CFN, NO_TIMEOUT );
CalcRMParameter( U_DPCH, &CphyRlSetup_U_DPCH_AMR, &CphyTrchConfig_U_DPCH_AMR );
CphyRlSetup( UNIT_BTS1, U_DPCH, 0, &CphyRlSetup_U_DPCH_AMR, CFN, NO_TIMEOUT );
CphyTrchConfig( UNIT_BTS1, U_DPCH, 0, &CphyTrchConfig_U_DPCH_AMR, CFN, NO_TIMEOUT );
CmacConfig( UNIT_BTS1, U_DPCH, 0, &CmacConfig_U_DPCH_AMR, CFN, NO_TIMEOUT );
CrlcConfig( UNIT_BTS1, CRLC_TR_ESTABLISH, DTCH, 0, &AMR_CrlcConfig_DTCH, TE, NO_TIMEOUT );
CrlcConfig( UNIT_BTS1, CRLC_TR_ESTABLISH, DTCH, 1, &AMR_CrlcConfig_DTCH, TE, NO_TIMEOUT );
CrlcConfig( UNIT_BTS1, CRLC_TR_ESTABLISH, DTCH, 2, &AMR_CrlcConfig_DTCH, TE, NO_TIMEOUT );
```

'Executing' library is independent for each layer (Rl(lower layer1), Trch(higher layer1), MAC(lower layer2) and RLC(higher layer2)). And Rl, Trch, MAC layer are independent for uplink and downlink. RLC is common both uplink and downlink. 'CalcRMparameter' is for calculating RM(Rate matching) parameter that is located Trch and MAC, but this calculation needs a parameter in the Rl. So we need to execute this library before 'Executing' each layer. After this 'Executing', parameters that is set up in previous slide is effected.

Example for modification related with lower layer 1

The following is a example for PSC(Primary Scrambling Code) replacement in 'Radio Bearer Setup' that is related with lower layer parameter. (pleaser refer the part2 in detail.) We need modify not only Layer3 message, but also…



Example for modification related with lower layer 2

We need modify not only Layer3 message, but also lower layer setting.

```
INT InitParam_PhyRlSetup_D_DPCH_AMR()
{
        CPHY_RL_SETUP_PAR *CphyRlSetupPar;

        CphyRlSetupPar = &CphyRlSetup_D_DPCH_AMR;
        memset( CphyRlSetupPar, 0, sizeof(CPHY_RL_SETUP_PAR) );
        CphyRlSetupPar->Offset        = 0;
//      CphyRlSetupPar->ScrCode     = 0x00000090;
        CphyRlSetupPar->ScrCode     = ScramblingCode;
        CphyRlSetupPar->SlotFormat      = SLOT_FORMAT_8;      /* for DPCCH */
                                :
        CphyRlSetupPar->TxDiversity = DIVERSITY_OFF;
        return 0;
}
```

Part 4: TE (Terminal Equipment) part

General

For TE layer, there are 2 key point as follow;

1. Parameter Setting & Config: we modify the TE layer parameter and Configure(execute) to be effected these parameters in TE layer.

2. Connecting : previous step1 is only setting up TE layer parameters, when we start the communciation(such as voice, packet, video), we need to connect TE.

Appendix : Only in the case of connecting with ISDN terminal, there is a different way for this.

TE parameter setting & Config (AMR voice) 1

```
{
    CTE_CONFIG_PAR  CteConfigAmrA;  /* Class A Config Parameter */
    CTE_CONFIG_PAR  CteConfigAmrB;  /* Class B Config Parameter */
    CTE_CONFIG_PAR  CteConfigAmrC;  /* Class C Config Parameter */

    CteConfigAmrA.TeType  = TE_TYPE_VOICE_AMR_A;
    CteConfigAmrA.Rate    = VOICE_RATE_12_2;
    CteConfigAmrA.TTI     = 2;
    CteConfigAmrA.NumOfTB = 1;
    CteConfigAmrA.TBS     = 81;
    CteConfigAmrA.Frame   = RLC_TR_DATA_REQ;
    CteConfigAmrA.Layer   = RLC;

    CteConfigAmrB.TeType  = TE_TYPE_VOICE_AMR_B;
    CteConfigAmrB.Rate    = VOICE_RATE_12_2;
    CteConfigAmrB.TTI     = 2;
    CteConfigAmrB.NumOfTB = 1;
    CteConfigAmrB.TBS     = 103;
    CteConfigAmrB.Frame   = RLC_TR_DATA_REQ;
    CteConfigAmrB.Layer   = RLC;
```

To be continued…

TE parameter setting & config (AMR voice) 2

```
    CteConfigAmrC.TeType  = TE_TYPE_VOICE_AMR_C;
    CteConfigAmrC.Rate    = VOICE_RATE_12_2;
    CteConfigAmrC.TTI     = 2;
    CteConfigAmrC.NumOfTB = 1;
    CteConfigAmrC.TBS     = 60;
    CteConfigAmrC.Frame   = RLC_TR_DATA_REQ;
    CteConfigAmrC.Layer   = RLC;

    CteConfig( DTCH, 0, &CteConfigAmrA, NO_TIMEOUT );
    CteConfig( DTCH, 1, &CteConfigAmrB, NO_TIMEOUT );
    CteConfig( DTCH, 2, &CteConfigAmrC, NO_TIMEOUT );
}
```

This is basically always same. (just copy and paste is enough.)

TE connecting (AMR voice)

This is for starting a voice call.

```
CteConnect( DTCH, 0,TE_PORT_NORMAL,TE_PORT_NORMAL, CALL_FROM_AIR, (UCHAR *)0, NO_TIMEOUT );
CteConnect( DTCH, 1,TE_PORT_NORMAL,TE_PORT_NORMAL, CALL_FROM_AIR, (UCHAR *)0, NO_TIMEOUT );
CteConnect( DTCH, 2,TE_PORT_NORMAL,TE_PORT_NORMAL, CALL_FROM_AIR, (UCHAR *)0, NO_TIMEOUT );
```

This is basically always same. (just copy and paste is enough.)

This is for disconnecting a voice call.

```
CteDisconnect( DTCH, 0, CALL_FROM_AIR, NO_TIMEOUT );
CteDisconnect( DTCH, 1, CALL_FROM_AIR, NO_TIMEOUT );
CteDisconnect( DTCH, 2, CALL_FROM_AIR, NO_TIMEOUT );
```

This is basically always same. (just copy and paste is enough.)

TE parameter setting & config (Video Telephony loopback)

```
{
    CTE_CONFIG_PAR CteConfigQ931;    /* Config Parameter */

    CteConfigQ931.TeType = TE_TYPE_TVLOOPBACK;
    CteConfigQ931.Rate    = 0;
    CteConfigQ931.TTI     = 2;
    CteConfigQ931.NumOfTB = 2;
    CteConfigQ931.TBS     = 640;    /* Bit Size */
    CteConfigQ931.Frame   = RLC_TR_DATA_REQ;
    CteConfigQ931.Layer   = RLC;

    CteConfig( DTCH, 0, &CteConfigQ931, NO_TIMEOUT );

};
```

This is basically always same. (just copy and paste is enough.)

TE connecting (Video Telephony loopback)

This is for starting a voice call.

```
CteConnect(DTCH,0,TE_PORT_LOOPBACK,TE_PORT_LOOPBACK,CALL_FROM_AIR,(UCHAR *)0,NO_TIMEOUT);
```

This is basically always same. (just copy and paste is enough.)

This is for disconnecting a voice call.

```
CteDisconnect(DTCH,0,CALL_FROM_AIR,NO_TIMEOUT);
```

This is basically always same. (just copy and paste is enough.)

TE parameter setting & config (IP Packet)

```
{
        CTE_CONFIG_PAR    CteConfigIPPacket;    /* Config Parameter */

        CteConfigIPPacket.TeType  = TE_TYPE_IPPACKET;
        CteConfigIPPacket.Rate    = 0;
        CteConfigIPPacket.TTI     = 1;
        CteConfigIPPacket.NumOfTB = 1;
        CteConfigIPPacket.TBS     = 336;       /* Bit Size */
        CteConfigIPPacket.Frame   = RLC_AM_DATA_REQ;
        CteConfigIPPacket.Layer   = RLC;

        CteConfig(DTCH,0,&CteConfigIPPacket,NO_TIMEOUT);
};
```

This is basically always same. (just copy and paste is enough.)

TE connecting (IP Packet)

This is for starting a voice call.

```
CteConnect(DTCH,0,TE_PORT_NORMAL,TE_PORT_NORMAL,CALL_FROM_AIR,(UCHAR *)0,NO_TIMEOUT);
```

This is basically always same. (just copy and paste is enough.)

This is for disconnecting a voice call.

```
CteDisconnect(DTCH,0,CALL_FROM_AIR,NO_TIMEOUT);
```

This is basically always same. (just copy and paste is enough.)

TE parameter setting & config (Video Telephony ISDN terminal)

```
{
    CTE_CONFIG_PAR CteConfigQ931;    /* Config Parameter */

    CteConfigQ931.TeType = TE_TYPE_ISDN_Q931UDI;
    CteConfigQ931.Rate    = 0;
    CteConfigQ931.TTI     = 2;
    CteConfigQ931.NumOfTB = 2;
    CteConfigQ931.TBS     = 640;    /* Bit Size */
    CteConfigQ931.Frame   = RLC_TR_DATA_REQ;
    CteConfigQ931.Layer   = RLC;

    CteConfig( DTCH, 0, &CteConfigQ931, NO_TIMEOUT );

};
```

This is basically always same. (just copy and paste is enough.)

TE connecting (Video Telephony ISDN terminal) 1

We do not need to execute ‘CteConnect’ and ‘CteDisconnect’ libraries in this case, instead of this, we need to execute the following ‘SndMessage’ and RcvMessage’ libraries. (please refer the W01_viode.c in detail.) The following is MO case.

```
/************************************************/
/*      Mobile Origination case arrow diagram     */
/* MD8470A                                  ISDN */
/*  |----- Q931 Setup                     --->|  */
/*  |                                        |  */
/*  |<---- Q931 Call Proceeding           -----|  */
/*  |                                        |  */
/*  |<---- Q931 Alert                     -----|  */
/*  |                                        |  */
/*  |<---- Q931 Connect                   -----|  */
/*  |                                        |  */
/*  |----- Q931 Connect Acknowledgement   ---->|  */
/*  |                                        |  */
/*               End Of Sequence                */
/************************************************/
```


TE connecting (Video Telephony ISDN terminal) 2

The following is the MT case.

```
/************************************************/
/*      Mobile Termination case arrow diagram     */
/* MD8470A                                  ISDN */
/*  |<---- Q931 Setup                     -----|  */
/*  |                                        |  */
/*  |----- Q931 Call Proceeding           ---->|  */
/*  |                                        |  */
/*  |----- Q931 Alert                     ---->|  */
/*  |                                        |  */
/*  |----- Q931 Connect                   ---->|  */
/*  |                                        |  */
/*  |<---- Q931 Connect Acknowledgement   -----|  */
/*  |                                        |  */
/*               End Of Sequence                */
/************************************************/
```

Part 5: Protocol Sequence Modification

General (Protocol Sequence Modification)

The protocol sequence in the scenario is consist on the 'SndMessage' and 'RcvMessage' part. So please add, delete and modify then is enough to modify the protocol sequence in the scenario. For modifying this, we need a wireless protocol sequence. (Based on the experience, we can care this wireless protocol sequence more easier.)

Anritsu

Specifications are subject to change without notice.

ANRITSU CORPORATION
5-1-1 Onna, Atsugi-shi, Kanagawa, 243-8555 Japan
Phone: +81-46-223-1111
Fax: +81-46-296-1264

● **U.S.A.**
ANRITSU COMPANY
TX OFFICE SALES AND SERVICE
1155 East Collins Blvd., Richardson, TX 75081, U.S.A.
Toll Free: 1-800-ANRITSU (267-4878)
Phone: +1-972-644-1777
Fax: +1-972-644-3416

● **Canada**
ANRITSU ELECTRONICS LTD.
700 Silver Seven Road, Suite 120, Kanata,
ON K2V 1C3, Canada
Phone: +1-613-591-2003
Fax: +1-613-591-1006

● **Brasil**
ANRITSU ELETRÔNICA LTDA.
Praca Amadeu Amaral, 27 - 1 andar
01327-010 - Paraiso, Sao Paulo, Brazil
Phone: +55-11-3283-2511
Fax: +55-11-3886940

● **U.K.**
ANRITSU LTD.
200 Capability Green, Luton, Bedfordshire LU1 3LU, U.K.
Phone: +44-1582-433280
Fax: +44-1582-731303

● **Germany**
ANRITSU GmbH
Nemetschek Haus Konrad-Zuse-Platz 1 81829
München, Germany
Phone: +49 (0) 89 442308-0
Fax: +49 (0) 89 442308-55

● **France**
ANRITSU S.A.
9, Avenue du Québec Z.A. de Courtabœuf 91951 Les
Ulis Cedex, France
Phone: +33-1-60-92-15-50
Fax: +33-1-64-46-10-65

● **Italy**
ANRITSU S.p.A.
Via Elio Vittorini, 129, 00144 Roma EUR, Italy
Phone: +39-06-509-9711
Fax: +39-06-502-2425

● **Sweden**
ANRITSU AB
Borgafjordsgatan 13 164 40 Kista, Sweden
Phone: +46-853470700
Fax: +46-853470730

● **Finland**
ANRITSU AB
Teknobulevardi 3-5, FI-01530 Vantaa, Finland
Phone: +358-9-4355-220
Fax: +358-9-4355-2250

● **Denmark**
Anritsu AB Danmark
Korskildelund 6 DK - 2670 Greve, Denmark
Phone: +45-36915035
Fax: +45-43909371

● **Singapore**
ANRITSU PTE LTD.
10, Hoe Chiang Road #07-01/02, Keppel Towers,
Singapore 089315
Phone: +65-6282-2400
Fax: +65-6282-2533

● **Hong Kong**
ANRITSU COMPANY LTD.
Suite 923, 9/F., Chinachem Golden Plaza, 77 Mody
Road, Tsimshatsui East, Kowloon, Hong Kong, China
Phone: +852-2301-4980
Fax: +852-2301-3545

● **P. R. China**
ANRITSU COMPANY LTD.
Beijing Representative Office
Room 1515, Beijing Fortune Building, No. 5 North
Road, the East 3rd Ring Road, Chao-Yang District
Beijing 100004, P.R. China
Phone: +86-10-6590-9230

● **Korea**
ANRITSU CORPORATION
8F Hyun Juk Bldg. 832-41, Yeoksam-dong,
Kangnam-ku, Seoul, 135-080, Korea
Phone: +82-2-553-6603
Fax: +82-2-553-6604

● **Australia**
ANRITSU PTY LTD.
Unit 3/170 Forster Road Mt. Waverley, Victoria, 3149,
Australia
Phone: +61-3-9558-8177
Fax: +61-3-9558-8255

● **Taiwan**
ANRITSU COMPANY INC.
7F, No. 316, Sec. 1, NeiHu Rd., Taipei, Taiwan
Phone: +886-2-8751-1816
Fax: +886-2-8751-1817

051114

Printed on 100% Recycled Paper

No. MD8470A-E-E-5-(1.00) 公知 Printed in Japan 2006-3 AKD