ap端涉及到的修改文件及代码如下：

~\vendor\qcom\proprietary\qcril\qcril\_qmi\qcril.c

if( cmd\_id >= QCRIL\_EVT\_HOOK\_MAX

&& (!(cmd\_id >= QCRIL\_EVT\_TINNO\_RIL\_BASE && cmd\_id <= QCRIL\_EVT\_TINNO\_MAX)))//zyh 1109

{

QCRIL\_LOG\_ERROR( "Received un expected command id = %lu", cmd\_id );

break;

}

~\vendor\qcom\proprietary\qcril\qcril\_qmi\qcrili.h

//20141031 zyh add for writing&reading nv data

QCRIL\_EVT\_TINNO\_RIL\_BASE = 0x91000,

QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_READ = QCRIL\_EVT\_TINNO\_RIL\_BASE + 1,

QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_WRITE = QCRIL\_EVT\_TINNO\_RIL\_BASE + 2,

QCRIL\_EVT\_TINNO\_MAX,

//20141031 zyh end

~\vendor\qcom\proprietary\qcril\qcril\_qmi\oem\_socket\qcril\_qmi\_oem\_socket.cc

424行代码开始：

if(RIL\_E\_SUCCESS!= audit\_result&&(param.event\_id!=0x91001)&&(param.event\_id!=0x91002))//zyh 1109

{

QCRIL\_LOG\_ERROR( "Send error response" );

qcril\_default\_request\_resp\_params(param.instance\_id,param.t, param.event\_id,audit\_result, &resp );

resp.android\_request\_id = RIL\_REQUEST\_OEM\_HOOK\_RAW;

qcril\_send\_request\_response( &resp );

}

736行代码开始：

boolean qmi\_ril\_get\_req\_details\_from\_oem\_req

(

qmi\_ril\_oem\_hook\_request\_details\_type \*oem\_hook\_req\_details,

RIL\_Errno \*audit\_result,

unsigned char \*data,

qcril\_request\_params\_type \*param,

int recv\_byte\_num

)

{

qcril\_request\_resp\_params\_type resp;//zyh 1109

if (!oem\_hook\_req\_details || !audit\_result || !data || !param)

{

return FALSE;

}

// parse the OEM hook request to distinguish between internal or customer specific requests

if ( ( RIL\_E\_SUCCESS == qmi\_ril\_parse\_oem\_hook\_header( data, oem\_hook\_req\_details ) ) && oem\_hook\_req\_details->is\_oem\_hook )

{

if ( ( ( oem\_hook\_req\_details->hook\_req > QCRIL\_EVT\_HOOK\_BASE ) && ( oem\_hook\_req\_details->hook\_req < QCRIL\_EVT\_HOOK\_MAX ) ) ||

( ( oem\_hook\_req\_details->hook\_req > QCRIL\_EVT\_OEM\_BASE ) && ( oem\_hook\_req\_details->hook\_req < QCRIL\_EVT\_OEM\_MAX ) ) )

{

// This is an OEM\_HOOK request, Convert it to look like a internal RIL REQUEST

// Move data pointer past the QCRILHook header and re-adjusting the length

param->event\_id = oem\_hook\_req\_details->hook\_req;

param->data = (char \*)data+ QCRIL\_HOOK\_HEADER\_SIZE;

param->datalen = oem\_hook\_req\_details->hook\_req\_len;

if ( ( oem\_hook\_req\_details->hook\_req > QCRIL\_EVT\_OEM\_BASE ) && ( oem\_hook\_req\_details->hook\_req < QCRIL\_EVT\_OEM\_MAX ) )

{

// externally provided by an OEM, otherwise will be dispatched via qcril\_dispatch\_event()

QCRIL\_LOG\_DEBUG("diverting inbound OEM HOOK request to external handler");

qcrilhook\_oem( param->instance\_id, param->event\_id, (char \*)data, recv\_byte\_num, param->t);

#if 1

//20141031 zyh add,Read&Write NV\_00114(true version) & NV\_00307(customer version)

// parse the OEM hook request to distinguish between internal or customer specific requests

QCRIL\_LOG\_ERROR("zyh print OEM HOOK request\_1,event\_id = 0x%x", param->event\_id);

if ((RIL\_E\_SUCCESS == qmi\_ril\_parse\_oem\_hook\_header(data, oem\_hook\_req\_details)) && oem\_hook\_req\_details->is\_oem\_hook)

{

if ((( oem\_hook\_req\_details->hook\_req > QCRIL\_EVT\_HOOK\_BASE) && (oem\_hook\_req\_details->hook\_req < QCRIL\_EVT\_HOOK\_MAX)) ||

((oem\_hook\_req\_details->hook\_req > QCRIL\_EVT\_OEM\_BASE) && (oem\_hook\_req\_details->hook\_req < QCRIL\_EVT\_OEM\_MAX )))

{

// This is an OEM\_HOOK request, Convert it to look like a internal RIL REQUEST

// Move data pointer past the QCRILHook header and re-adjusting the length

param->event\_id = oem\_hook\_req\_details->hook\_req;

QCRIL\_LOG\_ERROR("zyh print OEM HOOK request\_2,event\_id = 0x%x", param->event\_id);

param->data = (char \*)data + QCRIL\_HOOK\_HEADER\_SIZE;

param->datalen = oem\_hook\_req\_details->hook\_req\_len;

if ( ( oem\_hook\_req\_details->hook\_req > QCRIL\_EVT\_OEM\_BASE ) && ( oem\_hook\_req\_details->hook\_req < QCRIL\_EVT\_OEM\_MAX ) )

{

// externally provided by an OEM, otherwise will be dispatched via qcril\_dispatch\_event()

qmi\_client\_error\_type qmi\_client\_error;

RIL\_Errno rilRet = RIL\_E\_GENERIC\_FAILURE;

char oemResult[QMI\_VOICE\_TINNO\_GENRDWR\_BUFFER\_MAX];

uint32 sys\_prop\_id = 0;

//sys\_prop\_id = 215; //test

QCRIL\_LOG\_ERROR("zyh print OEM HOOK request,event\_id = 0x%x", param->event\_id);

memset(oemResult, 0, sizeof(oemResult));

memcpy(&sys\_prop\_id, (char \*)data + QCRIL\_HOOK\_HEADER\_SIZE, 4);

QCRIL\_LOG\_INFO("zyh print QCRIL\_EVT\_OEM,sys\_prop\_id = 0x%x", sys\_prop\_id);

if (param->event\_id == QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_READ)

{

voice\_get\_config\_req\_msg\_v02 get\_config\_req\_msg\_test;

voice\_get\_config\_resp\_msg\_v02 get\_config\_resp\_msg\_test;

memset(&get\_config\_req\_msg\_test, 0, sizeof(get\_config\_req\_msg\_test));

memset(&get\_config\_resp\_msg\_test, 0, sizeof(get\_config\_resp\_msg\_test));

QCRIL\_LOG\_INFO("--------QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_READ--------sys\_prop\_id:%x",sys\_prop\_id);

get\_config\_req\_msg\_test.sys\_prop\_id\_valid= TRUE;

get\_config\_req\_msg\_test.sys\_prop\_id = sys\_prop\_id;//NV\_DIR\_NUMBER\_PCS\_I;

QCRIL\_LOG\_INFO("qcril\_qmi\_client\_send\_msg\_sync returned error: %d", get\_config\_resp\_msg\_test.resp.error);

qmi\_client\_error = qcril\_qmi\_client\_send\_msg\_sync(QCRIL\_QMI\_CLIENT\_VOICE,

QMI\_VOICE\_GET\_CONFIG\_REQ\_V02,

&get\_config\_req\_msg\_test, sizeof(get\_config\_req\_msg\_test),

&get\_config\_resp\_msg\_test, sizeof(get\_config\_resp\_msg\_test) );

rilRet = qcril\_qmi\_util\_convert\_qmi\_response\_codes\_to\_ril\_result(qmi\_client\_error, &get\_config\_resp\_msg\_test.resp);

QCRIL\_LOG\_INFO("--------QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_READ--------qmi\_client\_error:%d >>> rilRet:%d", qmi\_client\_error, rilRet);

if (RIL\_E\_SUCCESS != rilRet)

{

//QCRIL\_LOG\_INFO("qcril\_qmi\_client\_send\_msg\_sync returned error 2: %d", get\_config\_resp\_msg\_test.resp.error);

}

else

{ /\* check response status \*/

QCRIL\_LOG\_INFO("qcril\_qmi\_client\_send\_msg\_sync returned: ===[0x%x]===resp.result:%d, valid:%d", sys\_prop\_id, get\_config\_resp\_msg\_test.resp.result, get\_config\_resp\_msg\_test.sys\_prop\_data\_valid);

memcpy( &oemResult, &get\_config\_resp\_msg\_test.sys\_prop\_data, sizeof( get\_config\_resp\_msg\_test.sys\_prop\_data ) );

}

}

else if (param->event\_id == QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_WRITE)

{

int setlen;

//uint32 nv\_item\_id = 215;//test

voice\_set\_config\_req\_msg\_v02 set\_config\_req\_msg\_test;

voice\_set\_config\_resp\_msg\_v02 set\_config\_resp\_msg\_test;

memset(&set\_config\_req\_msg\_test, 0, sizeof(set\_config\_req\_msg\_test));

memset(&set\_config\_resp\_msg\_test, 0, sizeof(set\_config\_resp\_msg\_test));

memcpy(&setlen, (char\*)data + QCRIL\_HOOK\_HEADER\_SIZE + 4, 4);

memcpy(oemResult, (char\*)data + QCRIL\_HOOK\_HEADER\_SIZE + 8, setlen);

oemResult[setlen] = '\0';

QCRIL\_LOG\_DEBUG( "--------zyh QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_WRITE--------data(%d):%s", setlen, oemResult);

//set\_config\_req\_msg\_test.nv\_item\_data\_valid = TRUE;

//memcpy(set\_config\_req\_msg\_test.nv\_item\_data, &nv\_item\_id, sizeof(nv\_item\_id));

//memset(set\_config\_req\_msg\_test.nv\_item\_data+sizeof(nv\_item\_id), 10, sizeof(set\_config\_req\_msg\_test.nv\_item\_data)-sizeof(nv\_item\_id));

memcpy(set\_config\_req\_msg\_test.sys\_prop\_data, &sys\_prop\_id, sizeof(sys\_prop\_id));

memcpy(set\_config\_req\_msg\_test.sys\_prop\_data+sizeof(sys\_prop\_id), oemResult, setlen);

set\_config\_req\_msg\_test.sys\_prop\_data\_valid = TRUE;

qmi\_client\_error = qcril\_qmi\_client\_send\_msg\_sync(QCRIL\_QMI\_CLIENT\_VOICE,

QMI\_VOICE\_SET\_CONFIG\_REQ\_V02,

&set\_config\_req\_msg\_test,

sizeof(set\_config\_req\_msg\_test),

&set\_config\_resp\_msg\_test,

sizeof(set\_config\_resp\_msg\_test) );

rilRet = qcril\_qmi\_util\_convert\_qmi\_response\_codes\_to\_ril\_result(qmi\_client\_error, &set\_config\_resp\_msg\_test.resp);

QCRIL\_LOG\_INFO("--------zyh QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_WRITE--------qmi\_client\_error:%d >>> rilRet:%d", qmi\_client\_error, rilRet);

QCRIL\_LOG\_INFO("--------zyh QCRIL\_EVT\_TINNO\_RIL\_SYSPROP\_WRITE--------valid:%d, sys\_prop\_outcome:%d", set\_config\_resp\_msg\_test.sys\_prop\_outcome\_valid, set\_config\_resp\_msg\_test.sys\_prop\_outcome);

sprintf(oemResult, "%d", set\_config\_resp\_msg\_test.sys\_prop\_outcome);

if ( RIL\_E\_SUCCESS != rilRet )

{

QCRIL\_LOG\_INFO("Response is invalid");

}

}

qcril\_default\_request\_resp\_params( param->instance\_id, param->t, param->event\_id, rilRet, &resp );

resp.resp\_len = QMI\_VOICE\_TINNO\_GENRDWR\_BUFFER\_MAX;//strlen( oemResult );

resp.resp\_pkt = ( void \* )oemResult ;

qcril\_send\_request\_response( &resp );

//2014/10/31 zyh. mark audit\_result as a success as everything is already handled above regardless of read/write success or failure

//so common handler outside while loop won't send response again.

\*audit\_result = RIL\_E\_SUCCESS;

}

}

else

{

// The request is not supported

\*audit\_result = RIL\_E\_REQUEST\_NOT\_SUPPORTED;

QCRIL\_LOG\_DEBUG("zyh print RIL\_E\_REQUEST\_NOT\_SUPPORTED,event\_id = 0x%x", param->event\_id);

return FALSE;

}

}

else

{

// The request is not supported

\*audit\_result = RIL\_E\_REQUEST\_NOT\_SUPPORTED;

QCRIL\_LOG\_DEBUG("zyh print RIL\_E\_REQUEST\_NOT\_SUPPORTED,event\_id = 0x%x", param->event\_id);

return FALSE;

}

#endif

\*audit\_result = RIL\_E\_SUCCESS;

return FALSE;

}

\*audit\_result = RIL\_E\_SUCCESS;

}

else

{

// The request is not supported

\*audit\_result = RIL\_E\_REQUEST\_NOT\_SUPPORTED;

return FALSE;

}

}

else

{

// The request is not supported

\*audit\_result = RIL\_E\_REQUEST\_NOT\_SUPPORTED;

return FALSE;

}

return TRUE;

}

~\vendor\qcom\proprietary\qmi\services\voice\_service\_v02.c

2699行开始：

static const uint8\_t voice\_ussd\_ind\_msg\_data\_v02[] =

{

……

//20141031 zyh add for writing nv data to modem

,QMI\_IDL\_TLV\_FLAGS\_LAST\_TLV | QMI\_IDL\_TLV\_FLAGS\_OPTIONAL | (QMI\_IDL\_OFFSET8(voice\_set\_config\_req\_msg\_v02, sys\_prop\_data) - QMI\_IDL\_OFFSET8(voice\_set\_config\_req\_msg\_v02, sys\_prop\_data\_valid)),

0x4F,

QMI\_IDL\_FLAGS\_IS\_ARRAY| QMI\_IDL\_GENERIC\_1\_BYTE | QMI\_IDL\_FLAGS\_SZ\_IS\_16,

QMI\_IDL\_OFFSET8(voice\_set\_config\_req\_msg\_v02, sys\_prop\_data),

00,02

//20141031 zyh end

……

};

2743行开始

static const uint8\_t voice\_set\_config\_resp\_msg\_data\_v02[] =

{

……

//20141031 zyh add for writing nv data to modem

,QMI\_IDL\_TLV\_FLAGS\_LAST\_TLV | QMI\_IDL\_TLV\_FLAGS\_OPTIONAL | (QMI\_IDL\_OFFSET8(voice\_set\_config\_resp\_msg\_v02, sys\_prop\_outcome) - QMI\_IDL\_OFFSET8(voice\_set\_config\_resp\_msg\_v02, sys\_prop\_outcome\_valid)),

0x4F,

QMI\_IDL\_GENERIC\_1\_BYTE,

QMI\_IDL\_OFFSET8(voice\_set\_config\_resp\_msg\_v02, sys\_prop\_outcome )

//20141031 zyh end

……

};

2796行开始:

static const uint8\_t voice\_get\_config\_req\_msg\_data\_v02[] =

{

……

//20141031 zyh add for reading nv data from modem

,QMI\_IDL\_TLV\_FLAGS\_LAST\_TLV | QMI\_IDL\_TLV\_FLAGS\_OPTIONAL | (QMI\_IDL\_OFFSET8(voice\_get\_config\_req\_msg\_v02, sys\_prop\_id) - QMI\_IDL\_OFFSET8(voice\_get\_config\_req\_msg\_v02, sys\_prop\_id\_valid)),

0x4F,

QMI\_IDL\_GENERIC\_4\_BYTE,

QMI\_IDL\_OFFSET8(voice\_get\_config\_req\_msg\_v02, sys\_prop\_id )

//20141031 zyh end

……

};

2853行开始:

static const uint8\_t voice\_get\_config\_resp\_msg\_data\_v02[] =

{

……

//20141031 zyh add for reading nv data from modem

,QMI\_IDL\_TLV\_FLAGS\_LAST\_TLV | QMI\_IDL\_TLV\_FLAGS\_OPTIONAL | (QMI\_IDL\_OFFSET8(voice\_get\_config\_resp\_msg\_v02, sys\_prop\_data) - QMI\_IDL\_OFFSET8(voice\_get\_config\_resp\_msg\_v02, sys\_prop\_data\_valid)),

0x4F,

QMI\_IDL\_FLAGS\_IS\_ARRAY| QMI\_IDL\_GENERIC\_1\_BYTE | QMI\_IDL\_FLAGS\_SZ\_IS\_16,

QMI\_IDL\_OFFSET8(voice\_get\_config\_resp\_msg\_v02, sys\_prop\_data),

00,02

//20141031 zyh end

……

};

3906行开始:

static const qmi\_idl\_service\_message\_table\_entry voice\_service\_command\_messages\_v02[] =

{

……

{QMI\_VOICE\_SET\_CONFIG\_REQ\_V02, QMI\_IDL\_TYPE16(0, 53), 1024},//zyh change 39 to 1024

{QMI\_VOICE\_GET\_CONFIG\_REQ\_V02, QMI\_IDL\_TYPE16(0, 55), 512},//zyh change 36 to 512

……

};

3953行开始:

static const qmi\_idl\_service\_message\_table\_entry voice\_service\_response\_messages\_v02[] =

{

……

{QMI\_VOICE\_GET\_CONFIG\_RESP\_V02, QMI\_IDL\_TYPE16(0, 56), 1024},//zyh changed 55 to 1024 as voice\_get\_config\_resp\_msg\_v02 is added more fields.

……

};

~\vendor\qcom\proprietary\qmi\services\voice\_service\_v02.h

172行开始：

//20141031 zyh add for reading&writing nv

#define QMI\_VOICE\_TINNO\_GENRDWR\_BUFFER\_MAX 512 //max buffer size for tinno general read/write.

//20141031 zyh end

5761行开始：

typedef struct

{

……

//20141031 zyh add for writing nv data to modem

uint8\_t sys\_prop\_data\_valid;

uint8\_t sys\_prop\_data[QMI\_VOICE\_TINNO\_GENRDWR\_BUFFER\_MAX];

//20141031 zyh end

……

} voice\_set\_config\_req\_msg\_v02; /\* Message \*/

5834行开始：

typedef struct

{

……

//20141031 zyh add for writing nv data to modem

uint8\_t sys\_prop\_outcome\_valid;

uint8\_t sys\_prop\_outcome;

//20141031 zyh end

……

} voice\_set\_config\_resp\_msg\_v02; /\* Message \*/

5925行开始：

typedef struct

{

……

//20141031 zyh add for reading nv data from modem

uint8\_t sys\_prop\_id\_valid;

uint32\_t sys\_prop\_id;

//20141031 zyh end

……

} voice\_get\_config\_req\_msg\_v02; /\* Message \*/

6033行开始：

typedef struct

{

……

//20141031 zyh add for reading nv data from modem

uint8\_t sys\_prop\_data\_valid;

uint8\_t sys\_prop\_data[QMI\_VOICE\_TINNO\_GENRDWR\_BUFFER\_MAX];

//20141031 zyh end

……

} voice\_get\_config\_resp\_msg\_v02; /\* Message \*/

~\vendor\qcom\proprietary\qcNvItems\src\com\qualcomm\qcnvitems\QcNvItems.java

45行开始:

//20141031 zyh add

private static final int INT\_SIZE = 4;

771行开始:

//20141031 zyh add for reading&writing NV\_FACTORY\_INFO\_I

public String getSysPropInfo(int nvItems) throws IOException

{

if (!mIsQcRilHookReady) {

// return if the QcRilHook isn't ready

return null;

}

AsyncResult result;

String strResult;

//result = mQcRilOemHook.sendQcRilHookMsg(0x91001,114);

result = mQcRilOemHook.sendQcRilHookMsg(0x91001,nvItems);

//result = mQcRilOemHook.sendQcRilHookMsg(0x91000+1,0x80000000+0x100);

if ((result.exception == null) && (result.result != null)) {

Log.w(LOG\_TAG, "zyh read" + bytesToHexString((byte[]) result.result));

strResult = new String((byte[]) result.result);

} else {

Log.w(LOG\_TAG, "zyh Unable to read Factory Info Value from NV Memory");

throw new IOException();

}

return strResult;

}

public void setSysPropInfo(int nvItems, String nvValues/\*, String spc\*/) throws IOException {

if (!mIsQcRilHookReady) {

// return if the QcRilHook isn't ready

return;

}

try {

//checkSpc(spc);

int bufSize = INT\_SIZE;

if (nvValues != null)

bufSize += (INT\_SIZE + nvValues.length());

ByteBuffer buf = ByteBuffer.allocate( bufSize);

buf.order(ByteOrder.nativeOrder());

buf.putInt(nvItems);

if (nvValues != null) {

buf.putInt(nvValues.length());

buf.put(nvValues.getBytes());

}

AsyncResult result = mQcRilOemHook.sendQcRilHookMsg(0x91002,buf.array());

if (result.exception != null) {

Log.w(LOG\_TAG, "zyh Unable to Set Factory Info");

throw new IOException();

}

} catch (InvalidParameterException e) {

Log.w(LOG\_TAG, e.toString());

}

}

//20141031 zyh end

modem端涉及到的修改文件及代码如下：

~\modem\_proc\mmcp\mmode\qmi\src\qmi\_tinno\_sys\_prop.c

新添加的文件,代码如下:

/\*===========================================================================

Q M I \_ T I N N O \_ S Y S \_ P R O P . C

DESCRIPTION

EXTERNALIZED FUNCTIONS

Copyright (c) 2014 TINNO Technologies Incorporated.

All Rights Reserved.

===========================================================================\*/

/\*===========================================================================

EDIT HISTORY FOR FILE

$Header: //source/qcom/qct/modem/mmcp/mmode/qmi/src/qmi\_tinno\_sys\_prop.c $

when who what, where, why

-------- --- ----------------------------------------------------------

2014,10 zyh process and format data for reading and writing modem

===========================================================================\*/

#include "mmcp\_variation.h"

#include "comdef.h"

#include "customer.h"

#include "amssassert.h"

#include "dsm.h"

#include "nv.h"

#include "cm.h"

#include "msg.h"

#include "err.h"

#include "modem\_mem.h"

#include "qmi\_voice\_cm\_if.h"

#include "qmi\_voice\_cmd\_list.h"

#include "qmi\_voice\_call\_list.h"

#include "qmi\_voice\_cm\_sups.h"

#include "qmi\_voice\_cm\_util.h"

#include "qmi\_voice.h"

#include "mm.h"

#include "fs\_lib.h"

#ifdef FEATURE\_DUAL\_SIM

#include "cm\_dualsim.h"

#endif /\*FEATURE\_DUAL\_SIM\*/

#include "pbmlib.h"

#include "qmi\_mmode\_task\_cmd.h"

#include "qmi\_mmode\_msgr\_msg.h"

#include "ref\_cnt\_obj.h"

#include "qm\_nv.h"

#include "qm\_util.h"

#include "qm\_efs.h"

#include "qmi\_voice\_msgr\_if.h"

/\*===========================================================================

INTERNAL DEFINITIONS AND TYPES

===========================================================================\*/

#define SYSPROP\_MDN 1

#define SYSPROP\_MIN 2

#define SYSPROP\_MNC 3

#define SYSPROP\_MIPMODE 6

#define SYSPROP\_MIPPROF 7

#define SYSPROP\_MNSS 8

/\*

\* based on max nv items : 0x7fffffff

\*/

#define SYSPROP\_EXTENSION\_BASE 0x80000000

#define SYSPROP\_EXTENSION\_CHECKMSL (SYSPROP\_EXTENSION\_BASE+1)

#define SYSPROP\_EXTENSION\_SPC (SYSPROP\_EXTENSION\_BASE+2)

#define SYSPROP\_EXTENSION\_MDN (SYSPROP\_EXTENSION\_BASE+3)

#define SYSPROP\_EXTENSION\_MIN (SYSPROP\_EXTENSION\_BASE+4)

#define SYSPROP\_EXTENSION\_MNC (SYSPROP\_EXTENSION\_BASE+5)

#define SYSPROP\_EXTENSION\_SID (SYSPROP\_EXTENSION\_BASE+6)

#define SYSPROP\_EXTENSION\_NID (SYSPROP\_EXTENSION\_BASE+7)

#define SYSPROP\_EXTENSION\_MIPMODE (SYSPROP\_EXTENSION\_BASE+8)

#define SYSPROP\_EXTENSION\_MIPPROF (SYSPROP\_EXTENSION\_BASE+9)

#define SYSPROP\_EXTENSION\_MNSS (SYSPROP\_EXTENSION\_BASE+10)

#define SYSPROP\_EXTENSION\_HASS (SYSPROP\_EXTENSION\_BASE+11)

#define SYSPROP\_EXTENSION\_AAASS (SYSPROP\_EXTENSION\_BASE+12)

#define SYSPROP\_EXTENSION\_MEID (SYSPROP\_EXTENSION\_BASE+13)

#define SYSPROP\_EXTENSION\_SERIAL (SYSPROP\_EXTENSION\_BASE+14)

#define SYSPROP\_1X\_EVDO\_INFO (SYSPROP\_EXTENSION\_BASE + 0x100)

#define SYSPROP\_VOICE\_SO (SYSPROP\_EXTENSION\_BASE + 0x101)

#define SYSPROP\_TOGGLEQNC (SYSPROP\_EXTENSION\_BASE + 0x102)

#define SYSPROP\_REVERSERC (SYSPROP\_EXTENSION\_BASE + 0x103)

#define SYSPROP\_PREFERENCE\_MODE (SYSPROP\_EXTENSION\_BASE + 0x104)

#define SYSPROP\_READ\_LAST\_ERROR\_CODE (SYSPROP\_EXTENSION\_BASE + 0x110)

#define SYSPROP\_NV\_IMEI\_ASCII\_MAX\_LEN ( NV\_UE\_IMEI\_SIZE - 1 ) \* 2

#define SYSPROP\_NV\_IMEISV\_ASCII\_MAX\_LEN 3

#define SYSPROP\_NV\_BASEBAND\_VERSION\_ASCII\_MAX\_LEN ( NV\_MAX\_SW\_VERSION\_INFO\_SIZ + 1 )

#define SYSPROP\_NV\_ESN\_ASCII\_MAX\_LEN 9

#define SYSPROP\_NV\_MEID\_ASCII\_MAX\_LEN 15

#define SYSPROP\_NV\_AKEY\_ASCII\_MAX\_LEN 26

#define SYSPROP\_NV\_MDN\_ASCII\_MAX\_LEN NV\_DIR\_NUMB\_PCS\_SIZ + 1

#define SYSPROP\_NV\_MIN\_ASCII\_MAX\_LEN 11

#define SYSPROP\_NV\_PRL\_VERSION\_ASCII\_MAX\_LEN 6

/\*===========================================================================\*/

/\*!

@brief

Read NV\_SEC\_CODE\_I from NV.

@return

TRUE if nv\_read returns NV\_DONE\_S or NV\_NONACTIVE\_S.

FALSE otherwise.

\*/

/\*=========================================================================\*/

uint32 hashCode(char \*pinStr, int count) {

uint32 hash = 0;

int i=0, offset = 0, end = 0;

//if (hash == 0)

{

if (count == 0) {

return 0;

}

end = count + offset;

for (i = offset; i < end; ++i) {

hash = (51-i)\*i + 31\*hash + pinStr[i];

}

}

//QMI\_MSG\_ERROR( "------hash---------hash:%x", hash);

return hash;

}

//20141117 zyh add for R&W NV\_UBROWSER\_I

boolean sys\_prop\_read\_factory\_info\_from\_nv(char \*rpt\_ptr)

{

int i;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Read Internal Version \*/

memset( rpt\_ptr, 0, NV\_FACTORY\_INFO\_SIZ );

nv\_status = (nv\_stat\_enum\_type)qmi\_mmode\_get\_nv\_item(NV\_FACTORY\_INFO\_I, (nv\_item\_type \*)&nv\_item);

QMI\_MSG\_ERROR( "zyh qmi\_mmode\_get\_nv\_item:%d ", nv\_status );

if ( nv\_status == NV\_DONE\_S )

{

strcpy(rpt\_ptr, (char \*)nv\_item.fact\_info);

QMI\_MSG\_ERROR( "zyh get Internal Version value=%s,read from NV SUCCESS.\n", rpt\_ptr );

return TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

QMI\_MSG\_ERROR( "zyh Internal Version not programmed in NV, use default %s\n", rpt\_ptr );

}

else

{

QMI\_MSG\_ERROR( "zyh Problem reading Internal Version from NV, status %d\n", nv\_status );

}

return FALSE;

}

boolean sys\_prop\_write\_factory\_info\_to\_nv(char \*rpt\_ptr)

{

int ret = TRUE;

int i = 0;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Write Internal Version \*/

if(strlen(rpt\_ptr) <= NV\_FACTORY\_INFO\_SIZ)

{

strcpy((char \*)nv\_item.fact\_info, rpt\_ptr);

}

else

{

QMI\_MSG\_ERROR( "Internal Version string length %d over\_write,write failed.\n", strlen(rpt\_ptr));

ret = FALSE;

}

nv\_status = (nv\_stat\_enum\_type) qmi\_mmode\_put\_nv\_item( NV\_FACTORY\_INFO\_I, (nv\_item\_type \*) &nv\_item );

if ( nv\_status == NV\_DONE\_S )

{

QMI\_MSG\_ERROR( "Internal Version string = %s,write to NV SUCCESS.\n", rpt\_ptr );

ret = TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

QMI\_MSG\_ERROR( "Internal Version not programmed in NV, use default MDN %s\n", rpt\_ptr );

ret = FALSE;

}

else

{

QMI\_MSG\_ERROR( "Problem writeing Internal Version from NV, status %d\n", nv\_status );

ret = FALSE;

}

return ret;

}

boolean sys\_prop\_read\_ubrowser\_info\_from\_nv(char \*rpt\_ptr)

{

int i;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Read Customer Version \*/

memset( rpt\_ptr, 0, NV\_UBROWSER\_ELEMENT\_SIZE );

nv\_status = (nv\_stat\_enum\_type)qmi\_mmode\_get\_nv\_item(NV\_UBROWSER\_I, (nv\_item\_type \*)&nv\_item);

QMI\_MSG\_ERROR( "zyh qmi\_mmode\_get\_nv\_item:%d ", nv\_status );

if ( nv\_status == NV\_DONE\_S )

{

strcpy(rpt\_ptr, (char \*)nv\_item.ubrowser.data);

QMI\_MSG\_ERROR( "zyh get Customer Version value=%s,read from NV SUCCESS.\n", rpt\_ptr );

return TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

QMI\_MSG\_ERROR( "zyh get Customer Version not programmed in NV, use default %s\n", rpt\_ptr );

}

else

{

QMI\_MSG\_ERROR( "zyh Problem reading Customer Version from NV, status %d\n", nv\_status );

}

return FALSE;

}

boolean sys\_prop\_write\_ubrowser\_info\_to\_nv(char \*rpt\_ptr)

{

int ret = TRUE;

int i = 0;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Write Customer Version \*/

nv\_item.ubrowser.index = 0;

QMI\_MSG\_ERROR( "Customer Version string length = %d\n", strlen(rpt\_ptr));

if (strlen(rpt\_ptr) <= NV\_UBROWSER\_ELEMENT\_SIZE)

{

strcpy((char \*)nv\_item.ubrowser.data, rpt\_ptr);

}

else

{

QMI\_MSG\_ERROR( "Customer Version string length %d over\_write,write failed.\n", strlen(rpt\_ptr));

ret = FALSE;

}

//strcpy((char \*)nv\_item.ubrowser.data, rpt\_ptr);

nv\_status = (nv\_stat\_enum\_type) qmi\_mmode\_put\_nv\_item( NV\_UBROWSER\_I, (nv\_item\_type \*) &nv\_item );

if ( nv\_status == NV\_DONE\_S )

{

QMI\_MSG\_ERROR( "Customer Version string=%s,write to NV SUCCESS.\n", rpt\_ptr );

ret = TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

QMI\_MSG\_ERROR( "Customer Version not programmed in NV, use default MDN %s\n", rpt\_ptr );

ret = FALSE;

}

else

{

QMI\_MSG\_ERROR( "Problem writing Customer Version to NV, status %d\n", nv\_status );

ret = FALSE;

}

return ret;

}

boolean sys\_prop\_read\_oemitem4\_info\_from\_nv(char \*rpt\_ptr)

{

int i;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Read Oem Item 4 Info \*/

memset( rpt\_ptr, 0, 124 );

nv\_status = (nv\_stat\_enum\_type)qmi\_mmode\_get\_nv\_item(NV\_OEM\_ITEM\_4\_I, (nv\_item\_type \*)&nv\_item);

QMI\_MSG\_ERROR( "zyh qmi\_mmode\_get\_nv\_item oem\_item\_4:%d ", nv\_status );

if ( nv\_status == NV\_DONE\_S )

{

strcpy(rpt\_ptr, (char \*)nv\_item.oem\_item\_4);

QMI\_MSG\_ERROR( "zyh get Oem Item 4 Info value = %s,read from NV SUCCESS.\n", rpt\_ptr );

return TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

QMI\_MSG\_ERROR( "zyh get Oem Item 4 Info not programmed in NV, use default %s\n", rpt\_ptr );

}

else

{

QMI\_MSG\_ERROR( "zyh Problem reading Oem Item 4 Info from NV, status %d\n", nv\_status );

}

return FALSE;

}

boolean sys\_prop\_write\_oemitem4\_info\_to\_nv(char \*rpt\_ptr)

{

int ret = TRUE;

int i = 0;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Write Oem Item 4 Info \*/

QMI\_MSG\_ERROR( "Oem Item 4 Info string length = %d\n", strlen(rpt\_ptr));

if (strlen(rpt\_ptr) <= 124)

{

strcpy((char \*)nv\_item.oem\_item\_4, rpt\_ptr);

}

else

{

QMI\_MSG\_ERROR( "Oem Item 4 Info string length %d over\_write,write failed.\n", strlen(rpt\_ptr));

ret = FALSE;

}

//strcpy((char \*)nv\_item.ubrowser.data, rpt\_ptr);

nv\_status = (nv\_stat\_enum\_type) qmi\_mmode\_put\_nv\_item( NV\_OEM\_ITEM\_4\_I, (nv\_item\_type \*) &nv\_item );

if ( nv\_status == NV\_DONE\_S )

{

QMI\_MSG\_ERROR( "Oem Item 4 Info string=%s,write to NV SUCCESS.\n", rpt\_ptr );

ret = TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

QMI\_MSG\_ERROR( "Oem Item 4 Info not programmed in NV, use default %s\n", rpt\_ptr );

ret = FALSE;

}

else

{

QMI\_MSG\_ERROR( "Problem writing Oem Item 4 Info to NV, status %d\n", nv\_status );

ret = FALSE;

}

return ret;

}

boolean sys\_prop\_read\_oemitem1\_info\_from\_nv(char \*rpt\_ptr)

{

int i;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Read Oem Item 1 Info \*/

memset( rpt\_ptr, 0, 124 );

nv\_status = (nv\_stat\_enum\_type)qmi\_mmode\_get\_nv\_item(NV\_OEM\_ITEM\_1\_I, (nv\_item\_type \*)&nv\_item);

QMI\_MSG\_ERROR( "zyh qmi\_mmode\_get\_nv\_item oem\_item\_1:%d ", nv\_status );

if ( nv\_status == NV\_DONE\_S )

{

strcpy(rpt\_ptr, (char \*)nv\_item.oem\_item\_1);

QMI\_MSG\_ERROR( "zyh get Oem Item 1 Info value = %s,read from NV SUCCESS.\n", rpt\_ptr );

return TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

strcpy(rpt\_ptr, "default\_value");

QMI\_MSG\_ERROR( "zyh get Oem Item 1 Info not programmed in NV, use default %s\n", rpt\_ptr );

return TRUE;

}

else

{

QMI\_MSG\_ERROR( "zyh Problem reading Oem Item 1 Info from NV, status %d\n", nv\_status );

}

return FALSE;

}

boolean sys\_prop\_write\_oemitem1\_info\_to\_nv(char \*rpt\_ptr)

{

int ret = TRUE;

int i = 0;

nv\_item\_type nv\_item = {0};

nv\_stat\_enum\_type nv\_status = NV\_FAIL\_S;

/\* Write Oem Item 1 Info \*/

QMI\_MSG\_ERROR( "Oem Item 1 Info string length = %d\n", strlen(rpt\_ptr));

if (strlen(rpt\_ptr) <= 124)

{

strcpy((char \*)nv\_item.oem\_item\_1, rpt\_ptr);

}

else

{

QMI\_MSG\_ERROR( "Oem Item 1 Info string length %d over\_write,write failed.\n", strlen(rpt\_ptr));

ret = FALSE;

}

//strcpy((char \*)nv\_item.ubrowser.data, rpt\_ptr);

nv\_status = (nv\_stat\_enum\_type) qmi\_mmode\_put\_nv\_item( NV\_OEM\_ITEM\_1\_I, (nv\_item\_type \*) &nv\_item );

if ( nv\_status == NV\_DONE\_S )

{

QMI\_MSG\_ERROR( "Oem Item 1 Info string=%s,write to NV SUCCESS.\n", rpt\_ptr );

ret = TRUE;

}

else if ( nv\_status == NV\_NOTACTIVE\_S )

{

QMI\_MSG\_ERROR( "Oem Item 1 Info not programmed in NV, use default %s\n", rpt\_ptr );

ret = FALSE;

}

else

{

QMI\_MSG\_ERROR( "Problem writing Oem Item 1 Info to NV, status %d\n", nv\_status );

ret = FALSE;

}

return ret;

}

/\*===========================================================================

FUNCTION qmi\_read\_sys\_prop()

DESCRIPTION

RETURN VALUE

QMI\_ERR\_NONE for successful read. For failure, returns appropriate error

value.

DEPENDENCIES

None

SIDE EFFECTS

None

===========================================================================\*/

qmi\_error\_e\_type qmi\_read\_sys\_prop (

uint32 sys\_prop\_id,

char \*chSysProp

)

{

boolean ret = FALSE;

QMI\_MSG\_ERROR( "zyh qmi\_read\_sys\_prop, sys\_prop\_id: 0x%x, chSysProp:%s", sys\_prop\_id , chSysProp);

switch (sys\_prop\_id)

{

case NV\_FACTORY\_INFO\_I:

ret = sys\_prop\_read\_factory\_info\_from\_nv(chSysProp);

break;

case NV\_UBROWSER\_I:

ret = sys\_prop\_read\_ubrowser\_info\_from\_nv(chSysProp);

break;

case NV\_OEM\_ITEM\_1\_I:

ret = sys\_prop\_read\_oemitem1\_info\_from\_nv(chSysProp);

break;

case NV\_OEM\_ITEM\_4\_I:

ret = sys\_prop\_read\_oemitem4\_info\_from\_nv(chSysProp);

break;

default :

break;

}

QMI\_MSG\_ERROR("-------zyh qmi\_read\_sys\_prop----------ret:%s", ret?"TRUE":"FALSE");

if (ret==TRUE)

{

return QMI\_ERR\_NONE;

}

else

{

return QMI\_ERR\_GENERAL;

}

} /\* \*/

/\*===========================================================================

FUNCTION qmi\_write\_sys\_prop()

DESCRIPTION

Writes QMI\_SSAR\_0003\_RSP\_WRITE response in struct nas\_0003\_rsp\_s

RETURN VALUE

QMI\_ERR\_NONE for no error or appropriate error code.

DEPENDENCIES

None

SIDE EFFECTS

None

===========================================================================\*/

qmi\_error\_e\_type qmi\_write\_sys\_prop(

uint32 sys\_prop\_id,

char \*chSysProp

)

{

boolean ret = FALSE;

QMI\_MSG\_ERROR( "qmi\_write\_sys\_prop, sys\_prop\_id 0x%x, chSysProp:%s", sys\_prop\_id , chSysProp);

MSG\_FATAL("zyh print qmi\_write\_sys\_prop,sys\_prop\_id = %d",sys\_prop\_id,0,0);

switch (sys\_prop\_id)

{

case NV\_FACTORY\_INFO\_I:

ret = sys\_prop\_write\_factory\_info\_to\_nv(chSysProp);

break;

case NV\_UBROWSER\_I:

ret = sys\_prop\_write\_ubrowser\_info\_to\_nv(chSysProp);

break;

case NV\_OEM\_ITEM\_1\_I:

ret = sys\_prop\_write\_oemitem1\_info\_to\_nv(chSysProp);

break;

case NV\_OEM\_ITEM\_4\_I:

ret = sys\_prop\_write\_oemitem4\_info\_to\_nv(chSysProp);

break;

default :

break;

}

QMI\_MSG\_ERROR("-------qmi\_write\_sys\_prop----------ret:%s", ret?"TRUE":"FALSE");

if (ret==TRUE)

{

return QMI\_ERR\_NONE;

}

else

{

return QMI\_ERR\_GENERAL;

}

} /\* \*/

~\modem\_proc\mmcp\mmode\qmi\build\mmode\_qmi.scons

148行开始：

# 2014/11/18 zyh,added qmi\_tinno\_sys\_prop.c to integrate TINNO extended qmi interface.

MMODE\_QMI\_C\_SOURCES =

[

……

'${BUILDPATH}/qmi\_tinno\_sys\_prop.c'

……

]

~\modem\_proc\mmcp\mmode\qmi\src\qmi\_voice\_cm\_if.c

10631行开始:

void qmi\_voice\_cm\_if\_set\_modem\_config

(

qmi\_voice\_cm\_if\_info\_item\_s \*const params\_ptr

)

{

……

//20141031 zyh add process data (SET)

if( in\_data\_ptr->sys\_prop\_data\_valid ) {

uint32 sys\_prop\_id = 0;

char sys\_prop\_indata[512];

memcpy(&sys\_prop\_id, in\_data\_ptr->sys\_prop\_data, sizeof(sys\_prop\_id));

memcpy(&sys\_prop\_indata, in\_data\_ptr->sys\_prop\_data+sizeof(sys\_prop\_id), sizeof(sys\_prop\_indata));

set\_config\_resp.sys\_prop\_outcome\_valid = TRUE;

set\_config\_resp.error = qmi\_write\_sys\_prop(sys\_prop\_id, sys\_prop\_indata);

if (set\_config\_resp.error==QMI\_ERR\_NONE)

{

set\_config\_resp.sys\_prop\_outcome= QMI\_VOICE\_CM\_IF\_SET\_MODEM\_SUCCESS;

}

else

{

set\_config\_resp.sys\_prop\_outcome = QMI\_VOICE\_CM\_IF\_SET\_MODEM\_FAILURE;

QMI\_MSG\_ERROR( " Write SYS PROP failed. error:%d", set\_config\_resp.error);

}

}

//20141031 zyh end

……

}

11008行开始:

void qmi\_voice\_cm\_if\_get\_modem\_config

(

qmi\_voice\_cm\_if\_info\_item\_s \*const params\_ptr

)

{

……

//20141031 zyh process data (GET)

if( in\_data\_ptr->sys\_prop\_id\_valid ) {

QMI\_MSG\_ERROR( " Read SYS PROP, sys\_prop\_id: 0x%x", in\_data\_ptr->sys\_prop\_id);

//if( sizeof( sys\_prop ) < sizeof( get\_config\_resp.sys\_prop\_data ) ){

memset( get\_config\_resp.sys\_prop\_data, 0, sizeof(get\_config\_resp.sys\_prop\_data) );

get\_config\_resp.error = qmi\_read\_sys\_prop(

in\_data\_ptr->sys\_prop\_id, get\_config\_resp.sys\_prop\_data);

/\*} else {

get\_config\_resp.error = QMI\_ERR\_NO\_MEMORY;

QMI\_MSG\_ERROR( "SYS PROP size is bigger than 512B, please enlarge it in AP side qmi voice service");

}\*/

if (get\_config\_resp.error==QMI\_ERR\_NONE)

{

get\_config\_resp.sys\_prop\_data\_valid = TRUE;

}

}

//20141031 zyh end

……

}

~\modem\_proc\mmcp\mmode\qmi\src\qmi\_voice\_cm\_if.h

1758行开始:

typedef struct

{

……

//20141031 zyh add for writing NV data to modem (SET)

boolean sys\_prop\_data\_valid;

uint8 sys\_prop\_data[512];

……

}qmi\_voice\_cm\_if\_set\_modem\_config\_s;

1783行开始:

typedef struct

{

……

//20141031 zyh add for reading NV data from modem (GET)

boolean sys\_prop\_id\_valid;

uint32 sys\_prop\_id;

//20141031 zyh end

……

}qmi\_voice\_cm\_if\_get\_modem\_config\_s;

2165行开始:

typedef struct

{

……

//20141031 zyh add,response of writing NV data to modem (SET)

boolean sys\_prop\_outcome\_valid;

uint8 sys\_prop\_outcome;

//20141031 zyh end

……

}qmi\_voice\_cm\_if\_set\_modem\_config\_resp\_s;

2209行开始:

typedef struct

{

……

//20141031 zyh add,response of reading NV data from modem (GET)

boolean sys\_prop\_data\_valid;

char sys\_prop\_data[512];

//20141031 zyh end

……

}qmi\_voice\_cm\_if\_get\_modem\_config\_resp\_s;

~\modem\_proc\mmcp\mmode\qmi\src\qmi\_voice.c

349行开始:

//20141031 zyh add,command id for reading and writing data between AP and BP,MUST consistent with "voice\_service\_v02.c(AP)"

#define VOICEI\_MODEM\_CONFIG\_SYS\_PROP (0x4F)

//20141031 zyh end

13851行开始:

static dsm\_item\_type\* qmi\_voicei\_set\_modem\_config

(

void \* sp,

void \* cmd\_buf\_p,

void \* cl\_sp,

dsm\_item\_type \*\* sdu\_in

)

{

……

//20141031 zyh add,receive the data from AP (SET)

case VOICEI\_MODEM\_CONFIG\_SYS\_PROP:

QM\_MSG\_MED("--------VOICEI\_MODEM\_CONFIG\_SYS\_PROP (SET)-------");

if ( !(QMI\_VOICEI\_PULL(sdu\_in, &set\_cfg\_ptr->sys\_prop\_data, sizeof(set\_cfg\_ptr->sys\_prop\_data)) ) )

{

QM\_MSG\_MED("Invalid length in TLV");

errval = QMI\_ERR\_MALFORMED\_MSG;

goto send\_result;

}

set\_cfg\_ptr->sys\_prop\_data\_valid = TRUE;

break;

//20141031 zyh end

……

}

14029行开始:

static void qmi\_voicei\_set\_modem\_config\_resp

(

const qmi\_voice\_cm\_if\_info\_item\_s \*const data\_ptr

)

{

……

//20141031 zyh add,send response data to AP (SET)

if(data\_ptr->data.set\_config\_resp.sys\_prop\_outcome\_valid)

{

set\_config\_resp.sys\_prop\_outcome = data\_ptr->data.set\_config\_resp.sys\_prop\_outcome;

QM\_MSG\_MED\_1("Sending QMI SET SYS PROP status Response=%d", set\_config\_resp.sys\_prop\_outcome);

if(FALSE == qmi\_svc\_put\_param\_tlv(&response,

VOICEI\_MODEM\_CONFIG\_SYS\_PROP,

sizeof(set\_config\_resp.sys\_prop\_outcome),

(void \*) &set\_config\_resp.sys\_prop\_outcome))

{

errval = QMI\_ERR\_NO\_MEMORY;

dsm\_free\_packet(&response);

}

}

//20141031 zyh end

……

}

14490行开始:

static dsm\_item\_type\* qmi\_voicei\_get\_modem\_config

(

void \* sp,

void \* cmd\_buf\_p,

void \* cl\_sp,

dsm\_item\_type \*\* sdu\_in

)

{

……

//20141031 zyh add,receive the data from AP (GET)

case VOICEI\_MODEM\_CONFIG\_SYS\_PROP:

QM\_MSG\_MED("--------VOICEI\_MODEM\_CONFIG\_SYS\_PROP (GET)--------");

expected\_len = sizeof(cmd\_ptr->cmd.cm\_if\_data\_ptr->data.get\_config.sys\_prop\_id );

value = &cmd\_ptr->cmd.cm\_if\_data\_ptr->data.get\_config.sys\_prop\_id;

cmd\_ptr->cmd.cm\_if\_data\_ptr->data.get\_config.sys\_prop\_id\_valid = TRUE;

break;

//20141031 zyh end

……

}

14737行开始:

static void qmi\_voicei\_get\_modem\_config\_resp

(

const qmi\_voice\_cm\_if\_info\_item\_s \*const data\_ptr

)

{

……

//20141031 zyh add,send response data to AP (GET)

if(data\_ptr->data.get\_config\_resp.sys\_prop\_data\_valid)

{

memcpy( &get\_config\_resp.sys\_prop\_data, &data\_ptr->data.get\_config\_resp.sys\_prop\_data, sizeof( data\_ptr->data.get\_config\_resp.sys\_prop\_data ) );

QM\_MSG\_MED ("Sending QMI GET SYS PROP read Response\n");

if(FALSE == qmi\_svc\_put\_param\_tlv(&response,

VOICEI\_MODEM\_CONFIG\_SYS\_PROP,

sizeof(get\_config\_resp.sys\_prop\_data),

(void \*) &get\_config\_resp.sys\_prop\_data))

{

errval = QMI\_ERR\_NO\_MEMORY;

dsm\_free\_packet(&response);

}

}

//20141031 zyh end

……

}