ETSI TS 126 268 V10.0.0 (2011-04)

Technická špecifikácia

Digitálny bunkový telekomunikačný systém (fáza 2+);
Univerzálny mobilný telekomunikačný systém (UMTS);
Prenos dát eCall; Riešenie vnútropásmového modemu; Referenčný kód ANSI-C
(3GPP TS 26.268 verzia 10.0.0 časť 10)

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
eCall data transfer; In-band modem solution; ANSI-C reference code
(3GPP TS 26.268 version 10.0.0 Release 10)

Referenčné číslo

RTS/TSGS-0426268va00

Kľúčové slová

GSM, UMTS

ETSI

650 Route des Lucioles
F–06921 Sophia Antipolis Cedex – France

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Neziskové združenie registrované
na podprefektúre de Grasse (06) N° 7803/88

Dôležité upozornenie

Jednotlivé kópie tohto dokumentu možno stiahnuť z

http://pda.etsi.org

Tento dokument môže byť dostupný vo viacerých elektronických verziách alebo v tlačenej forme. V prípade existujúceho alebo viditeľného rozdielu v obsahu medzi takýmito verziami je referenčnou verziou verzia v prenosnom dokumentovom formáte (Portable Document Format – PDF).

V prípade sporu je referenčným výtlačok vytlačený na tlačiarni ETSI z verzie PDF uchovávanej na určenom sieťovom serveri sekretariátu ETSI.

Používatelia tohto dokumentu by mali brať do úvahy, že dokument môže byť revidovaný alebo sa môže zmeniť jeho postavenie. Informácie o postavení tohto dokumentu a ďalších dokumentov ETSI sú dostupné na

http://portal.etsi.org/tb/status/status.asp

Ak nájdete v tomto dokumente chyby, svoje pripomienky zašlite na

http://portal.etsi.org/chaircor/ETSI_support.asp

Oznam o autorských právach

Obsah

Práva duševného vlastníctva

Práva duševného vlastníctva, ktoré majú alebo môžu mať zásadný význam pre tento dokument, mohli byť oznámené organizácii ETSI. Informácie o týchto zásadných právach duševného vlastníctva, ak existujú, sú pre členov i nečlenov ETSI verejne dostupné a môžu ich nájsť v dokumente ETSI SR 000 314 s názvom Práva duševného vlastníctva (IPR), ktorý možno získať na sekretariáte ETSI. Najnovšie znenie je dostupné na serveri ETSI (http://webapp.etsi.org/IPR/home.asp).

V súlade so svojou politikou v oblasti práv duševného vlastníctva ETSI nevyhľadáva ani neskúma nijaké práva duševného vlastníctva. Neposkytuje ani záruku týkajúcu sa existencie iných IPR, ktoré nie sú uvedené v dokumente ETSI SR 000 314 (alebo v jeho aktualizovaných vydaniach na serveri ETSI), ktoré majú, môžu mať, alebo môžu nadobudnúť zásadný význam pre predkladaný dokument.

Predhovor

Technickú špecifikáciu (TS) pripravil projekt partnerstva tretej generácie (3GPP) ETSI.

Dokument môže odkazovať na technické špecifikácie alebo správy s využitím ich identít 3GPP, identít UMTS alebo identít GSM. Tieto sa majú interpretovať ako odkazy na súvisiace vydania ETSI.

Krížové odkazy medzi identitami GSM, UMTS, 3GPP a ETSI sa môžu nájsť na http://webapp.etsi.org/key/queryform.asp.

Predslov

Dokument vytvoril projekt partnerstva tretej generácie (3GPP).

Obsah dokumentu je predmetom pokračujúcich prác v TSG a môže sa zmeniť pri nasledujúcom formálnom schvaľovaní TSG. Ak TSG zmení obsah dokumentu, TSG ho znovu vydá so zmeneným dátumom vydania a zvýšeným číslom verzie:

Verzia x.y.z

kde:

x prvá číslica:

1 predložené do TSG na informáciu;

2 predložené do TSG na schválenie;

3 alebo viac označuje dokument schválený TSG v procese zmeny.

y druhá číslica sa zvyšuje pri všetkých podstatných zmenách, napríklad technické rozšírenie, opravy, modernizácia, atď.

z tretia číslica sa zvyšuje keď sa do dokumentu zahrnuli iba editorské zmeny.

1. Predmet

Technická špecifikácia (TS) obsahuje elektronickú kópiu kódu ANSI-C riešenia vnútropásmového modemu eCall na spoľahlivý prenos dát MSD z IVS do PSAP cez hovorový kanál bunkových sietí. Kód ANSI–C je potrebný na bitovoexaktnú implementáciu modemu IVS a modemu PSAP, ktorú uvádza technická špecifikácia 3GPP TS 26.267 [1].

2. Normatívne referenčné dokumenty

Nasledujúce dokumenty obsahujú ustanovenia, ktoré cez odkazy v texte tvoria ustanovenia tohto dokumentu.

- Odkazy sú buď špecifikované (určené dátumom vydania, číslom edície, číslom verzie atď.), alebo nešpecifikované.
- Pri nešpecifikovanom odkaze sa neskoršie revízie neaplikujú.
- Pri nešpecifikovaných odkazoch sa aplikuje najnovšia verzia. V prípade odkazu na dokument 3GPP (vrátane dokumentu GSM) nešpecifikovaný odkaz implicitne odkazuje na najnovšiu verziu daného dokumentu v tom istom vydaní ako tento dokument.

[1] 3GPP TS 26.267: "eCall Data Transfer; In-band modem solution; General Description".

Pozri aj odkazy v 3GPP TS 26.267 [1].

3. Skratky

V dokumente sa používajú skratky:

ACK	ACKnowledgement	potvrdenie
ANSI	American National Standards Institute	Americký národný úrad pre normalizáciu
CRC	Cyclic Redundancy Check	kontrola cyklickým redundantným kódom
FEC	Forward Error Correction	korekcia chýb v doprednom smere
GSM	Global System for Mobile communications	globálny systém mobilných komunikácií
HARQ	Hybrid Automatic Repeat-reQuest	hybridná automatická žiadosť o opakovanie
I/O	Input/Output	vstup/výstup
IVS	In-Vehicle System	systém vo vozidle
MSD	Minimum Set of Data	minimálny súbor dát
NACK	Negative ACKnowledgement	negatívne potvrdenie
PCM	Pulse Code Modulation	impulzová kódová modulácia
PSAP	Public Safety Answering Point	kontaktné stredisko integrovaného záchranného systému
RAM	Random Access Memory	pamäť s náhodným prístupom
ROM	Read Only Memory	permanentná pamäť
RX	Receive	príjem
TX	Transmit	vysielanie

4 Štruktúra kódu C

Kapitola dáva prehľad o štruktúre presného bitového kódu C a poskytuje prehľad obsahu a organizácie kódu C, ktorý je s týmto dokumentom zviazaný.

Kód C sa verifikoval na týchto systémoch:

- Windows XP SP2 a Microsoft Visual Studio V8.0;
- Linux (Suse Linux) s použitím prekladačov gcc v3.4.2 a v4.1.2.

4.1 Obsah zdrojového kódu C

Distribuované súbory s príponou "c" obsahujú zdrojový kód a súbory s príponou "h" sú súbory záhlavia.

Ďalšie vysvetlenie k súborom sa nachádza v súbore Readme.txt file, ktorý sa čiastočne reprodukuje:

```
Obsah balíka

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


adresár 'ecall':

    Obsahuje úplný referenčný zdrojový kód eCall ANSI C s pevnou rádovou čiarkou.


    modem_ivs.c     : implementácia modemu IVS na najvyššej úrovni

    modem_psap.c    : implementácia modemu PSAP na najvyššej úrovni


    modemx.h        : súbor záhlavia pre modem_ivs.c a modem_psap.c


    ecall_defines.h : voľby kompilačného času a konštanty predspracovania


    ecall_control.h : súbor záhlavia riadenia obsluhy správ

    ecall_fec.h     : súbor záhlavia kódovača a dekódovača FEC

    ecall_modem.h   : súbor záhlavia modulátora a demodulátora

    ecall_sync.h    : súbor záhlavia synchronizácie

    ecall_rom.h     : súbor záhlavia dát ROM


    ecall_control.c : riadenie obsluhy správ

    ecall_fec.c     : kódovač a dekódovač FEC
```

```
    ecall_modem.c   : modulátor and demodulátor

    ecall_sync.c    : synchronizácia

    ecall_rom.c     : dáta ROM
```

adresár 'test_setup':

obsahuje simulačný rámec softvéru eCall, ktorý sa skompiluje a bude bežať v systémoch MS Windows.

adresár 'test_vec':

```
    Obsahuje binárne dáta PCM 104 súborov) a záznamy portov prijímača/vysielača

    vo formáte ASCII (104 súborov) na skúšanie modemov eCall IVS a PSAP.
```

Formát PCM je šestnásťbitový so znamienkom, prvý bajt najmenej významný, pri frekvencii vzorkovania 8 kHz.

Dátové súbory zodpovedajú 26 skúšobným prípadom a vytvorili sa zo simulačného rámca eCall.

```
    campaign_short.txt    : konfiguračný súbor pre 26 skúšobných prípadov


    pcmdlout<index>.pcm   : výstup dát PCM z hlasového kódovača DL = vstup do IVS

    pcmulout<index>.pcm   : výstup dát PCM z hlasového kódovača UL = vstup do PSAP


    pcmdlin<index>.pcm    : skúšobné vektory pre výstup modemu PSAP

    pcmulin<index>.pcm    : skúšobné vektory pre výstup modemu IVS


    portivsrx<index>.txt  : skúšobné vektory pre záznamy portov IVS (prijímač)

    portivstx<index>.txt  : skúšobné vektory pre záznamy portov IVS (vysielač)


    portpsaprx<index>.txt : skúšobné vektory pre záznamy portov PSAP (prijímač)

    portpsaptx<index>.txt : skúšobné vektory pre záznamy portov PSAP (vysielač)
```

standalone.c

Hlavný (main)() obal na beh modemu IVS alebo PSAP na vopred uložených súboroch PCM alebo záznamoch portov prijímača/vysielača. Na získanie zoznamu volieb príkazového riadka sa vyvolá zodpovedajúci vykonateľný súbor s voľbou '-h' (help).

```
standalone.h

    súbor záhlavia pre standalone.c



Makefile.win

    NMAKE Vytvárací súbor pre Microsoft Visual Studio 2005 a novšie verzie:
```

vytvára 'standalone.exe' zo standalone.c a zdrojov eCall, voľby vytvárania sú RELEASE a DEBUG.

```
Makefile.glx

    Vytvárací súbor pre GNU Linux s použitím gcc.

    Vytvára 'standalone' zo standalone.c a zdrojov eCall,

    voľby vytvárania sú RELEASE a DEBUG.



verify.bat

    Dávkový súbor Windows.
```

vykonáva 'standalone.exe' v šiestich odlišných režimoch modemu na 26 skúšobných prípadoch, obsiahnutých v adresári 'test_vec' a vykonáva porovnanie skúšobného vektora s príslušným výstupom PCM a zaznamenanými dátami portu.

```
verify.sh

    scenár pre interpretátor príkazov Linux (shell script)

    Vykonáva 'standalone' v režime '-m ivs' a '-m psap' na 26 skúšobných prípadoch
```

(adresár 'pcm'); vykonáva porovnanie skúšobného vektora s príslušnými dátami PCM na výstupe modemu.

4.2 Vykonávanie programu

Vysvetlenie kompilácie kódu a vykonávania sa nachádza v súbore readme.txt, ktorý sa tu čiastočne uvádza:

```
Začíname

--------

3GPP TS 26.268 poskytuje zdrojový kód modemu eCall, rámec softvérovej simulácie a
samostatný obal, ktorý umožňuje modemu IVS alebo PSAP prevádzku s vopred uloženými
referenčnými dátami.

Nasledujúce funkcie reprezentujú rozhranie modemu eCall a opierajú sa o príslušné
implementácie prijímača a vysielača každého modemu:

    * void IvsReset(const Ord8 *msd, int length);

    * void IvsProcess(Int16 *pcm);
```

* void IvsSendStart(void);

* void PsapReset(void);

* void PsapProcess(Int16 *pcm);

* void PsapSendStart(void);

* void PsapSendHlack(const Ord8 data);

Externá aplikácia musí dodatočne implementovať funkcie spätného volania:

* void IvsCatchEvent(IvsEvent ie);

* void IvsReceiveAck(void);

* void IvsReceiveHlack(const Ord8 data);

* void PsapCatchEvent(PsapEvent pe);

* void PsapReceiveMsd(const Ord8 *msd, int length);

* void Abort(const char *format, ...);

* void LogInfo(const char *format, ...);

IvsCatchEvent a PsapCatchEvent informujú o dôležitých modemových udalostiach a môžu sa použiť na vykonanie činností, ako sú umlčanie alebo zrušenie mlčania hlasového kanála.

Iné funkcie spätného volania označujú:

* IvsReceiveAck : príjem ACK nižšej vrstvy,

* IvsReceiveHlack : príjem správy HLACK,

* PsapReceiveMsd : úspešný príjem MSD.

Abort a LogInfo sa majú implementovať v uvedenom poradí chybu funkcie s premenlivým počtom argumentov a obsluhu tlače záznamu. Pre príklady implementácie všetkých funkcií spätných volaní pozri standalone.c.

Na simuláciu v reálnom čase cez hlasové kódovače 3GPP FR a AMR a na záznam dát PCM vo forme vstupu do samostatného obalu, sa zdroje eCall musia integrovať do simulačného rámca; adresár 'test_setup' obsahuje jeden takýto rámec tak, ako sa použil vo výberových skúškach 3GPP.

Na kompiláciu a beh kódu modemu eCall je potrebné sledovať uvedené inštrukcie. Na skúšanie kódu sa poskytli dva dávkové súbory:

* verify.bat : systémy MS Windows

* verify.sh : systémy Linux

Pre každý z 26 skúšobných prípadov z campaign_short.txt v adresári 'test_vec' sa spustí samostatný obal v šiestich odlišných režimoch modemu (tri režimy IVS a tri režimy PSAP).

Výsledné súbory PCM a záznamov portov sa nakoniec porovnajú so skúšobnými vektormi v adresári 'test_vec'.

V režimoch 'psap' a 'psaprx' sa má na záver každého skúšobného prípadu úspešne prijať správa MSD.

Kompilácia kódu

Systémy MS Windows

Kompilácia predpokladá inštalovanie MS Visual Studio 2005 alebo novšej verzie.

Na nastavenie premenných prostredia na vytvorenie je potrebné spustiť 'vcvars32.bat', ktorý sa má nachádzať v podadresári 'bin' inštalácie VC.

Na vytvorenie standalone.exe zo standalone.c a zdrojov eCall (alebo na vyčistenie) spustite

nmake /f Makefile.win,

nmake /f Makefile.win clean,

Zdrojový kód sa má skompilovať bez akýchkoľvek chýb alebo upozornení.

Spustí sa'verify.bat' na overenie vykonávania vo vzťahu k skúšobným vektorom.

Systémy GNU Linux

Kompilácia pod systémom Linux sa má skúšať s:

* GNU Make verzia 3.81,

* gcc verzia 4.1.3 a 4.2.4.

Na vytvorenie samostatne vykonateľného a čistiaceho súboru sa použije:

make -f Makefile.glx,

make -f Makefile.glx clean,

Na skúšaných platformách sa má kód kompilovať bez chýb alebo upozornení.

Spustí sa 'verify.sh' na overenie vykonávania vo vzťahu k skúšobným vektorom.

Simulačný rámec

Simulačný rámec softvéru eCall sa nachádza v adresári 'test_setup'.

Dôležité poznámky:

* Pre podmienky použitia pozri LICENSE.TXT a README.TXT!

```
    * Softvér G.711 je časťou odporúčania ITU-T Rec. G.191, (C) ITU 2000.

      Distribuuje sa s autorizovaním ITU ako súčasť softvéru na nastavenie skúšok pre
      3GPP TS 26.268.

    * Pracovný rámec sa musí skompilovať a spustiť na systémoch MS Windows po pripojení
      hlasových kódovačov FR a AMR do systému vo forme vykonateľných programov Windows a
      prostredníctvom špecifických funkcií API.

Na vytvorenie (alebo čistenie) sa pracovný rámec spolu s eCall IVS a PSAP presunie do
podadresára 'c' z 'test_setup' a spustí (nezabudnúť 'vcvars32.bat'):

    nmake /f makefile_ecall,

    nmake /f makefile_ecall clean.

Pracovný rámec má päť funkcií spätného volania už zo skôr implementovaných funkcií.

Podľa prednastavenia sa binárne súbory (*.exe *.lib) vytvárajú v podadresári 'bin'.

Na spustenie vykonateľných programov sa môžu použiť tieto dva dávkové súbory:

    demosim.bat  : spustí testsim_demo.exe,

    demosock.bat : spustí testlab.exe a modem_demo.exe v zásuvnom režime.
```

4.3 Premenné, konštanty a tabuľky

4.3.1 Opis konštánt použitých v kóde C

Kapitola obsahuje zoznam všetkých globálnych konštánt definovaných v ecall_defines.h., spolu s niekoľkými vysvetľujúcimi komentármi.

Konštanta	Hodnota	Opis
`#define MAX(a,b)`	`((a)>(b) ? (a) : (b))`	
`#define MIN(a,b)`	`((a)<(b) ? (a) : (b))`	
`#define ABS(a)`	`((a)<0 ? (-a) : (a))`	
`#define SIGN(a)`	`((a)<0 ? (-1) : (1))`	
`#define PCM_LENGTH`	160	dĺžka rámca PCM
`#define MSD_MAX_LENGTH`	140	dĺžka správy MSD (bajtov)
`/* Synchronizácia */`		
`#define SYNC_BADCHECK`	(3)	chybná kontrola nepretržitosti synchronizácie
`#define SYNC_BADTRACK`	(4)	chybné sledovanie nepretržitosti synchronizácie
`#define SYNC_IDXLEN`	(75)	dĺžka synchronizačného indexu
`#define SYNC_THRESHOLD`	(10e6)	synchronizačný prah

```
#define LOCK_RESYNC             (2)             správy na zablokovanie po strate
                                                synchronizácie

#define LOCK_START_UL           (2)             správy START na zosynchronizovanie(UL)

#define LOCK_START_DL           (3)             správy START na zosynchronizovanie(DL)

#define FAIL_RESTART            (3)             počet správ START na opakovaný štart

#define NRF_WAKEUP              (3)             počet rámcov na aktiváciu

#define NRF_SYNC                (13)            dĺžka synchronizácie v rámcoch

#define NRF_OBSERVE             (10)            počet sledovaných rámcov synchronizácie
     frames

#define NRF_RESYNC              (60)            rámce s obnovenou synchronizáciou po jej
                                                strate

#define NRS_CP                  (2)             počet vzoriek vedľa špičiek

#define NRS_TRACK               (240)           počet sledovaných vzoriek

#define NRS_CHECK               (480)           počet kontrolovaných vzoriek

#define PNSEQ_OSF               (22)            frekvencia prevzorkovania postupnosti PN

#define PEAK_DIST_PP            (30*PNSEQ_OSF)  vzdialenosť vonkajších kladných špičiek

#define PEAK_DIST_NN            (54*PNSEQ_OSF)  vzdialenosť záporných špičiek

#define PEAK_DIST_PN            (12*PNSEQ_OSF)  vzdialenosť od kladnej k zápornej špičke

/* Vzostupný/zostupný formát */

#define ARQ_MAX                 (8)             počet redundantných verzií

#define NRB_TAIL                (3)             počet záverečných bitov kódovača

#define NRB_CRC                 (28)            rád polynómu CRC

#define NRB_INFO                (8*MSD_MAX_LENGTH)

#define NRB_INFO_CRC            (8*MSD_MAX_LENGTH + NRB_CRC)

#define NRB_CODE_ARQ            (1380)

#define NRB_CODE_BUFFER         (3*(8*MSD_MAX_LENGTH + NRB_CRC) + 4*NRB_TAIL)

#define SET_LLMSG               (16)            nastavenie veľkosti správ nižšej vrstvy

#define SET_HLMSG               (16)            nastavenie veľkosti správ vyššej vrstvy

#define NRF_DLDATA              (3)             zostupné dátové rámce

#define NRF_DLMUTE1LL           (3)             1. správa mlčania nižšej vrstvy

#define NRF_DLMUTE1HL           (1)             1. správa mlčania vyššej vrstvy

#define NRF_DLCHUNK             (NRF_SYNC + NRF_DLMUTE1HL + 2*NRF_DLDATA)

/* spracovanie IVS/PSAP */

#define NRF_MEMCTRL             (7)
```

```
#define NRS_MEMSYNC             (508 + 38*NRS_CP)

#define IVS_THRESHOLD           (40000)        prah pre riadiace správy

#define IVS_GOSTART             (6)            prah pre nespoľahlivý START

#define IVS_TXFAST              (10)           podmienka NACK pre rýchly režim
                                                modulátora

#define IVS_TXINC               (87)           zvýšenie vzorky pri opakovanom štarte

#define PSAP_NUMSTART           (500)          počet správ START

#define PSAP_NUMACK             (5)            počet správ ACK

#define PSAP_NUMHLACK           (5)            počet správ PSAP HLACK

#define PSAP_THRESHOLD          (40)           prah pre typ modulátora

#define FEC_VAR                 (30206)        odlišnosť: 1/4550000 v Q37

#define FEC_MEAN                (0xB9999A)     priemer: 5.8 v Q21

#define FEC_ITERATIONS          (8)            počet iterácií dekódovača

#define FEC_STATES              (8)            počet stavov dekódovača

#define IntLLR                  Int16          veľkosť premenných vyrovnávacieho
                                               zásobníka pravdepodobností hodnoty bitov

#define LLR_MAX                 ((Int32)(0x7fff-1))

#define LOGEXP_RES              (401)          rozlíšenie tabuľky LOGEXP

#define LOGEXP_DELTA            (-6)           určenie vnútorného činiteľa Q

#define LOGEXP_QIN              (8)            vstup činiteľa Q hodnôt LLR
```

4.3.2 Definície typov

Použili sa definície typov z ecall_defines.h, ecall_modem.h, ecall_sync.h a modemx.h:

Definícia	Opis
`typedef enum { False, True } Bool;`	logická premenná
`typedef enum { Minus = -1, Zero, Plus } Tern;`	trojitá premenná
`typedef signed char Int8;`	osembitová premenná so znamienkom
`typedef signed short int Int16;`	šestnásťbitová premenná so znamienkom
`typedef signed int Int32;`	tridsaťdvabitová premenná so znamienkom
`typedef unsigned char Ord1;`	binárny symbol
`typedef unsigned char Ord8;`	osembitová premenná bez znamienka

```
typedef unsigned short int Ord16;      šestnásťbitová premenná bez znamienka

typedef unsigned int       Ord32;      tridsaťdvabitová premenná bez znamienka

typedef enum {

  ModUndef,

  Mod3bit4smp,

  Mod3bit8smp

} ModType;                             typ modulátora na vzostupný prenos

typedef struct {

  ModType type;                        identifikácia typu modulátora

  Int16 bpsym;                         bitov na symbol

  Int16 spmf;                          vzoriek na modulačný rámec

  Int16 mfpf;                          modulačných rámcov na rámec = PCM_LENGTH/spmf

  Int16 decpos1;                       pozícia 1. dekódovacieho pokusu

  Int16 decpos2;                       pozícia 2. dekódovacieho pokusu

  Int16 wutperiod;                     perióda aktivačného tónu vo vzorkách

  Int16 nfmute1;                       počet rámcov mlčania 1. interval

  Int16 nfmute4;                       počet rámcov mlčania 4. interval

  Int16 nfmuteall;                     celkový počet rámcov mlčania

  Int16 nfdata;                        počet dátových rámcov = NRB_CODE_ARQ/(mfpf*bpsym)

  const Int16 *ulPulse;

  const Int16 *ulPulseMatch;

  const Int16 *mgTable;

  const Int16 *wakeupSin;

  const Int16 *wakeupCos;

} ModState;                            stav modulátora na vzostupný prenos

typedef struct {

  Int32 *mem;              /* pamäť pre synchronizáciu */

  Int32 *memWakeup;        /* pamäť pre detektor aktivačného tónu */


  SyncSub syncPos;         /* normálna synchronizácia (neinvertovaná) */

  SyncSub syncNeg;         /* invertovaná synchronizácia */

  Int32 amplitude[3];      /* amplitúdy (priemerná, maximálna, pamäť) */
```

```
  Int32 shape[2*NRS_CP+1];  /* tvar špičky spôsobujúcej synchronizačnú udalosť */

  Bool flag;                /* indikácia úspešnej synchronizácie */

  Bool invert;              /* indikácia inverznej synchronizácie */

  Bool resync;              /* indikácia výskytu opakovanej synchronizácie */

  Int16 delay;              /* synchronizačné oneskorenie */

  Int16 delayMem;           /* synchronizačné oneskorenie (pamäť) */

  Int16 npeaks;             /* počet detegovaných synchronizačných špičiek */

  Int16 npeaksMem;          /* počet detegovaných synchronizačných špičiek (pamäť) */

  Int16 events;             /* počet rovnakých synchronizačných udalostí nasledujúcich po
                               sebe*/

  Tern  check;              /* indikácia výsledku kontroly synchronizácie (trojitá
                               premenná) */

  Int16 checkCnt;           /* počítadlo následných chybných kontrol synchronizácie */

  Int16 index;              /* rámec referencie na vyhodnotenie synchronizácie */

} SyncState;

typedef struct {

  Int32 amplitude[2];       /* amplitúdy (priemerná, maximálna) */

  Int32 shape[2*NRS_CP+1];  /* tvar špičky spôsobujúcej synchronizačnú udalosť */

  Bool  flag;               /* indikácia úspešnej synchronizácie */

  Int16 delay;              /* synchronizačné oneskorenie */

  Int16 npeaks;             /* počet detegovaných synchronizačných špičiek */

  Int16 npeaksChk;          /* počet synchronizačných špičiek detegovaných kontrolou
                               synchronizácie */

} SyncSub;

typedef enum {

  DlMsgNoop = -2,

  DlMsgReset,

  DlMsgStart,

  DlMsgNack,

  DlMsgAck,

  DlMsgHlack = SET_LLMSG

} DlData;                             identifikátory zostupných správ

typedef enum {

  DlCntStart = -2,
```

```
  DlCntWait,
  DlCntNext
} DlCount;                          počítadlo zostupných správ
typedef enum {
  IVSEVENT_IDLE,
  IVSEVENT_SENDINGSTART,
  IVSEVENT_SENDINGMSD,
  IVSEVENT_RESTARTMSD,
  IVSEVENT_CONTROLSYNC,
  IVSEVENT_CONTROLLOCK,
  IVSEVENT_LLACKRECEIVED,
  IVSEVENT_HLACKRECEIVED,
  IVSEVENT_TIMEOUT
} IvsEvent;
typedef enum {
  IvsIdle,
  IvsTrigger,
  IvsStart,
  IvsSendMsd,
  IvsAck,
  IvsHlack
} IvsState;                         stavové identifikátory IVS
typedef struct {
  IvsState   state;                 stav prijímača IVS
  CtrlRxData ctrl;                  riadiaca štruktúra IVS
  SyncState  sync;                  synchronizačná štruktúra IVS
  Int16 dlData;                     symbol zostupnej správy
  Int16 dlIndex;                    počítadlo zostupných rámcov
  Int16 dlMsgCnt;                   počítadlo zostupných správ
  Int16 memCtrl[NRF_MEMCTRL*PCM_LENGTH];
  Int32 memSync[NRS_MEMSYNC];
} IvsRxData;
```

```
typedef struct {
  IvsState   state;                    stav vysielača IVS
  CtrlTxData ctrl;                     riadiaca štruktúra IVS
  ModState mod;                        štruktúra modulátora IVS
  Int16 delay;                         prenosový posun vo vzorkách
  Int16 rv;                            redundantná verzia
  Int16 ulN;                           počet vzostupných rámcov
  Int16 ulIndex;                       počítadlo vzostupných rámcov
  Int16 ulDelay;                       vzostupný prenosový posun vo vzorkách
  Int16 dlMsgOld;                      predchádzajúca riadiaca správa
  Bool  pendingStart;                  /* indikácia čakajúcej správy START */
  Int16 overallNack;                   /* kumulatívne počítanie NACK */
  Int16 stateCnt[SET_LLMSG + 1];       stavové počítadlá
  Int16 stateIgn[SET_LLMSG + 1];       počítadlo nespoľahlivých správ
  Ord1  memCode[NRB_CODE_BUFFER];
  Int16 memDelay[2*PCM_LENGTH];
} IvsTxData;
typedef struct {
  IvsRxData rx;                        štruktúra prijímača IVS
  IvsTxData tx;                        štruktúra vysielača IVS
} IvsData;
typedef enum {
  PSAPEVENT_IDLE,
  PSAPEVENT_SENDINGSTART,
  PSAPEVENT_SENDINGRESTART,
  PSAPEVENT_SENDINGLLACK,
  PSAPEVENT_SENDINGHLACK,
  PSAPEVENT_CONTROLSYNC,
  PSAPEVENT_CONTROLLOCK,
  PSAPEVENT_MSDSYNC,
  PSAPEVENT_MSDRECEIVED,
  PSAPEVENT_TIMEOUT
```

```
} PsapEvent;

typedef enum {

  PsapIdle,

  PsapTrigger,

  PsapStart,

  PsapNack,

  PsapAck,

  PsapHlack,

} PsapState;                          stavové identifikátory PSAP

typedef struct {

  PsapState  state;                   stav prijímača PSAP

  CtrlRxData ctrl;                    riadiaca štruktúra PSAP

  SyncState sync;                     synchronizačná štruktúra PSAP

  ModState  mod;                      štruktúra modulátora PSAP

  Int16 rv;                           redundantná verzia

  Int16 ulN;                          počet vzostupných rámcov bez mlčania

  Int16 ulIndex;                      počítadlo vzostupných rámcov

  Int16 mgIndex;                      vzostupná pozícia v tabuľke medzier s mlčaním

  Int16 ulTrials;                     vzostupné dekódovanie trás

  Int16 ulSyncTail;                   počítadlo sledovania synchronizácie po úspešnej
                                      synchronizácii

  Ord8  dlHlackData;                  zostupná správa vyššej vrstvy (4 bity)

  Int16 dlData;                       symbol zostupnej správy

  Int16 dlIndex;                      počítadlo zostupných rámcov

  Int16 dlMsgCnt;                     počítadlo zostupných správ

  Ord8   *msd;                        MSD v bajtovom vyjadrení

  Ord1   *msdBin;                     MSD v binárnom vyjadrení

  Int16  *memCtrl;                    vyrovnávací zásobník na riadenie a demoduláciu dát

  IntLLR *memCode;                    vyrovnávací zásobník pravdepodobností hodnoty
                                      bitov na dekódovanie

  char buffer[0

    + sizeof(IntLLR)* NRB_CODE_ARQ

    + sizeof(Int16) * NRF_MEMCTRL*PCM_LENGTH
```

```
    + sizeof(Int32) * NRS_MEMSYNC
    + sizeof(Int32) * 2*(NRF_SYNC+1)];
} PsapRxData;
typedef struct {
  CtrlTxData ctrl;                    riadiaca štruktúra PSAP
} PsapTxData;
typedef struct {
  PsapRxData rx;                      štruktúra prijímača PSAP
  PsapTxData tx;                      štruktúra vysielača PSAP
  Int16 msgCounter;                                          počítadlo správ
} PsapData;
typedef enum {
  CtrlRxIdle,
  CtrlRxSync,
  CtrlRxLock,
  CtrlTxIdle,
  CtrlTxSend
} PortState;
typedef struct {
  PortState state;                    stav portu
  Bool invert;                        príznak inverzie portu
  union {
    CtrlTxPort tx;                    port riadenia vysielača
    CtrlRxPort rx;                    port riadenia prijímača
  } u;
  const char *owner;                  identifikácia vlastníka portu
} CtrlPort;
typedef struct {
  Int16 dlData;                       symbol správy
  Int16 dlIndex;                      počítadlo rámcov správy
} CtrlTxPort;
typedef struct {
```

```
  Int16 dlData;                       detegovaný symbol správy

  Int16 dlMetric;                     merač prijímača

} CtrlRxPort;

typedef struct {

  CtrlPort port;                      štruktúra portu

} CtrlTxData;

typedef struct {

  CtrlPort port;                      štruktúra portu

  SyncState *sync;                    ukazovateľ synchronizačnej štruktúry

  Int16 *buffer;                      ukazovateľ vyrovnávacieho zásobníka riadenia
                                      prijímača

  Ord8  dlHlackData;                  zostupná správa vyššej vrstvy (4 bity)

  Tern  dlRead;                       indikácia synchronizácie (trojitá premenná)

  Int16 dlIndex;                      počítadlo interných rámcov

  Int16 dlSyncLock;                   počet požadovaných synchronizačných udalostí

} CtrlRxData;
```

4.3.3 Opis pevných tabuliek použitých v kóde C

Kapitola obsahuje zoznam všetkých pevných tabuliek (ROM) definovaných v `ecall_rom.c`.

Typ/Konštanta	Veľkosť	Opis
`/* Synchronizácia */`		
`const Int16 wakeupSin500`	`[16]`	sínusový tvar vlny pri 500 Hz
`const Int16 wakeupCos500`	`[16]`	kosínusový tvar vlny pri 500 Hz
`const Int16 wakeupSin800`	`[10]`	sínusový tvar vlny pri 800 Hz
`const Int16 wakeupCos800`	`[10]`	kosínusový tvar vlny pri 800 Hz
`const Int16 syncPulseForm`	`[5]`	synchronizačný impulz
`const Int16 syncSequence`	`[15]`	postupnosť synchronizačných impulzov
`const Int16 syncIndexPreamble`	`[SYNC_IDXLEN]`	pozície synchronizačných impulzov
`const Int16 syncFrame`	`[10*PCM_LENGTH]`	preddefinovaný synchronizačný signál
`/* vzostupný/zostupný formát */`		
`const Int16 indexBits`	`[24]`	pozície bitov pre turbo dekódovač
`// rýchly režim modulátora:`		

```
const Int16 m4smp_ulPulse     [16]                      vzostupný tvar vlny

const Int16 m4smp_ulPulseMatch[64]                      priradené filtrované vzostupné
                                                         tvary vlny

const Int16 m4smp_mgTable     [66]                      tabuľka indikujúca medzery s
                                                         mlčaním

// robustný režim modulátora:

const Int16 m8smp_ulPulse     [32]                      vzostupný tvar vlny

const Int16 m8smp_ulPulseMatch[128]                     priradené filtrované vzostupné
                                                         tvary vlny

const Int16 m8smp_mgTable     [116]                     tabuľka indikujúca medzery s
                                                         mlčaním

const Int16 dlPcmData         [4][NRF_DLDATA*PCM_LENGTH]  zostupný vysielaný signál

const Int16 dlPcmDataMatch    [4][NRF_DLDATA*PCM_LENGTH]  DL MF signál

/* kódovač/dekódovač FEC */

const Ord16 stateTransMat     [8][2]                    FEC: stav prechodu

const Ord16 stateTrans        [16]                      FEC: stav prechodu

const Ord16 revStateTransMat  [8][2]                    FEC: reverzné stavy prechodov

const Ord16 revStateTrans     [16]                      FEC: reverzné stavy prechodov

const Ord1  outputParityMat   [8][2]                    FEC: indikátor výstupnej parity

const Ord1  outputParity      [16]                      FEC: indikátor výstupnej parity

const Ord1  crcPolynomial     [NRB_CRC+1]               koeficienty polynómu CRC

const Ord1  scramblingSeq     [NRB_INFO_CRC]            postupnosť bitového skramblovania

const Ord16 interleaverSeq    [NRB_INFO_CRC]            postupnosť prekladača

const Ord16 redVerIndex       [8][NRB_CODE_ARQ]         indexový vektor pre proces HARQ

const IntLLR logExpTable      [LOGEXP_RES]              vyhľadávacia tabuľka(funkcia
                                                         logExp)
```

4.3.4 Statické premenné použité v kóde C

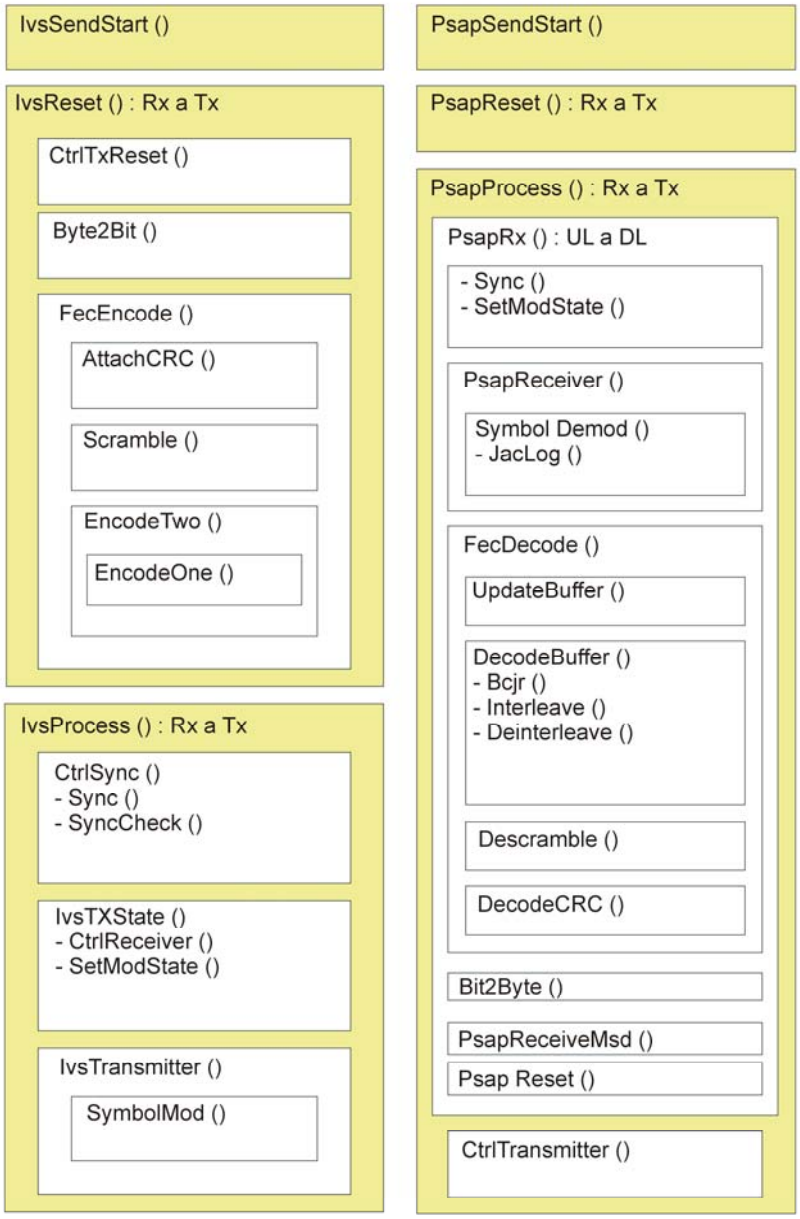
Kapitola obsahuje zoznam statických premenných (RAM) definovaných v zdrojových súboroch.

Definícia	Opis
`IvsData ivs`	statická pamäť IVS
`PsapData psap`	statická pamäť PSAP
`IntLLR chLLRbuffer[NRB_CODE_BUFFER]`	vyrovnávací zásobník pravdepodobnosti hodnoty bitov turbo dekódovača

4.4 Funkcie kódu C

Kapitola obsahuje záhlavia použitých funkcií IVS a PSAP. Zodpovedajú rozsiahlemu funkčnému opisu IVS a PSAP, ktorý sa uvádza v technickej špecifikácii 3GPP TS 26.267 [1].

Obrázok 1 podáva prehľad o najdôležitejších funkciách a ich hierarchickom vzťahu.



Obrázok 1 – Hierarchický prehľad funkcií

4.4.1 Funkcie rozhrania

```
/*============================================================================*/
/* Implementácia IVS: IvsReset                                                */
/*----------------------------------------------------------------------------*/
/* Opis: reset IVS pred príjmom nového MSD                                    */
/*                                                                            */
/* Vstup:  const Ord8* msd     -> vysielané MSD                               */
/*         int         length  -> dĺžka MSD (rovnajúca sa MSD_MAX_LENGTH)     */
/*----------------------------------------------------------------------------*/
void IvsReset(const Ord8 *msd, int length)
void IvsRxReset(void)
void IvsTxReset(const Ord8 *msd, int length)
/*============================================================================*/
/* Implementácia IVS: IvsProcess                                              */
/*----------------------------------------------------------------------------*/
/* Opis: funkcia modemu IVS, ktorá spracováva dáta PCM                        */
/*                                                                            */
/* Vstup/výstup:  Int16* pcm  <-> vstupný a výstupný rámec šestnásťbitových   */
/*---------------------------------- vzoriek PCM -----------------------------*/
void IvsProcess(Int16 *pcm)
void IvsRxProcess(const Int16 *pcm)
void IvsTxProcess(Int16 *pcm)
/*============================================================================*/
/* Implementácia IVS: IvsSendStart                                            */
/*----------------------------------------------------------------------------*/
/* Opis: inicializuje IVS, aby spustil vysielanie správ SEND                  */
/*----------------------------------------------------------------------------*/
void IvsSendStart(void);
/*============================================================================*/
/* Implementácia IVS: IvsReceiveAck                                           */
/*----------------------------------------------------------------------------*/
```

```
/* Opis: funkcia spätného volania, indikujúca prijatú správu ACK              */
/*----------------------------------------------------------------------------*/
void IvsReceiveAck(void);
/*============================================================================*/
/* Implementácia IVS: IvsReceiveHlack                                         */
/*----------------------------------------------------------------------------*/
/* Opis: funkcia spätného volania, indikujúca prijaté správy vyššej vrstvy    */
/*                                                                            */
/* Vstup:   const Ord8 data -> identifikátor dátových symbolov                */
/*----------------------------------------------------------------------------*/
void IvsReceiveHlack(const Ord8 data);
/*============================================================================*/
/* Implementácia PSAP: PsapSendStart                                          */
/*----------------------------------------------------------------------------*/
/* Opis: inicializuje PSAP, aby spustilo prenos MSD                           */
/*----------------------------------------------------------------------------*/
void PsapSendStart(void)
/*============================================================================*/
/* Implementácia PSAP: PsapSendHlack                                          */
/*----------------------------------------------------------------------------*/
/* Opis: inicializuje PSAP, aby poslalo správy vyššej vrstvy                  */
/*                                                                            */
/* Vstup:   const Ord8 data -> identifikátor dátových symbolov                */
/*----------------------------------------------------------------------------*/
void PsapSendHlack(const Ord8 data);
/*============================================================================*/
/* Implementácia PSAP: PsapReset                                              */
/*----------------------------------------------------------------------------*/
/* Opis: reset PSAP pred príjmom nového MSD                                   */
/*----------------------------------------------------------------------------*/
void PsapReset(void)
void PsapRxReset(void)
```

```
void PsapTxReset(void)

/*============================================================================*/
/* Implementácia PSAP: PsapProcess                                            */
/*----------------------------------------------------------------------------*/
/* Opis: funkcia modemu PSAP, ktorá spracováva dáta PCM                       */
/*                                                                            */
/* Vstup/Výstup: Int16* pcm <-> vstupný a výstupný rámec šestnásťbitových     */
/*                              vzoriek PCM                                   */
/*----------------------------------------------------------------------------*/
void PsapProcess(Int16 *pcm)

void PsapRxProcess(const Int16 *pcm)

void PsapTxProcess(Int16 *pcm)
```

4.4.2 Funkcie vysielača IVS

```
/*============================================================================*/
/* FUNKCIA IVS: IvsTransmitter                                                */
/*----------------------------------------------------------------------------*/
/* Opis: funkcia vysielača IVS                                                */
/*                                                                            */
/* Vstup:  const ModState* ms      -> stav modulátora                         */
/*         const Ord1*     buffer  -> vyrovnávací zásobník kódových bitov     */
/*         Int16           rv      -> redundantná verzia                      */
/*         Int16           index   -> pozícia vo vzostupnom rámci             */
/* Výstup: Int16*          pcm     <- výstupné dáta                           */
/*----------------------------------------------------------------------------*/
void IvsTransmitter(const ModState *ms, const Ord1 *buffer, Int16 *pcm,
                    Int16 rv, Int16 index)
/*============================================================================*/
/* POMOCNÁ FUNKCIA: IvsTxState                                                */
/*----------------------------------------------------------------------------*/
/* Opis: stav zariadenia IVS, ktoré vyhodnocuje správy spätnej väzby          */
/*                                                                            */
/* Vstup: Int16 msg       -> nový symbol riadiacej správy                     */
```

```
/*        Int16 metric    -> merač prijímača (-1: symbol sa ignoruje)         */
/*        Bool  syncLock  -> indikuje zosynchronizovanie prijímača riadenia   */
/*----------------------------------------------------------------------------*/
void IvsTxState(Int16 msg, Int16 metric, Bool syncLock)
/*============================================================================*/
/* FUNKCIA IVS: SymbolMod                                                     */
/*----------------------------------------------------------------------------*/
/* Opis: modulátor symbolov                                                   */
/*                                                                            */
/* Vstup:  const ModState* ms      -> stav modulátora                         */
/*         Int16           symbol  -> index symbolu                           */
/* Výstup: Int16*          mPulse  <- modulovaná výstupná postupnosť          */
/*----------------------------------------------------------------------------*/
void SymbolMod(const ModState *ms, Int16 symbol, Int16 *mPulse)
/*============================================================================*/
/* FUNKCIA IVS: Byte2Bit                                                      */
/*----------------------------------------------------------------------------*/
/* Opis: konverzia bajtového vektora na bitový vektor                        */
/*                                                                            */
/* Vstup:  Ord8* in      -> vector of input bytes                             */
/*         Int16 length  -> length of input                                   */
/* Výstup: Ord1* out     <- vector of output bits                             */
/*----------------------------------------------------------------------------*/
void Byte2Bit(const Ord8 *in, Ord1 *out, Int16 length)
/*============================================================================*/
/* Funkcia kódovača: FecEncode                                                */
/*----------------------------------------------------------------------------*/
/* Opis: kódovanie MSD                                                        */
/*                                                                            */
/* Vstup/Výstup: Ord1 *buffer  <-> vstup informačných bitov, výstup kódovaných*/
/*                                 bitov                                      */
/*----------------------------------------------------------------------------*/
```

```
void FecEncode(Ord1 *buffer)
/*============================================================================*/
/* FUNKCIA KÓDOVAČA: AttachCrc                                                */
/*----------------------------------------------------------------------------*/
/* Opis: pripojenie bitov CRC                                                 */
/*                                                                            */
/* Vstup:     const Ord1* infoBits     -> vstup informačných bitov            */
/* Výstup:    Ord1*       infoWithCrc  <- bity s pripojeným CRC               */
/*----------------------------------------------------------------------------*/
void AttachCrc(const Ord1 *infoBits, Ord1 *infoWithCrc)
/*============================================================================*/
/* FUNKCIA KÓDOVAČA: Scramble                                                 */
/*----------------------------------------------------------------------------*/
/* Opis: bitové skramblovanie                                                 */
/*                                                                            */
/* Vstup:     const Ord1* in   -> neskramblovaná vstupná bitová postupnosť    */
/* Výstup:    Ord1*       out  <- skramblovaná výstupná bitová postupnosť     */
/*----------------------------------------------------------------------------*/
void Scramble(const Ord1 *in, Ord1 *out)
/*============================================================================*/
/* FUNKCIA KÓDOVAČA: EncodeTwo                                                */
/*----------------------------------------------------------------------------*/
/* Opis: kódovanie bitovej postupnosti                                        */
/*                                                                            */
/* Vstup/Výstup:  Ord1* codedBits  <-> skramblované bity do kódovaných bitov  */
/*----------------------------------------------------------------------------*/
void EncodeTwo(Ord1 *codedBits)
/*============================================================================*/
/* FUNKCIA KÓDOVAČA: EncodeOne                                                */
/*----------------------------------------------------------------------------*/
/* Opis: konvolučné kódovanie každého prvku                                   */
/*                                                                            */
```

```
/* Vstup:         Int16 encNr       -> číslo prvku                          */
/* Vstup/Výstup:  Ord1* codedBits  <-> kódované bity                        */
/*--------------------------------------------------------------------------*/
void EncodeOne(Ord1 *codedBits, Int16 encNr)
```

4.4.3 Funkcie prijímača PSAP

```
/*==========================================================================*/
/* POMOCNÁ FUNKCIA: PsapRxUplink                                            */
/*--------------------------------------------------------------------------*/
/* Opis: stav zariadenia PSAP UL, podľa stavu sa určuje činnosť prijímača PSAP*/
/*                                                                          */
/* Vstup:  const Int16* pcm  -> vstupný rámec šestnásťbitových vzoriek PCM  */
/*--------------------------------------------------------------------------*/
void PsapRxUplink(const Int16 *pcm)
/*==========================================================================*/
/* POMOCNÁ FUNKCIA: PsapRxDownlink                                          */
/*--------------------------------------------------------------------------*/
/* Opis: stav zariadenia PSAP UL, podľa stavu sa určuje činnosť vysielača PSAP*/
/*--------------------------------------------------------------------------*/
void PsapRxDownlink(void)
/*==========================================================================*/
/* FUNKCIA PSAP: PsapReceiver                                               */
/*--------------------------------------------------------------------------*/
/* Opis: funkcia prijímača PSAP (dekódovanie sa vykoná mimo)                */
/*                                                                          */
/* Vstup:  const ModState* ms        -> stav modulátora                     */
/*         const Int16*    pcm       -> vstupné dáta na demoduláciu         */
/* Výstup: IntLLR*         softBits  <- demodulovaná postupnosť             */
/*                                       pravdepodobnosti hodnoty bitov     */
/*--------------------------------------------------------------------------*/
void PsapReceiver(const ModState *ms, const Int16 *pcm, IntLLR *softBits)
/*==========================================================================*/
/* FUNKCIA PSAP: SymbolDemod                                                */
```

```
/*---------------------------------------------------------------------------*/
/* Opis: demodulátor symbolov                                                */
/*                                                                           */
/* Vstup:  const ModState* ms        -> stav modulátora                      */
/*         const Int16*    mPulse    -> prijatý sled impulzov                */
/* Výstup: IntLLR*         softBits  <- demodulovaná postupnosť              */
/*                                       pravdepodobnosti hodnoty bitov      */
/*---------------------------------------------------------------------------*/
void SymbolDemod(const ModState *ms, const Int16 *mPulse, IntLLR *softBits)
/*===========================================================================*/
/* FUNKCIA PSAP: Bit2Byte                                                    */
/*---------------------------------------------------------------------------*/
/* Opis: konverzia bitového vektora na bajtový vektor                        */
/*                                                                           */
/* Vstup:  const Ord1* in      -> vektor vstupných bitov                     */
/*         Int16       length  -> dĺžka výstupu                              */
/* Výstup: Ord8*       out     <- vektor výstupných bajtov                   */
/*---------------------------------------------------------------------------*/
void Bit2Byte(const Ord1 *in, Ord8 *out, Int16 length)
/*===========================================================================*/
/* FUNKCIA PSAP: MpyLacc                                                     */
/*---------------------------------------------------------------------------*/
/* Opis: násobenie tridsaťdvabitového čísla so šestnásťbitovým číslom        */
/*       (tridsaťdvabitovýbitový výsledok)                                   */
/*                                                                           */
/* Vstup:  Int32 var32  -> tridsaťdvabitové číslo                            */
/*         Int16 var16  -> šestnásťbitové číslo                              */
/* Návrat: Int32        <- výsledok                                          */
/*---------------------------------------------------------------------------*/
Int32 MpyLacc(Int32 var32, Int16 var16)
/*===========================================================================*/
/* FUNKCIA DEKÓDOVAČA: FecDecode                                             */
```

```
/*---------------------------------------------------------------------------*/
/* Opis: dekódovanie na získanie MSD                                         */
/*                                                                           */
/* Vstup:  const IntLLR* in   -> prijaté pravdepodobnosti hodnoty bitov      */
/*         Int16         rv   -> redundantná verzia                          */
/* Výstup: Ord1*         out  <- dekódovaný MSD v binárnom vyjadrení          */
/* Návrat: Bool               <- výsledok kontroly CRC                       */
/*---------------------------------------------------------------------------*/
Bool FecDecode(const IntLLR *in, Int16 rv, Ord1 *out)
/*===========================================================================*/
/* DECODER FUNCTION: UpdateBuffer                                            */
/*---------------------------------------------------------------------------*/
/* Opis: vyrovnávací zásobník LLR vzostupného kanála s novými                */
/*       pravdepodobnosťami hodnoty bitov                                    */
/*                                                                           */
/* Vstup:         const IntLLR* softInBits    -> prijaté pravdepodobnosti    */
/*                                               hodnoty bitov               */
/*                Int16         rv            -> redundantná verzia          */
/* Vstup/Výstup:  IntLLR*       chLLRbuffer  <-> vyrovnávací zásobník        */
/*                                               dekódovača                  */
/*---------------------------------------------------------------------------*/
void UpdateBuffer(IntLLR *chLLRbuffer, const IntLLR *softInBits, Int16 rv)
/*===========================================================================*/
/* FUNKCIA DEKÓDOVAČA: DecodeBuffer                                          */
/*---------------------------------------------------------------------------*/
/* Opis: dekódovanie vyrovnávacieho zásobníka LLR                            */
/*                                                                           */
/* Vstup:  const IntLLR* syst1    -> RX systematické pravdepodobnosti hodnoty */
/*                                   bitov                                   */
/*         const IntLLR* syst2    -> prekladané RX systematické záverečné bity*/
/*         const IntLLR* parity1  -> RX paritné pravdepodobnosti hodnoty bitov*/
/*         const IntLLR* parity2  -> prekladané RX paritné pravdepodobnosti   */
```

```
/*                                  hodnoty bitov                             */
/* Výstup: Ord1*         decBits  <- dekódované bity                         */
/*-------------------------------------------------------------------------*/
void DecodeBuffer(const IntLLR *syst1, const IntLLR *syst2,
                  const IntLLR *parity1, const IntLLR *parity2, Ord1 *decBits)
/*=========================================================================*/
/* FUNKCIA DEKÓDOVAČA: Bcjr                                                */
/*-------------------------------------------------------------------------*/
/* Opis: algoritmus BCJR                                                   */
/*                                                                         */
/* Vstup:         const IntLLR* parity      -> prijaté paritné             */
/*                                             pravdepodobnosti hodnoty bitov */
/* Vstup/Výstup:  IntLLR*       extrinsic  <-> externá informácia          */
/*-------------------------------------------------------------------------*/
void Bcjr(const IntLLR *parity, IntLLR *extrinsic)
/*=========================================================================*/
/* FUNKCIA DEKÓDOVAČA: Interleave                                          */
/*-------------------------------------------------------------------------*/
/* Opis: prekladač turbo kódovania                                         */
/*                                                                         */
/* Vstup:     const IntLLR* in   -> vstupná postupnosť                     */
/* Výstup:    IntLLR*       out  <- výstupná postupnosť                    */
/*-------------------------------------------------------------------------*/
void Interleave(const IntLLR *in, IntLLR *out)
/*=========================================================================*/
/* FUNKCIA DEKÓDOVAČA: Deinterleave                                        */
/*-------------------------------------------------------------------------*/
/* Opis: spätný prekladač turbo kódovania                                  */
/*                                                                         */
/* Vstup/Výstup:  IntLLR* inout  <-> vstupná a spätne preložená výstupná   */
/*                                  postupnosť                             */
/*-------------------------------------------------------------------------*/
```

```
void Deinterleave(IntLLR *inout)
/*============================================================================*/
/* FUNKCIA DEKÓDOVAČA: Descramble                                             */
/*----------------------------------------------------------------------------*/
/* Opis: deskramblujú sa dekódované bity                                      */
/*                                                                            */
/* Vstup/Výstup:  Ord1* inout  <-> vstupná a výstupná bitová postupnosť       */
/*----------------------------------------------------------------------------*/
void Descramble(Ord1 *inout)
/*============================================================================*/
/* FUNKCIA DEKÓDOVAČA: DecodeCrc                                              */
/*----------------------------------------------------------------------------*/
/* Opis: kontrola CRC dekódovaných bitov                                      */
/*                                                                            */
/* Vstup:  const Ord1* codedBits  -> kontrolovaná postupnosť dekódovaných     */
/*                                   bitov                                    */
/* Návrat: Bool                   <- výsledok kontroly CRC                    */
/*----------------------------------------------------------------------------*/
Bool DecodeCrc(const Ord1 *codedBits)
/*============================================================================*/
/* FUNKCIA DEKÓDOVAČA: GammaQ                                                 */
/*----------------------------------------------------------------------------*/
/* Opis: výpočet gama hodnôt pre algoritmus BCJR                              */
/*                                                                            */
/* Vstup:  Int16         k          -> pozícia bitu                           */
/*         Int16         l          -> stav                                   */
/*         const IntLLR* parity     -> prijaté paritné bity                   */
/*         const IntLLR* extrinsic  -> suma externých  a systematických bitov */
/* Návrat: IntLLR                   <- hodnota gama(k,l)                      */
/*----------------------------------------------------------------------------*/
IntLLR GammaQ(Int16 k, Int16 l, const IntLLR *parity, const IntLLR *extrinsic)
/*============================================================================*/
```

```
/* POMOCNÁ FUNKCIA: JacLog                                                    */
/*----------------------------------------------------------------------------*/
/* Opis: Jacobiho logaritmus                                                  */
/*                                                                            */
/* Vstup:  IntLLR a  -> prvá hodnota                                          */
/*         IntLLR b  -> druhá hodnota                                         */
/* Návrat: IntLLR    <- Jacobiho logaritmus                                   */
/*----------------------------------------------------------------------------*/
IntLLR JacLog(Int32 a, Int32 b)
```

4.4.4 Funkcie vysielača PSAP

Pozri funkcie riadiaceho spoja.

4.4.5 Funkcie prijímača IVS

Pozri funkcie riadiaceho spoja.

4.4.6 Funkcie synchronizácie (IVS a PSAP)

```
/*============================================================================*/
/* FUNKCIA: Sync                                                              */
/*----------------------------------------------------------------------------*/
/* Opis: hlavná synchronizačná funkcia                                        */
/*                                                                            */
/* Vstup/Výstup:  SyncState*   sync    <-> stav synchronizácie                */
/* Vstup:         const Int16* pcm      -> vstupný rámec                      */
/*                const char*  caller   -> identifikácia modemu               */
/*                Bool         invert   -> príznak inverzie portu             */
/*----------------------------------------------------------------------------*/
void Sync(SyncState *sync, const Int16 *pcm, const char *caller, Bool invert)
/*============================================================================*/
/* POMOCNÁ FUNKCIA: CtrlSync                                                  */
/*----------------------------------------------------------------------------*/
/* Opis: synchronizačná funkcia riadiacej správy                              */
/*                                                                            */
/* Vstup/Výstup:  CtrlRxData*  control  <-> riadiaca štruktúra                */
/* Vstup:         const Int16* pcm   > vstupný rámec šestnásťbitových vzoriek */
```

```
/*                                          PCM samples                        */
/*-------------------------------------------------------------------------------*/
void CtrlSync(CtrlRxData *control, const Int16 *pcm)
/*===============================================================================*/
/* POMOCNÉ FUNKCIE: SyncSubPut, SyncSubGet, SyncSubCpy                           */
/*-------------------------------------------------------------------------------*/
/* Vstup/Výstup:  SyncState* sync  <-> stav synchronizácie                       */
/* Vstup/Výstup:  SyncSub*   ssub  <-> synchronizačný subsystém                  */
/*-------------------------------------------------------------------------------*/
void SyncSubPut(SyncState *sync, SyncSub *ssub)
void SyncSubGet(SyncState *sync, SyncSub *ssub)
void SyncSubCpy(const SyncSub *ssubIn, SyncSub *ssubOut)
/*===============================================================================*/
/* POMOCNÁ FUNKCIA: SyncSubRun                                                   */
/*-------------------------------------------------------------------------------*/
/* Opis: vyhodnotenie synchronizačnej špičky                                     */
/*                                                                               */
/* Vstup/Výstup:  SyncSub*     ssub    <-> synchronizačný subsystém              */
/* Vstup:         const char*  caller   -> identifikácie modemu                  */
/*                const Int32* pPos     -> pozície kladných špičiek              */
/*                const Int32* pCorr    -> korelačné hodnoty kladných špičiek    */
/*                const Int32* nPos     -> pozície záporných špičiek             */
/*                const Int32* nCorr    -> korelačné hodnoty záporných špičiek*/
/*-------------------------------------------------------------------------------*/
void SyncSubRun(SyncSub *ssub, const char *caller,
                const Int32 *pPos, const Int32 *pCorr,
                const Int32 *nPos, const Int32 *nCorr)
/*===============================================================================*/
/* FUNKCIA IVS: SyncCheck                                                        */
/*-------------------------------------------------------------------------------*/
/* Opis: kontrola platnosti zosynchronizovania                                   */
/*                                                                               */
```

```
/* Vstup/Výstup:  SyncState*   sync   <-> stav synchronizácie                */
/* Vstup:         const Int16* pcm     -> vstupný rámec                     */
/*                const char*  caller  -> identifikácia modemu              */
/*--------------------------------------------------------------------------*/
void SyncCheck(SyncState *sync, const Int16 *pcm, const char *caller)
/*==========================================================================*/
/* FUNKCIA IVS: SyncTrack                                                   */
/*--------------------------------------------------------------------------*/
/* Opis: monitor vzostupnej synchronizácie                                  */
/*                                                                          */
/* Vstup/Výstup:  SyncState* sync    <-> stav synchronizácie                */
/* Vstup:         Bool       invert   -> príznak inverzie portu             */
/*--------------------------------------------------------------------------*/
void SyncTrack(SyncState *sync, Bool invert)
/*==========================================================================*/
/* FUNKCIA: SyncFilter                                                      */
/*--------------------------------------------------------------------------*/
/* Opis: implementácia synchronizačného filtra                              */
/*                                                                          */
/* Vstup/Výstup:  SyncState*   sync    <-> stav synchronizácie              */
/* Vstup:         const Int16* pcm      -> vstupný rámec                    */
/*                Bool         invert   -> príznak inverzie portu           */
/*--------------------------------------------------------------------------*/
void SyncFilter(SyncState *sync, const Int16 *pcm, Bool invert)
/*==========================================================================*/
/* POMOCNÁ FUNKCIA: ToneDetect                                              */
/*--------------------------------------------------------------------------*/
/* Opis: detekcia tónu pri 500 Hz alebo 800 Hz                              */
/*                                                                          */
/* Vstup/Výstup:  SyncState*   sync  <-> stav synchronizácie                */
/* Vstup:         const Int16* pcm    -> vstupný rámec                      */
/*--------------------------------------------------------------------------*/
```

```
void ToneDetect(SyncState *sync, const Int16 *pcm)

/*============================================================================*/
/* POMOCNÁ FUNKCIA: PeakUpdate                                                */
/*----------------------------------------------------------------------------*/
/* Opis: aktualizácia pozície synchronizačnej špičky                          */
/*                                                                            */
/* Vstup:  const Int32* pos   -> vektor pozícií                               */
/*         const Int32* corr  -> vektor korelačných hodnôt                    */
/*         Int16        dist  -> kontrolovaná vzdialenosť                     */
/* Návrat: Int16              <- aktualizovaná pozícia špičky                 */
/*----------------------------------------------------------------------------*/
Int16 PeakUpdate(const Int32 *pos, const Int32 *corr, Int16 dist)

/*============================================================================*/
/* POMOCNÁ FUNKCIA: PeakCheck                                                 */
/*----------------------------------------------------------------------------*/
/* Opis: kontrola synchronizačných špičiek                                    */
/*                                                                            */
/* Vstup/Výstup:  SyncSub*     ssub    <-> synchronizačný subsystém           */
/* Vstup:         const char*  caller   -> identifikácia modemu               */
/*                const Bool*  pdet     -> vektor príznakov detekcie špičiek  */
/*                const Int16* p        -> vektor čísiel rámcov               */
/*                const Int32* corr(X)  -> vektor korelačných hodnôt          */
/*                Int16        pos1     -> pozícia špičky 1                   */
/*                Int16        pos2     -> pozícia špičky 2                   */
/*                Int16        npeaks   -> počet detegovaných špičiek         */
/*                Int16        delay    -> cieľové oneskorenie pri úspešnej   */
/*                                         synchronizácii                     */
/*----------------------------------------------------------------------------*/
void PeakCheck(SyncSub *ssub,
               const char *caller, const Bool *pdet, const Int16 *p,
               const Int32 *corrP, const Int32 *corrN, const Int32 *corr,
               Int16 pos1, Int16 pos2, Int16 npeaks, Int16 delay)
/*============================================================================*/
```

```
/* POMOCNÁ FUNKCIA: SyncReset                                                 */
/*----------------------------------------------------------------------------*/
/* Vstup/Výstup:  SyncState* sync      <-> stav synchronizácie                */
/* Vstup:         Int32*     mem        -> ukazovateľ synchronizačnej pamäte  */
/*                Int32*     memWakeup  -> ukazovateľ synchronizačnej         */
/*                                         aktivačnej pamäte                  */
/*----------------------------------------------------------------------------*/
void SyncReset(SyncState *sync, Int32 *mem, Int32 *memWakeup)
/*============================================================================*/
/* POMOCNÁ FUNKCIA: SyncSubReset                                              */
/*----------------------------------------------------------------------------*/
/* Vstup/Výstup:  SyncSub* ssub  <-> synchronizačný subsystém                 */
/*----------------------------------------------------------------------------*/
void SyncSubReset(SyncSub *ssub)
```

4.4.7 Funkcie riadiaceho spoja

```
/*============================================================================*/
/* POMOCNÁ FUNKCIA: CtrlPortName                                              */
/*----------------------------------------------------------------------------*/
/* Opis: konvertor na výstup záznamu                                          */
/*                                                                            */
/* Vstup:  PortOwner   owner  -> identifikácia modemu                         */
/* Návrat: const char*        <- názov portu ako reťazec znakov               */
/*----------------------------------------------------------------------------*/
const char* CtrlPortName(PortOwner owner)
/*============================================================================*/
/* RIADIACA FUNKCIA: CtrlTxProcess                                            */
/*----------------------------------------------------------------------------*/
/* Opis: funkcia procesov vysielača riadenia                                  */
/*                                                                            */
/* Vstup/Výstup:  CtrlTxData* control  <-> riadiaca štruktúra                 */
/*                Int16* pcm       -> rámec šestnásťbitových vzoriek PCM      */
/*----------------------------------------------------------------------------*/
```

```
void CtrlTxProcess(CtrlTxData *control, Int16 *pcm)
/*============================================================================*/
/* POMOCNÁ FUNKCIA: CtrlTxMod                                                 */
/*----------------------------------------------------------------------------*/
/* Opis: riadenie vysielača správ, ktorý používa vopred uložené postupnosti   */
/*                                                                            */
/* Vstup:     Int16  symbol  -> symbol správy nižšej alebo vyššej vrstvy      */
/*            Int16  index   -> pozícia v rámci správy                        */
/* Výstup:    Int16* pcm     <- výstupné dáta                                 */
/*----------------------------------------------------------------------------*/
void CtrlTxMod(Int16 *pcm, Int16 symbol, Int16 index)
/*============================================================================*/
/* RIADIACA FUNKCIA: CtrlRxProcess                                            */
/*----------------------------------------------------------------------------*/
/* Opis: riadiaca funkcia procesov prijímača                                  */
/*                                                                            */
/* Vstup/Výstup:  CtrlRxData*  control  <-> riadiaca štruktúra                */
/* Vstup:         const Int16* pcm  -> vstupný rámec šestnásťbitových vzoriek */
/*                                        PCM                                 */
/*----------------------------------------------------------------------------*/
void CtrlRxProcess(CtrlRxData *control, const Int16 *pcm)
/*============================================================================*/
/* POMOCNÁ FUNKCIA: CtrlRxDemod                                               */
/*----------------------------------------------------------------------------*/
/* Opis: riadenie prijímača správ                                             */
/*                                                                            */
/* Vstup:     const Int16* pcm     -> vstupný vyrovnávací zásobník PCM        */
/* Výstup:    Int16*       metric  <- činiteľ spoľahlivosti (-1: preskočiť)   */
/* Návrat:    Int16                <- demodulovaná správa                     */
/*----------------------------------------------------------------------------*/
Int16 CtrlRxDemod(const Int16 *pcm, Int16 *metric)
/*============================================================================*/
```

```
/* RIADIACA FUNKCIA: CtrlTxReset                                              */
/*----------------------------------------------------------------------------*/
/* Opis: funkcia resetovania riadenia vysielača                               */
/*                                                                            */
/* Vstup/Výstup:  CtrlTxData* control  <-> riadiaca štruktúra                 */
/* Vstup:         const char* owner     -> identifikácia modemu               */
/*----------------------------------------------------------------------------*/
void CtrlTxReset(CtrlTxData *control, const char *owner)
/*============================================================================*/
/* RIADIACA FUNKCIA: CtrlRxReset                                              */
/*----------------------------------------------------------------------------*/
/* Opis: funkcia resetovania riadenia prijímača                               */
/*                                                                            */
/* Vstup/Výstup:  CtrlRxData* control   <-> riadiaca štruktúra                */
/* Vstup:         const char* owner      -> identifikácia modemu              */
/*                SyncState*  sync       -> ukazovateľ synchronizačnej        */
/*                                          štruktúry                         */
/*                Int16*      buffer     -> ukazovateľ vyrovnávacieho         */
/*                                          zásobníka riadenia prijímača      */
/*                Int16       syncLock   -> počet požadovaných                */
/*                                          synchronizačných udalostí         */
/*----------------------------------------------------------------------------*/
void CtrlRxReset(CtrlRxData *control, const char *owner,
                 SyncState *sync, Int16 *buffer, Int16 syncLock)
```

4.4.8 Iné pomocné funkcie (IVS a PSAP)

```
/*============================================================================*/
/* POMOCNÁ FUNKCIA: SetModState                                               */
/*----------------------------------------------------------------------------*/
/* Opis: nastavenie stavu modulátora                                          */
/*                                                                            */
/* Vstup:         Int16     modType   -> typ použitého modulátora             */
/* Vstup/Výstup:  ModState* ms       <-> štruktúra modulátora                 */
```

```
/*-------------------------------------------------------------------------*/

void SetModState(ModState *ms, ModType modType)
```

Príloha A (informatívna)

História zmien

História zmien							
Dátum	**TSG SA#**	**TSG Doc.**	**CR**	**Rev**	**Predmet/Komentár**	**staré**	**nové**
2009-03	43	SP-090201			Schválené v TSG SA#43	2.0.0	8.0.0
2009-06	44	SP-090251	0001	1	Korekcia chaosu s 3GPP TS 26.267, ktorý sa týkal synchronizácie	8.0.0	8.1.0
2009-06	44	SP-090251	0002	1	Korekcia týkajúca sa inicializácie modulátora	8.0.0	8.1.0
2009-06	44	SP-090251	0003	1	Korekcia chaosu s 3GPP TS 26.267, ktorý sa týkal prenosu ACK	8.0.0	8.1.0
2009-06	44	SP-090251	0004	1	Rozšírenie nastavenia skúšky eCall, aby sa umožnilo skúšanie zhody správ ACK	8.0.0	8.1.0
2009-06	44	SP-090251	0005	2	Oddelenie funkcií vysielača a prijímača IVS a PSAP v kóde C	8.0.0	8.1.0
2009-09	45	SP-090565	0006	1	Integrácia potvrdzovacej správy vyššej vrstvy	8.1.0	8.2.0
2009-09	45	SP-090576	0007		Integrácia signalizačnej voľby, ktorú iniciuje IVS	8.1.0	8.2.0
2009-09	45	SP-090565	0008		Zmena parametra v nastavení skúšky eCall	8.1.0	8.2.0
2009-09	45	SP-090565	0009		Aktualizácia rozhraní medzi prijímačom a vysielačom na skúšanie zhody	8.1.0	8.2.0
2009-09	45	SP-090565	0010		Korekcie a opravy chýb v zdrojovom kóde C	8.1.0	8.2.0
2009-12	46				Verzia pre vydanie 9	8.2.0	9.0.0
2010-06	48	SP-100297	0012	1	Korekcia podmienok detekcie ACK	9.0.0	9.1.0
2010-06	48	SP-100297	0014	1	Detektor na obsluhu inverzie vzorky PCM v sieti	9.0.0	9.1.0
2010-06	48	SP-100297	0016	1	Modifikácie signálu spätnej väzby na zvýšenie robustnosti pri prítomnosti sieťových kompenzátorov ozveny	9.0.0	9.1.0
2010-09	49	SP-100462	0018		Korekcie niekoľkých chýb v referenčnom kóde eCall	9.1.0	9.2.0
2010-09	49	SP-100462	0020		Aktualizácia softvéru pracovného rámca skúšky eCall na obsluhu nových možností vnútropásmového modemu	9.1.0	9.2.0
2010-12	50	SP-100783	0022	1	Korekcia synchronizačných postupov vo vnútropásmovom modeme eCall	9.2.0	9.3.0
2010-12	50	SP-100783	0024	1	Korekcia stavu zariadenia vo vnútropásmovom modeme eCall	9.2.0	9.3.0
2010-12	50	SP-100783	0026	1	Korekcia detektora inverzie	9.2.0	9.3.0

2011-03	51	SP-110033	0028		Korekcia synchronizačného algoritmu pre vnútropásmový modem eCall	9.3.0	9.4.0
2011-03	51	SP-110033	0030		Korekcia operácie invertovania signálu a nastavenie udalosti správy spätného volania	9.3.0	9.4.0
2011-03	51				Verzia pre vydanie 10	9.4.0	10.0.0

História

História dokumentu		
V10.0.0	Apríl 2011	Vydanie