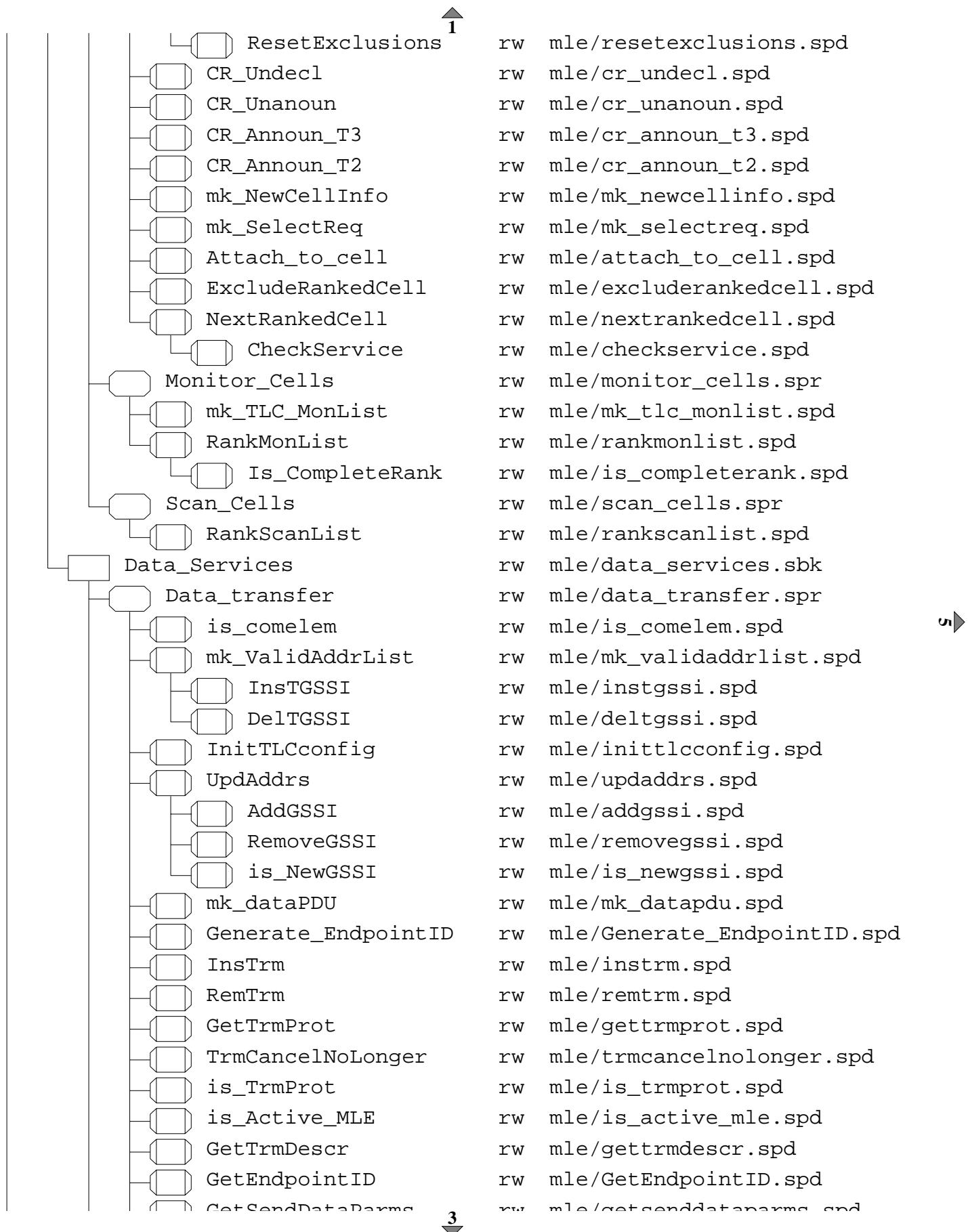
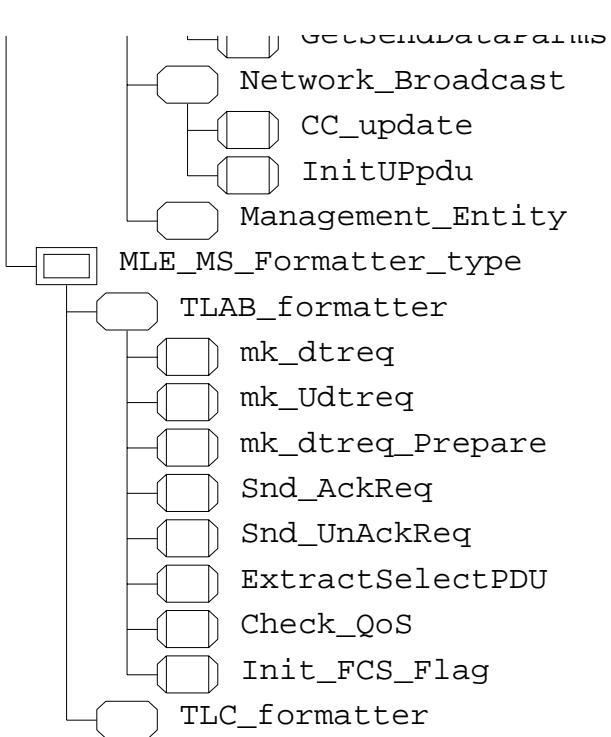


	SDT	rw /home/tetra/henry/working/common/mle.sdt
		rw /home/tetra/henry/working/
	AA_Common_Types	rw common/Common_Types.sun
	BA_Layer2_Service_Primitives	rw 12/L2servp.sun
	CA_Layer2_Private	rw 12/layer2_private.sun
	DA_MLE_Service_Primitives	rw common/MLE_Service_Primitives.
	DB_MLE_Protocol_Primitives	rw common/mlepdu.sun
	DC_MLE	rw common/mle.sun
	MLE_MS_Protocol_Type	rw mle/mle_ms_protocol_type1.sbt
	Attachment	rw mle/attachment.sbk
	CellSurveillance	rw mle/cellsurveillance.spr
	Attachment_Management	rw mle/attachment_management.spr
	SurveilInfo	rw mle/surveilinfo.spd
	GetScramblingCode	rw mle/GetScramblingCode.spd
	InitSupportedChannels	rw mle/initsupportedchannels.spd
	SetInitialCellSelectionState	rw mle/setinitialcellselect
	InScanList	rw mle/inscanlist.spd
	StartMonitoring	rw mle/startmonitoring.spd
	StartScanning	rw mle/startscanning.spd
	CellUpdateSIN1	rw mle/cellupdatesysinfo.spd
	CellUpdateSIN2	rw mle/cellupdatesync.spd
	UpdateCellList	rw mle/updatecelllist.spd
	InsertCellInfo	rw mle/insertcellinfo.spd
	CheckCriteria	rw mle/checkcriteria.spd
	IsInScanList	rw mle/isinscanlist.spd
	IsInMonList	rw mle/isinmonlist.spd
	InitiateReselection	rw mle/initiatereselection.spd
	CheckMSService	rw mle/checkmsservice.spd
	CompareService	rw mle/comparereservice.spd
	ChkRadioLinkFailure	rw mle/chkradiolinkfailure.spd
	SelectReselectionKind	rw mle/selectreselectionkind.spd
	InitialCellSelect	rw mle/initialcellselect.spd
	AllExcluded	rw mle/allexcluded.spd



rw	mle/resetexclusions.spd
rw	mle/cr_undecl.spd
rw	mle/cr_unanoun.spd
rw	mle/cr_announ_t3.spd
rw	mle/cr_announ_t2.spd
rw	mle/mk_newcellinfo.spd
rw	mle/mk_selectreq.spd
rw	mle/attach_to_cell.spd
rw	mle/excluderankedcell.spd
rw	mle/nextrankedcell.spd
rw	mle/checkservice.spd
rw	mle/monitor_cells.spr
rw	mle/mk_tlc_monlist.spd
rw	mle/rankmonlist.spd
rw	mle/is_completerank.spd
rw	mle/scan_cells.spr
rw	mle/rankscanlist.spd
rw	mle/data_services.sbk
rw	mle/data_transfer.spr
rw	mle/is_comelem.spd
rw	mle/mk_validaddrlist.spd
rw	mle/instgssi.spd
rw	mle/deltgssi.spd
rw	mle/inittlcconfig.spd
rw	mle/updadddrs.spd
rw	mle/addgssi.spd
rw	mle/removegssi.spd
rw	mle/is_newgssi.spd
rw	mle/mk_datapdu.spd
rw	mle/Generate_EndpointID.spd
rw	mle/instrm.spd
rw	mle/remtrm.spd
rw	mle/gettrmprot.spd
rw	mle/trmcancelnolonger.spd
rw	mle/is_trmprot.spd
rw	mle/is_active_mle.spd
rw	mle/gettrmdescr.spd
rw	mle/GetEndpointID.spd
rw	mle/getsenddataparms_and



2

rw	mle/generaldatapdu.spr
rw	mle/network_broadcast.spr
rw	mle/cc_updat.spd
rw	mle/inituppdu.spd
rw	mle/management_entity.spr
rw	mle/mle_formatter_type.sbt
rw	mle/tlab_formatter1.spr
rw	mle/mk_bldtreq.spd
rw	mle/mk_ubldtreq.spd
rw	mle/mk_dtreq_Preparespd
rw	mle/snd_bl_ackreq.spd
rw	mle/snd_bl_unackreq.spd
rw	mle/extractselectpdu.spd
rw	mle/check_qos.spd
rw	mle/init_fcs_flag.spd
rw	mle/tlc_formatter1.spr

sun



ionstate.spd

4

2

6

5

3



Package AA_Common_Types

1(16)



```
/* Options, parameters and constants. */
/*
The predefined values are given in the following format:
([D], [L]/[H]),
where
D is the default value,
L is the minimum (low range) value and
H is the maximum (high range).

All the fields in the formula may or may not be given depending
on the textual description defining the values.

All the values are the ones supported by the model in respect of
the defining text (e.g. considering the range of some constants).
*/
/* Timebase values for timer duration scaling. */

SYNONYM TIMESLOT Duration = 1; /* 14.67 ms */
SYNONYM TDMA_FRAME Duration = 4*TIMESLOT; /* 56.67 ms */
SYNONYM MULTIFRAME Duration = 18*TDMA_FRAME; /* 1.02 s */
SYNONYM SECOND Duration = 18*TDMA_FRAME; /* 17.65 TDMA frames*/

SYNONYM SIG_FRAME Duration = 4*TIMESLOT; /* 56.67 ms */
```



Package AA_Common_Types

2(16)



```
/*
**
** CMCE
**
*/
/* Number of concurrent instances of the CC service */
/* Always One in this model */
SYNONYM NO_OF_CC Natural = 1;

/* Circuit mode speech Supported TRUE (TRUE/FALSE) */
SYNONYM CIRCUIT_MODE_SPEECH_SUPPORTED Boolean = TRUE;

/* Circuit mode data Supported (TRUE/FALSE) */
SYNONYM CIRCUIT_MODE_DATA_SUPPORTED Boolean = TRUE;

/* DTMF Supported (TRUE/FALSE) */
SYNONYM DTMF_SUPPORTED Boolean = TRUE; /* EXTERNAL */

/* On/Off hook signalling supported (TRUE/FALSE) */
SYNONYM ON_OFF_HOOK_SUPPORTED Boolean = TRUE; /* EXTERNAL */

/* Direct setup signalling supported (TRUE/FALSE) */
SYNONYM DIRECT_CALL_SUPPORTED Boolean = TRUE; /* EXTERNAL */

/* Call SetUp Timer value for Called MS 1-30 Sec */
SYNONYM T_301 Duration = 30*SECOND; /* External */

/* Call SetUp Timer value for Calling MS 1-60 Sec */
SYNONYM T_302 Duration = 60*SECOND; /* External */

/* Call Initiated Timer value 1-60 Sec*/
SYNONYM T_303 Duration = 60*SECOND; /* External */

/* Break Resume Timer value 4-6 Sec*/
SYNONYM T_306 Duration = 5*SECOND; /* External */

/* Break Restore Timer value 6-8 Sec*/
SYNONYM T_307 Duration = 7*SECOND; /* External */

/* Call Disconnect Timer value MS 1-10 Sec*/
SYNONYM T_308 Duration = 10*SECOND; /* External */

/* Call Length Timer value 5-NoMax Sec*/
SYNONYM T_310 Duration = 900*SECOND; /* External */

/* Call Transmission Timer value 1-300 Sec*/
SYNONYM T_311 Duration = 30*SECOND; /* External */
```



Package AA_Common_Types

3(16)



```
/*
**
** MM
**
*/
/* Home ITSI Number MCC,MNC,SSI */
SYNONYM HOME_ITSI TSI_Type = (. 357,975,4545 .); /* EXTERNAL */

/* TETRA Equipment Identity 60 BITS */
SYNONYM TEI TEIType = 774488; /* EXTERNAL */

/* The following values are used in Class Of MS */
/***********************/

/* Duplex supported (TRUE/FALSE) */
SYNONYM DUPLEX_SUPPORTED Boolean = TRUE; /* EXTERNAL */

SYNONYM SINGLE_MULTI_SLOT_SUPPORTED Boolean = TRUE; /* EXTERNAL */
SYNONYM CONCURRENT_MULTI_CARRIER_OPERATION_SUPPORTED Boolean = FALSE; /* EXTERNAL */
SYNONYM END_TO_END_ENCRYPTION_SUPPORTED Boolean = FALSE; /* EXTERNAL */
SYNONYM AIR_INTERFACE_ENCRYPTION_SUPPORTED Boolean = FALSE; /* EXTERNAL */
SYNONYM CLCH_NEEDED_ON_CARRIER_CHANGE_SUPPORTED Boolean = FALSE; /* EXTERNAL */
SYNONYM CONCURRENT_CHANNELS_SUPPORTED Boolean = FALSE; /* EXTERNAL */
SYNONYM MINIMUM_MODE_SUPPORTED Boolean = TRUE; /* EXTERNAL */
SYNONYM CARRIER_SPECIFIC_SIGNALLING_CHANNEL_SUPPORTED Boolean = FALSE; /* EXTERNAL */

/* TETRA Air Interface Standard Version Number 0 (0/7) */
SYNONYM TETRA_AIR_INTERFACE_STANDARD_NUMBER_0TO7 = 0; /* EXTERNAL */
/***********************/

/* Registration Timer Sec*/
SYNONYM T_351 Duration = 30*SECOND; /* External */

/*
**
** MLE
**
*/
/* Cell re-selection preparation response time (in seconds)*/
SYNONYM T_370 Duration = 5*SECOND;
```



Package AA_Common_Types

4(16)



```
/*
**
** LLC
**
*/

/* Advanced link supported (FALSE, TRUE/FALSE) */
SYNONYM ADVANCED_LINK Boolean = TRUE;
SYNONYM BS_ADVANCED_LINK Boolean = TRUE;

/* Advanced link setup needed for unacknowledged AL (TRUE, TRUE/TRUE) */
SYNONYM AL_UNACK_SETUP_NEEDED Boolean = TRUE;

/* AL-DISC PDU as a response to an AL-SETUP-PDU supported in case
   advanced link is not supported (FALSE, TRUE/FALSE) */
SYNONYM UNSUPPORTED_ADVANCED_LINK_INDICATION = TRUE;
SYNONYM BS_UNsupported_ADVANCED_LINK_INDICATION = TRUE;

/* Received duplicates suppressed in Advanced Link unacknowledged service
   (TRUE, TRUE/FALSE) */
SYNONYM AL_UNACK_SUPPRESS_RECEIVED_DUPLICATES = TRUE;

/* Received SDUs delivered in SDU-numbering order in
   Advanced Link unacknowledged service (TRUE, TRUE/FALSE) */
SYNONYM AL_UNACK_DATA_DELIVERED_IN_ORDER = TRUE;

/* Advanced link supported (TRUE, TRUE/TRUE) */
SYNONYM PRIORITY_ORDERING Boolean = TRUE;

/* Advanced link supported (TRUE, TRUE/TRUE) */
SYNONYM CANCEL_OPERATION Boolean = TRUE;

/* Advanced link supported (FALSE, FALSE/False) */
SYNONYM PRE_EMPTIVE_CANCEL Boolean = FALSE;

/* Service request queue size for all link instances (8, 1/ Platform dependent)
SYNONYM QUEUE_SIZE Natural = 8;

/* Optional FCS in Basic Link (TRUE, TRUE/TRUE) */
SYNONYM BL_FCS Boolean = TRUE;

/* Maximum number of simultaneous link instances (1, 1/1) */
SYNONYM MAX_CONCURRENT_LINKS Natural = 1;

/* Initial timer durations for LLC, Part 2, Annex A. The unit of
   timer values is next available signalling opportunity */

/* Basic Link timers */

/* Sender retry timer (4, -/-) */
SYNONYM T_251 DURATION = 4*SIG_FRAME;
```



Package AA_Common_Types

5(16)



```
/* Basic Link constants */
/* Maximum length of TL-SDU (2595, -/-) */
SYNONYM N251 Natural = 2595;

/* Maximum number of TL-SDU retransmissions for acknowledged service (1/5, 3/5 ) */
SYNONYM N252_NO_STEALING_REPEATS Natural = 1;
SYNONYM N252_WITH_STEALING_REPEATS Natural = 3;

/* Number of TL-SDU repetitions for unacknowledged service (1/5) */
SYNONYM N253 Natural = 1;

/* Advanced Link constants */

/* Advanced link number (1/4) */
SYNONYM N261 Natural = 1;

/* Maximum number of connection setup retries (1/5) */
SYNONYM N262 Natural = 3;

/* Maximum number of disconnection retries (3/5) */
SYNONYM N263 Natural = 3;

/* Maximum Number of timeslots used per TDMA frame (1/4) */
SYNONYM N264 MaxTransmissionRateType = 1;

/* Maximum length of TL-SDU (4096) */
SYNONYM N271 MaxLengthOf_TL_SDU_Type = 4096_O;

/* Window size for TL-SDU in acknowledged service (1/3) */
SYNONYM N272 Natural = 3;

/* Maximum number of TL-SDU retransmissions (0/7) */
SYNONYM N273 Natural = 3;

/* Maximum number of segment retransmissions (0/15) */
SYNONYM N274 Natural = 3;

/* Window size for TL-SDU in unacknowledged service (1/3) */
SYNONYM N281 Natural = 3;

/* Number of repetitions for unacknowledged information (0/7) */
SYNONYM N282 Natural = 1;

/* Advanced Link timers */

/* Acknowledgement waiting timer (9, -/-) */
SYNONYM T_252 DURATION = 9*SIG_FRAME;

/* Setup waiting timer (4, -/-) */
SYNONYM T_261 DURATION = 4*SIG_FRAME;

/* Disconnection waiting timer (4, -/-) */
SYNONYM T_263 DURATION = 4*SIG_FRAME;

/* Receiver not ready validity timer for the data sending entity (36, -/-) */
SYNONYM T_271 DURATION = 36*TDMA_FRAME;

/* Receiver not ready validity timer for the data receiving entity (18, -/-) */
SYNONYM T_272 DURATION = 18*TDMA_FRAME;
```



Package AA_Common_Types

6(16)



```
/*
**
** CONP
**
*/
/* CONP supported (FALSE, TRUE/FALSE) */
SYNONYM CONP_SUPPORTED Boolean = TRUE;

/* is The calling entity supporting the fastSelect (TRUE, TRUE/FALSE)*/
SYNONYM CALLING_FAST_SELECT Boolean = TRUE;

/* is The called entity supporting the fastSelect (TRUE, TRUE/FALSE)*/
SYNONYM CALLED_FAST_SELECT Boolean = TRUE;

/*
**
** S-CLNP
**
*/
/* S_CLNP supported (TRUE, TRUE/FALSE) */
SYNONYM SCLNP_SUPPORTED Boolean = TRUE;

/* SCLNP re-sends packet after MLE_CLOSE - MLE_OPEN (FALSE, TRUE/FALSE) */
SYNONYM SCLNP_RESENGS_AFTER_CLOSE Boolean = FALSE;
```



Package AA_Common_Types

7(16)



```
/*
**
** MAC
**
*/

/* Event label inactivity time-out (30, -/-) */
SYNONYM T_201 DURATION = 30*MULTIFRAME; /* External */

/* Fragmentation time-out (9, -/-) /* Downlink signalling frames */
SYNONYM T_202 DURATION = 9*SIG_FRAME; /* External */

/* Random Access time-out (5, 5/60) */
SYNONYM T_205 DURATION = 5*MULTIFRAME;

/* Reserved Access waiting time-out (18, -/-) /* Downlink signalling frames */
SYNONYM T_206 DURATION = 18*SIG_FRAME; /* External */

/* Inactivity time-out on SCCH (30, -/-) */
SYNONYM T_208 DURATION = 30*MULTIFRAME; /* External */

/* Inactivity time-out on traffic channel (18, -/-) */
SYNONYM T_209 DURATION = 18*MULTIFRAME; /* External */

/* Timer for returning to energy economy mode (18, -/-) /* TDMA frames */
SYNONYM T_210 DURATION = 18*TDMA_FRAME; /* External */

/* ACCH time-out for transmission of TCH (36, -/-) /* TDMA frames */
SYNONYM T_211 DURATION = 36*TDMA_FRAME; /* External */

/* ACCH time-out for reception of TCH (18, -/-) /* TDMA frames */
SYNONYM T_212 DURATION = 18*TDMA_FRAME; /* External */

/* DTX timer (18, -/-) /* TDMA frames */
SYNONYM T_213 DURATION = 18*TDMA_FRAME; /* External */

/* Stealing timer (6, -/-) /* uplink opportunities */
SYNONYM T_214 DURATION = 6*SIG_FRAME; /* External */

/* MAC Constants */
/* Maximum size of TM-SDU (2632, 1/2632) /* Bits */
SYNONYM N202 Natural = 2632;

/* Number of wrong AACHs to leave assigned channel (3, -/-) */
SYNONYM N208 Natural = 3; /* External */

/* Quality threshold for serving cell (4, -/-) */
SYNONYM N210 Natural = 4; /* External */

/* Number of invalid AACHs to stop transmission of TCH (3, -/-) */
SYNONYM N211 Natural = 3; /* External */

/* Number of invalid AACHs to stop reception of TCH (3, -/-) */
SYNONYM N212 Natural = 3; /* External */

/* Number of valid AACHs to allow reception of TCH (3, -/-) */
SYNONYM N213 Natural = 3; /* External */

/* Number of transmissions if stealing repeats flag is set (4, -/-) */
SYNONYM N214 Natural = 4; /* External */
```



Package AA_Common_Types

8(16)



```
/* Types for validation models (IDEAL, PRESETTABLE, SELECTABLE) */

SYNONYM IDEAL Natural = 1;
SYNONYM NON_DETERMINISTIC Natural = 2;
SYNONYM SELECTABLE Natural = 3;

SYNONYM LLC_VALIDATION_MODEL Natural = IDEAL;
SYNONYM SCLNP_VALIDATION_MODEL Natural = SELECTABLE;
```

```
/* Definition of Natural Constants */
SYNTYPE 0TO1 = /* 1 bit */
    Natural CONSTANTS 0:1
ENDSYNTYPE;

SYNTYPE 0TO3 = /* 2 bits */
    Natural CONSTANTS 0:3
ENDSYNTYPE;

SYNTYPE 0TO7 = /* 3 bits */
    Natural CONSTANTS 0:7
ENDSYNTYPE;

SYNTYPE 0TO15 = /* 4 bits */
    Natural CONSTANTS 0:15
ENDSYNTYPE;

SYNTYPE 0TO31 = /* 5 bits */
    Natural CONSTANTS 0:31
ENDSYNTYPE;

SYNTYPE 0TO63 = /* 6 bits */
    Natural CONSTANTS 0:63
ENDSYNTYPE;

SYNTYPE 0TO127 = /* 7 bits */
    Natural CONSTANTS 0:127
ENDSYNTYPE;

SYNTYPE 0TO255 =
    Natural CONSTANTS 0:255
ENDSYNTYPE;

SYNTYPE 0TO1023 = /* 10 bits */
    Natural CONSTANTS 0:1023
ENDSYNTYPE;

SYNTYPE 0TO2047 = /* 11 bits */
    Natural CONSTANTS 0:2047
ENDSYNTYPE;

SYNTYPE 0TO4095 = /* 12 bits */
    Natural CONSTANTS 0:4095
ENDSYNTYPE;
```

```
SYNTYPE 0TO16383 = /* 14 bits */
    Natural CONSTANTS 0:16383
ENDSYNTYPE;

SYNTYPE 0TO65535 = /* 16 bits */
    Natural CONSTANTS 0:65535
ENDSYNTYPE;

SYNTYPE 0TO1048575 = /* 20 bits */
    Natural CONSTANTS 0:1048575
ENDSYNTYPE;

SYNTYPE 0TO16777215 = /* 24 bits */
    Natural CONSTANTS 0:16777215
ENDSYNTYPE;
```

```
SYNONYM BITS_IN_OCTET Natural = 8;
```



Package AA_Common_Types

9(16)



```
/* Definition of TSI types from
   ETS 300 392-1 Clause 7.2 */

SYNTYPE MCC_Type = /* 10 bits */
  Natural CONSTANTS 0:999
ENDSYNTYPE;

SYNTYPE MNC_Type = /* 14 bits */
  OCTO16383
ENDSYNTYPE;

NEWTYPE TSI_ExtensionType STRUCT
  MCC MCC_Type;
  MNC MNC_Type;
ENDNEWTYPE;

NEWTYPE TSI_ExtensionType2 STRUCT
  Present Boolean;
  TSI_Extension TSI_ExtensionType;
ENDNEWTYPE;

SYNTYPE SSI_Type = /* 24 BITS */
  OCTO16777215
ENDSYNTYPE;

SYNONYM BROADCAST_ADDRESS SSI_Type =
  16777215; /* All One's */

NEWTYPE SSI_Type2 STRUCT
  Present Boolean;
  SSI SSI_Type;
ENDNEWTYPE;

NEWTYPE TSI_Type STRUCT
  MCC MCC_Type;
  MNC MNC_Type;
  SSI SSI_Type;
ENDNEWTYPE;

SYNTYPE ITSI_Type =
  TSI_Type
ENDSYNTYPE;

SYNTYPE ATSI_Type =
  TSI_Type
ENDSYNTYPE;

SYNTYPE GTSI_Type =
  TSI_Type
ENDSYNTYPE;

SYNTYPE ISSI_Type =
  SSI_Type
ENDSYNTYPE;

SYNTYPE ASSI_Type =
  SSI_Type
ENDSYNTYPE;
```

```
SYNTYPE GSSI_Type =
  SSI_Type
ENDSYNTYPE;

NEWTYPE GSSI_Type2 STRUCT
  Present Boolean;
  GSSI GSSI_Type;
ENDNEWTYPE;

SYNTYPE USSI_Type =
  SSI_Type
ENDSYNTYPE;

/* Definition of TMI types from
   ETS 300 392-1 Clause 7.3 */

SYNTYPE SMI_Type = /* 24 BITS */
  SSI_Type
ENDSYNTYPE;

NEWTYPE TMI_Type STRUCT
  MCC MCC_Type;
  MNC MNC_Type;
  SMI SMI_Type;
ENDNEWTYPE;

/* Definition of MNI types from
   ETS 300 392-1 Clause 7.6 */

NEWTYPE MNI_Type STRUCT
  MCC MCC_Type;
  MNC MNC_Type;
ENDNEWTYPE;
```



Package AA_Common_Types

10(16)



```
/* Parameter Definitions for Service Primitives, clause 11 */
NEWTYPE CommunicationTypeType LITERALS
  POINT_TO_POINT,
  POINT_TO_MULTIPOINT,
  POINT_TO_MULTIPOINT_ACK,
  BROADCAST
ENDNEWTYPE CommunicationTypeType;

NEWTYPE CommunicationTypeType2 STRUCT
  Present Boolean;
  CommunicationType CommunicationTypeType;
ENDNEWTYPE;

/* Parameter Definitions for Service Primitives, clause 16.10 */

/* Clause 16.10.11 */
SYNTYPE FrameType =
  0TO31
ENDSYNTYPE;

/* Clause 16.10.30, Length = 14 bits */
SYNTYPE LocationAreaType = 0TO16383
ENDSYNTYPE;

/* Clause 16.10.38 */
SYNTYPE MultiFrameType =
  0TO63
ENDSYNTYPE;

/* Clause 16.10.41 */
SYNTYPE ProprietaryType =
  Natural
ENDSYNTYPE;

NEWTYPE ProprietaryType3 STRUCT
  ElementID Natural;
  Length Natural;
  Proprietary ProprietaryType;
  M_Bit Boolean;
ENDNEWTYPE;

/* Clause 16.10.50 */
SYNTYPE TEIType = Natural
ENDSYNTYPE;

/* Parameter Definition for service primitives, clause 18.5 */

/* Clause 18.5.10 */
SYNTYPE MainCarrierNoType =
  0TO4095 /* 12 bit */
ENDSYNTYPE;
```



Package AA_Common_Types

11(16)



```
/* Clause 18.5.11 */
NEWTYPE MainCarrierNoExtType STRUCT
    FrequencyBand 0TO15;
    Offset 0TO3;
    DuplexSpacing 0TO7;
    ReverseOperation 0TO1;
ENDNEWTYPE MainCarrierNoExtType;

NEWTYPE MainCarrierNoExtType2 STRUCT
    Present Boolean;
    MainCarrierNoExt MainCarrierNoExtType;
ENDNEWTYPE MainCarrierNoExtType2;

/* Clause 18.5.3 */
SYNTYPE CellIdentifierType =
    0TO31 /* 5 bit */;
ENDSYNTYPE;

/* Parameter definitions for service primitives, clause 20.2.4 */

/* Clause 20.2.4.1 */
NEWTYPE AddrTypeType
    LITERALS
        T_ISSI, T_ASSI, T_USSI, T_SMI, T_GSSI;
ENDNEWTYPE;

/* Clause 20.2.4.2 */
NEWTYPE ChanType STRUCT
    MainCarrierNo MainCarrierNoType;
    MainCarrierNoExt MainCarrierNoExtType2;
ENDNEWTYPE ChanType;

SYNTYPE ChanListIndex =
    Natural CONSTANTS 0:5
ENDSYNTYPE ChanListIndex;

/* Clause 20.2.4.2 */
NEWTYPE ChanListType
    ARRAY(ChanListIndex, ChanType);
ENDNEWTYPE;

NEWTYPE ChannelArrayType STRUCT
    Length Natural;
    ChanList ChanListType;
ENDNEWTYPE;

/* Clause 20.2.4.3 */
SYNTYPE ChannelChangeAcceptedType =
    Boolean
ENDSYNTYPE;

/* Clause 20.2.4.4 */
SYNTYPE ChannelChangeResponseRequiredType =
    Boolean
ENDSYNTYPE;

/* Clause 20.2.4.5 and 16.10.8 */
NEWTYPE DistributionOn18thFrameType
    LITERALS
        TimeSlot1,
        TimeSlot2,
        TimeSlot3,
        TimeSlot4;
ENDNEWTYPE;
```



Package AA_Common_Types

12(16)



```
/* Clause 20.2.4.6 */
NEWTYPE EncryptionType
LITERALS
    ENCRYPTED,
    CLEAR;
ENDNEWTYPE;

NEWTYPE EncryptionType2 STRUCT
    Present Boolean;
    Encryption EncryptionType;
ENDNEWTYPE;

/* Clause 20.2.4.7 */
SYNTYPE Endpoint_ID_Type =
    Natural
ENDSYNTYPE;

/* Clause 20.2.4.8 */
NEWTYPE EnergySavingModeType
LITERALS
    NO_EG, EG1, EG2,
    EG3, EG4, EG5, EG6, EG7;
ENDNEWTYPE;

NEWTYPE EnergySavingModeType2 STRUCT
    Present Boolean;
    EnergySavingMode
        EnergySavingModeType;
ENDNEWTYPE;

/* Clause 20.2.4.9 */
NEWTYPE EnergyEconomyStartpointType STRUCT
    Frame FrameType;
    MultiFrame MultiFrameType;
ENDNEWTYPE;

/* Clause 20.2.4.10 */
SYNTYPE FCS_FlagType =
    Boolean
ENDSYNTYPE;

/* Clause 20.2.4.11 */
NEWTYPE GroupCallReleaseType
LITERALS
    STAY,
    LEAVE;
ENDNEWTYPE;

/* Type for link identification in handle */
NEWTYPE Link_ID_Type
LITERALS
    NULL,
    ACK,
    UNACK,
    L2;
ENDNEWTYPE;
```



```
/* Clause 20.2.4.18 */
NEWTYPE MainAddrType STRUCT
    Addr SSI_Type;
ENDNEWTYPE;

/* Clause 20.2.4.19 */
NEWTYPE MLE_ActivityType
LITERALS
    MLE_ACTIVE,
    MLE_NOT_ACTIVE;
ENDNEWTYPE;

/* Clause 20.2.4.20 */
SYNTYPE NewEndpoint_ID_Type =
    Endpoint_ID_Type
ENDSYNTYPE;

/* Clause 20.2.4.21 */
SYNTYPE NumberOfTimeslotsType =
    Natural
ENDSYNTYPE;

/* Clause 20.2.4.22 */
NEWTYPE U_PlaneFlagType
LITERALS
    U_PLANE_ON,
    U_PLANE_OFF;
ENDNEWTYPE;

/* Clause 20.2.4.22 */
NEWTYPE TX_GrantFlagType
LITERALS
    TX_GRANTED,
    TX_CEASED;
ENDNEWTYPE;

/* Clause 20.2.4.22 */
NEWTYPE DuplexFlagType
LITERALS
    SIMPLEX,
    DUPLEX;
ENDNEWTYPE;

NEWTYPE DuplexFlagType2 STRUCT
    Present Boolean;
    DuplexFlag DuplexFlagType;
ENDNEWTYPE;

/* Clause 20.2.4.22 */
SYNTYPE InterleavingDepthType =
    Natural
ENDSYNTYPE;
```



Package AA_Common_Types

13(16)



```
/* Clause 20.2.4.22 */
NEWTYPE CircuitModeType
LITERALS
  TCH_S,
  TCH_7_2,
  TCH4.8_N1,
  TCH4.8_N4,
  TCH4.8_N8,
  TCH2.4_N1,
  TCH2.4_N4,
  TCH2.4_N8;
ENDNEWTYPE CircuitModeType;

/* Clause 20.2.4.22 */
NEWTYPE OperatingModeType STRUCT
  U_PlaneFlag U_PlaneFlagType;
  TX_GrantFlag TX_GrantFlagType;
  DuplexFlag DuplexFlagType;
  TypeOfCircuit CircuitModeType;
  Encryption EncryptionType;
  Endpoint_ID Endpoint_ID_Type;
ENDNEWTYPE;

/* Clause 20.2.4.23 */
SYNTYPE PathLossC1Type =
  Natural
ENDSYNTYPE;

/* Clause 20.2.4.23 */
SYNTYPE PathLossC2Type =
  Natural
ENDSYNTYPE;

/* Clause 20.2.4.23 */
NEWTYPE PathLossType
STRUCT
  C1 PathLossC1Type;
  C2 PathLossC2Type;
ENDNEWTYPE;

/* Clause 20.2.4.24 */
SYNTYPE PDU_PriorityType =
  Natural
  CONSTANTS 0:7
ENDSYNTYPE;

SYNONYM MIN_PDU_PRIORITY
PDU_PriorityType = 0;

SYNONYM MAX_PDU_PRIORITY
PDU_PriorityType = 7;

SYNONYM EMERGENCY_PRIORITY
PDU_PriorityType = MAX_PDU_PRIORITY;

/* Clause 20.2.4.25 */
SYNTYPE QualityIndicationType =
  Natural
ENDSYNTYPE;

/* Clause 20.2.4.26 */
SYNTYPE MaxTransmissionRateType =
  Natural
  CONSTANTS 1:4
  /* The number of timeslots used */
ENDSYNTYPE;

/* Clause 20.2.4.26 */
NEWTYPE MeanTransmissionRateType
LITERALS
  NetworkDependent,
  1_32,
  1_16,
  1_8,
  1_4,
  1_2,
  Maximum;
  /* The notation 1_32 = 1/32 */
ENDNEWTYPE;

/* Clause 20.2.4.26 */
NEWTYPE MaxLengthOf_TL_SDU_Type
LITERALS
  32_O,
  64_O,
  128_O,
  256_O,
  512_O,
  1024_O,
  2048_O,
  4096_O;
  /* Notation 32_O = 32 octets */
OPERATORS
  ORDERING
ENDNEWTYPE;

/* Clause 20.2.4.26 */
SYNTYPE TL_SDU_WindowSizeType =
  Natural
  CONSTANTS 1:3
  /* Maximum window size is LLC
   parameter N.272 */
ENDSYNTYPE;

/* Clause 20.2.4.26 */
NEWTYPE ThroughputType STRUCT
  MaxTransmissionRate
  MeanTransmissionRate
  MeanTransmissionRateType;
  MaxLengthOf_TL_SDU
  MaxLengthOf_TL_SDU_Type;
  TL_SDU_WindowSize
  TL_SDU_WindowSizeType;
ENDNEWTYPE;

/* Clause 20.2.4.26 */
SYNTYPE Max_TL_SDU_ReTransmissionsType =
  Natural
  CONSTANTS 0:7
ENDSYNTYPE;
```



Package AA_Common_Types

14(16)



```
/* Clause 20.2.4.26 */
SYNTYPE MaxSegmentReTransmissionsType =
    Natural
        CONSTANTS 0:15
ENDSYNTYPE;

/* Clause 20.2.4.26 */
NEWTYPE TransferFailureProbabilityType STRUCT
    Max_TL_SDU_ReTransmissions Max_TL_SDU_ReTransmissionsType;
    MaxSegmentReTransmissions MaxSegmentReTransmissionsType;
ENDNEWTYPE;

/* Clause 20.2.4.26 */
NEWTYPE QoS_Type STRUCT
    Throughput ThroughputType;
    TransferFailureProbability TransferFailureProbabilityType;
ENDNEWTYPE;

/* Clause 20.2.4.27 */
NEWTYPE ReportType
    LITERALS
        HANDLE,
        SERVICE_DEFINITION,
        SERVICE_CHANGE,
        RESET_REPORT,
        SETUP_FAILURE,
        SERVICE_TEMPORARILY_NOT_SUPPORTED,
        SERVICE_NOT_SUPPORTED,
        REJECTION,
        CLOSE,
        INCOMING_DISCONNECTION,
        DISCONNECTION_FAILURE,
        LOCAL_DISCONNECTION,
        FIRST_COMPLETE_TRANSMISSION,
        SUCCESSFUL_TRANSFER,
        FAILED_TRANSFER,
        ABORTED_SDU_NOT_COMPLETELY_SENT,
        ABORTED_SDU_SENT_AT_LEAST_ONCE,
        LAYER_TWO_TRANSMISSION_ACTIVITIES_CONTINUING,
        FIRST_COMPLETE_TRANSMISSION_BY_RANDOM_ACCESS,
        SUCCESSFUL_COMPLETE_TRANSMISSION_BY_RANDOM_ACCESS,
        COMPLETE_TRANSMISSION_BY_STEALING_OR_BY_RESERVED_ACCESS,
        RANDOM_ACCESS_FAILURE,
        FRAGMENTATION_FAILURE;
    ENDNEWTYPE;

/* Derived definition from Clause xxx, proposed by PT 93 */
NEWTYPE TLC_ReportType
    LITERALS
        USAGE_MARKER_MISMATCH,
        DOWNLINK_FAILURE,
        UPLINK_FAILURE,
        MAXIMUM_PATH_DELAY_EXCEEDED;
    ENDNEWTYPE;
```



Package AA_Common_Types

15(16)



```
/* Clause 20.2.4.28 */
NEWTYPE ScanningMeasurementMethodType
  LITERALS
    FOREGROUND,
    BACKGROUND,
    INTERRUPTING;
ENDNEWTYPE;

/* Clause 20.2.4.29 and 16.10.44 */
SYNTYPE SCCH_InformationType =
  OTO15
ENDSYNTYPE;

/* Clause 20.2.4.30 */
SYNTYPE ScramblingCodeType =
  MNI_Type
ENDSYNTYPE;

/* Clause 20.2.4.31 */
NEWTYPE SetupReportType
  LITERALS
    SUCCESS,
    SERVICE_DEFINITION,
    SERVICE_CHANGE,
    RESET_REPORT;
ENDNEWTYPE;

NEWTYPE DisconnectionReportType
  LITERALS
    SUCCESS,
    CLOSE,
    REJECT,
    SERVICE_NOT_SUPPORTED,
    SERVICE_TEMPORARILY_UNAVAILABLE;
ENDNEWTYPE;

NEWTYPE LinkServiceType
  LITERALS
    UNACKNOWLEDGED,
    ACKNOWLEDGED;
ENDNEWTYPE;

/* Clause 20.2.4.35, 18.5.22 */
SYNTYPE SubscriberClassType =
  OT065535
ENDSYNTYPE;
```

```
NEWTYPE SubscriberClassType2 STRUCT
  Present Boolean;
  SubscriberClass SubscriberClassType;
ENDNEWTYPE;

SYNONYM ALL_SUBSCRIBERCLASS_ALLOWED
  SubscriberClassType = 65535;

/* Clause 20.2.4.36 */
SYNTYPE ThresholdLevelType =
  Natural
ENDSYNTYPE;

/* Clause 20.2.4.37 */
SYNTYPE ThresholdValuesType =
  Natural
ENDSYNTYPE;

/* Clause 20.2.4.38 */
SYNTYPE TL_SDU_IndexType =
  Natural
  CONSTANTS 1:110 /* N.271 */
ENDSYNTYPE;

NEWTYPE TL_SDU_ArrayType
  ARRAY(TL_SDU_IndexType, Natural)
ENDNEWTYPE;

NEWTYPE TL_SDU_Type
  STRUCT
    A TL_SDU_ArrayType := (. 0 .);
  /*#ADT(W(B))
  #BODY
  #ifdef XREADANDWRITEF
extern char * yWri_(#(tl_sdu_type))( void * Va
{
  static char xCharTmp[30];
  sprintf(xCharTmp,
    "(. TL-SDU : %d %d %d %d %d .) ",
    (*(#(tl_sdu_type) *)Value).a.A[0],
    (*(#(tl_sdu_type) *)Value).a.A[1],
    (*(#(tl_sdu_type) *)Value).a.A[2],
    (*(#(tl_sdu_type) *)Value).a.A[3],
    (*(#(tl_sdu_type) *)Value).a.A[4]);
  return xCharTmp;
}
#endif
*/
ENDNEWTYPE;

/* Clause 20.2.4.39 */
SYNTYPE TL_SDU_LengthType =
  Natural /* N.271 */
ENDSYNTYPE;

NEWTYPE ValidAddressType STRUCT
  AddrKind AddrTypeType;
  Addr SSI_Type;
ENDNEWTYPE;
```



Package AA_Common_Types

16(16)



```
/* Clause 20.2.4.42 */
SYNTYPE ValidAddressesIndexType =
    Natural CONSTANTS 0:7
ENDSYNTYPE;

NEWTYPE ValidAddrList
    ARRAY(ValidAddressesIndexType,
        ValidAddressType);
ENDNEWTYPE;

/* Clause 20.2.4.42 */
NEWTYPE ValidAddressesType STRUCT
    Length Natural;
    Element ValidAddrList;
ENDNEWTYPE ValidAddressesType;
```

```
USE AA_Common_Types;
```

Package BA_Layer2_Service_Primitives

1 (7)

```
/* TLA-SAP Service Primitives */
```

```
NEWTYPE TransferReportType
LITERALS
    TRANSFER_REPORT_OK,
    TRANSFER_REPORT_BS_DL_NO_RX_UR_DD1,
    TRANSFER_REPORT_BS_DL_NO_RX_UR_DD2,
    TRANSFER_REPORT_BS_UL_NO_RX_UD2_DR1,
    TRANSFER_REPORT_BS_UL_NO_RX_UD2_DR2,
    TRANSFER_REPORT_MS_DL_NO_RX_DD2,
    TRANSFER_REPORT_MS_UL_NO_RX_DR1,
    TRANSFER_REPORT_MS_UL_NO_RX_DR2,
    TRANSFER_REPORT_L2_SDU_QUEUE_FULL,
    TRANSFER_REPORT_CANCEL_FAILED_UNKNOWN_ENDPOINT,
    TRANSFER_REPORT_CANCEL_PERFORMED;
ENDNEWTYPE;
```

```
/* Clause 19.2.3.2.2 + PT93 recommendation */
NEWTYPE TLA_DataRequestType STRUCT
    AddrType AddrTypeType;
    MainAddr MainAddrType;
    EndpointID Endpoint_ID_Type;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
    PDU_Priority PDU_PriorityType;
    ScramblingCode ScramblingCodeType;
    SubscriberClass SubscriberClassType;
    FCS_Flag FCS_FlagType;

    ProtocolReq Layer2ProtocolRequestType;
    MaxNumAccessAttempts AccessAttemptType;
ENDNEWTYPE;
```

```
/* Clause 19.2.3.2.2 + PT93 recommendation */
NEWTYPE TLA_DataIndicationType STRUCT
    AddrType AddrTypeType;
    MainAddr MainAddrType;
    EndpointID Endpoint_ID_Type;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
    PDU_Priority PDU_PriorityType;
    FCS_Flag FCS_FlagType;

    Report TransferReportType;
    NumAccessAttempts AccessAttemptType;
ENDNEWTYPE;
```

```
/* Clause 19.2.3.2.2 */
NEWTYPE TLA_DataConfirmType STRUCT
    AddrType AddrTypeType;
    MainAddr MainAddrType;
    EndpointID Endpoint_ID_Type;
    Report TransferReportType;
ENDNEWTYPE;
```

```
NEWTYPE Layer2ProtocolRequestType
LITERALS
    PROTOCOL_REQ_ACK_ALL_BLOCKS_RETRANS,
    PROTOCOL_REQ_ACK_NO_BLOCKS_RETRANS,
    PROTOCOL_REQ_UNACK;
ENDNEWTYPE;
```

```
SYNTYPE AccessAttemptType =
    0TO15
ENDSYNTYPE;
```

```
/* Clause ???, this SP is not specified in the ETS, however it
may be necessary to handle the MLE_Cancel_Req */
NEWTYPE TLA_CancelRequestType STRUCT
    EndpointID Endpoint_ID_Type;
ENDNEWTYPE;
```

```

/* Clause 19.2.3.2.2 + PT93 recommendation */
NEWTYPE TLA_DataRequestType STRUCT
    AddrType AddrTypeType;
    MainAddr MainAddrType;
    EndpointID Endpoint_ID_Type;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
    PDU_Priority PDU_PriorityType;
    ScramblingCode ScramblingCodeType;
    SubscriberClass SubscriberClassType;
    FCS_Flag FCS_FlagType;

    ProtocolReq Layer2ProtocolRequestType;
    MaxNumAccessAttempts AccessAttemptType;
ENDNEWTYPE;

/* Clause 19.2.3.2.2 + PT93 recommendation */
NEWTYPE TLA_DataIndicationType STRUCT
    AddrType AddrTypeType;
    MainAddr MainAddrType;
    EndpointID Endpoint_ID_Type;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
    PDU_Priority PDU_PriorityType;
    FCS_Flag FCS_FlagType;

    Report TransferReportType;
    NumAccessAttempts AccessAttemptType;
ENDNEWTYPE;

/* Clause 19.2.3.2 */
NEWTYPE TLA_DataConfirmType STRUCT
    AddrType AddrTypeType;
    MainAddr MainAddrType;
    EndpointID Endpoint_ID_Type;
    Report TransferReportType;
ENDNEWTYPE;

```

```
USE AA_Common_Types;
```

Package BA_Layer2_Service_Primitives

2(7)

```
/* Clause 19.2.3.2.3 */
NEWTYPE TLA_UnitdataRequestType STRUCT
  AddrType AddrTypeType;
  MainAddr MainAddrType;
  EndpointID_Present Boolean;
  EndpointID Endpoint_ID_Type;
  TL_SDU TL_SDU_Type;
  TL_SDU_Length TL_SDU_LengthType;
  PDU_Priority PDU_PriorityType;
  ScramblingCode ScramblingCodeType;
  SubscriberClass SubscriberClassType;
  FCS_Flag FCS_FlagType;
ENDNEWTYPE;

/* Clause 19.2.3.2.3 */
NEWTYPE TLA_UnitdataIndicationType STRUCT
  AddrType AddrTypeType;
  MainAddr MainAddrType;
  EndpointID Endpoint_ID_Type;
  TL_SDU TL_SDU_Type;
  TL_SDU_Length TL_SDU_LengthType;
  PDU_Priority PDU_PriorityType;
  FCS_Flag FCS_FlagType;
ENDNEWTYPE;

/* Clause 19.2.3.2.3 */
NEWTYPE TLA_UnitdataConfirmType STRUCT
  AddrType AddrTypeType;
  MainAddr MainAddrType;
  EndpointID Endpoint_ID_Type;
  Report TransferReportType;
ENDNEWTYPE;
```

```
/* External synonyms for Selectable options, which are used if
   ValidationOption = 2
*/
SYNONYM EXT_CANCEL_POSSIBLE Boolean= TRUE;
SYNONYM EXT_TRANSMISSION_SUCCESS Boolean=TRUE;
```

```
/* Informative signal for validation purpose only */
/* Internal synchronisation signals */
SIGNAL
  BS_READY, MS_READY, REQUEST_MS, REQUEST_BS;
```

```
USE AA_Common_Types;
```

Package BA_Layer2_Service_Primitives

3(7)



```
/* Primitives at the TLB/TMB-SAP,
 clause 19.2.3.3 */

/* Clause 19.2.3.3.1 */
NEWTYPE TLB_Broadcast1RequestType STRUCT
    Chan   ChanType;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
    PDU_Priority PDU_PriorityType;
ENDNEWTYPE;

/* Clause 19.2.3.3.1 */
NEWTYPE TLB_Broadcast1IndicationType STRUCT
    Chan   ChanType;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
ENDNEWTYPE;

/* Proposed by PT93 */
NEWTYPE TLB_Broadcast1ConfirmType STRUCT
    Report TransferReportType;
ENDNEWTYPE;

/* Clause 19.2.3.3.1 */
NEWTYPE TLB_Broadcast2RequestType STRUCT
    Chan   ChanType;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
    PDU_Priority PDU_PriorityType;
ENDNEWTYPE;

/* Clause 19.2.3.3.1 */
NEWTYPE TLB_Broadcast2IndicationType STRUCT
    Chan   ChanType;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
ENDNEWTYPE;

/* Proposed by PT93 */
NEWTYPE TLB_Broadcast2ConfirmType STRUCT
    Report TransferReportType;
ENDNEWTYPE;
```

```
USE AA_Common_Types;
```

Package BA_Layer2_Service_Primitives

4(7)



```
/* Primitives at the TLC/TMC-SAP, clause 19.2.3.4 */

/* Clause xxx, proposed by PT 93 */
NEWTYPE TLC_ConfigureRequestType STRUCT
    ThresholdValues_present Boolean;
    ThresholdValues ThresholdValuesType;
    EnergySavingMode_present Boolean;
    EnergySavingMode EnergySavingModeType;
    MLE_Activity_present Boolean;
    MLE_Activity MLE_ActivityType;
    ValidAddresses_present Boolean;
    ValidAddresses ValidAddressesType;
ENDNEWTYPE;

/* Clause xxx, proposed by PT 93 */
NEWTYPE TLC_ConfigureConfirmType STRUCT
    ThresholdValues_present Boolean;
    ThresholdValues ThresholdValuesType;
    EnergySavingMode_present Boolean;
    EnergySavingMode EnergySavingModeType;
    ValidAddresses_present Boolean;
    ValidAddresses ValidAddressesType;
ENDNEWTYPE;

/* Clause 19.2.3.4.2 */
NEWTYPE TLC_ServingIndicationType STRUCT
    Chan ChanType;
    PathlossC1 PathLossC1Type;
    QualityIndication_present Boolean;
    QualityIndication QualityIndicationType;
ENDNEWTYPE;

/* Clause 19.2.3.4.3 */
NEWTYPE TLC_MonitorIndicationType STRUCT
    Chan ChanType;
    PathlossC2 PathLossC2Type;
    QualityIndication_present Boolean;
    QualityIndication QualityIndicationType;
ENDNEWTYPE;
```

```
USE AA_Common_Types;
```

Package BA_Layer2_Service_Primitives

5 (7)



```
/* Clause 19.2.3.4.3 */
NEWTYPE TLC_MonitorRequestType STRUCT
    Chan ChanType;
    ChannelArray_present Boolean;
    ChannelArray ChannelArrayType;
ENDNEWTYPE;

/* Clause xxx, proposed by PT 93 */
NEWTYPE TLC_ReportIndicationType STRUCT
    Handle_present Boolean;
    Handle Endpoint_ID_Type;
    Report TLC_ReportType;
ENDNEWTYPE;

/* Clause 19.2.3.4.1 */
NEWTYPE TLC_ScanRequestType STRUCT
    Chan ChanType;
    ScanningMeasurementMethod
        ScanningMeasurementMethodType;
    ThresholdLevel_present Boolean;
    ThresholdLevel ThresholdLevelType;
ENDNEWTYPE;

/* Clause 19.2.3.4.1 */
NEWTYPE TLC_ScanConfirmType STRUCT
    Chan ChanType;
    ScanningMeasurementMethod
        ScanningMeasurementMethodType;
    /* replaced by Signal quality ? */
    PathlossC1 PathLossC1Type;
    Report ReportType; /* ? */
ENDNEWTYPE;

/* Clause 20.3.5.4.7 */
NEWTYPE TLC_ScanReportIndicationType STRUCT
    Chan ChanType;
    PathlossC1 PathLossC1Type;
    Report_present Boolean;
    Report ReportType; /* ? */
ENDNEWTYPE;

/* Clause 20.3.5.4.8 */
NEWTYPE TLC_SelectRequestType STRUCT
    Chan ChanType;
    ThresholdLevel_present Boolean;
    ThresholdLevel ThresholdLevelType;
    MainCarrierNo_present Boolean;
    MainCarrierNo MainCarrierNoType;
ENDNEWTYPE;
```

```
/* Clause 20.3.5.4.8 */
NEWTYPE TLC_SelectConfirmType STRUCT
    Chan ChanType;
    ThresholdLevel ThresholdLevelType;
    MainCarrierNo_present Boolean;
    MainCarrierNo MainCarrierNoType;
    Report_present Boolean;
    Report ReportType;
ENDNEWTYPE;

NEWTYPE TLC_AddListRequestType STRUCT
    AddrList ValidAddressesType;
ENDNEWTYPE;
```

```
USE AA_Common_Types;
```

Package BA_Layer2_Service_Primitives

6(7)

```
/* Service primitive definitions for MLE-LLC, clause 20.3.4 */
SIGNAL
  TLA_DATA_request(TLA_DataRequestType),
  TLA_DATA_indication(TLA_DataIndicationType),
  TLA_DATA_confirm(TLA_DataConfirmType),
  TLA_UNITDATA_request(TLA_UnitdataRequestType),
  TLA_UNITDATA_confirm(TLA_UnitdataConfirmType),
  TLA_UNITDATA_indication(TLA_UnitdataIndicationType),
  TLA_CANCEL_request(TLA_CancelRequestType),

  TLB_BROADCAST1_request(TLB_Broadcast1RequestType),
  TLB_BROADCAST2_request(TLB_Broadcast2RequestType),
  TLB_BROADCAST1_indication(TLB_Broadcast1IndicationType),
  TLB_BROADCAST2_indication(TLB_Broadcast2IndicationType),
  TLB_BROADCAST1_confirm(TLB_Broadcast1ConfirmType),
  TLB_BROADCAST2_confirm(TLB_Broadcast2ConfirmType),

  TLC_CONFIGURE_request(TLC_ConfigureRequestType),
  TLC_CONFIGURE_confirm(TLC_ConfigureConfirmType),
  TLC_SERVING_indication(TLC_ServingIndicationType),
  TLC_MONITOR_request(TLC_MonitorRequestType),
  TLC_MONITOR_indication(TLC_MonitorIndicationType),
  TLC_REPORT_indication(TLC_ReportIndicationType),
  TLC_SCAN_request(TLC_ScanRequestType),
  TLC_SCAN_confirm(TLC_ScanConfirmType),
  TLC_SCAN_REPORT_indication(TLC_ScanReportIndicationType),
  TLC_SELECT_request(TLC_SelectRequestType),
  TLC_SELECT_confirm(TLC_SelectConfirmType);

SIGNALLIST TLA_DataRequests =
  TLA_DATA_request,
  TLA_CANCEL_request;

SIGNALLIST TLAU_DataRequests =
  TLA_UNITDATA_request,
  TLA_CANCEL_request;
```

```
USE AA_Common_Types;
```

Package BA_Layer2_Service_Primitives

7(7)



```
SIGNALLIST TLA_DataIndications =
    TLA_DATA_indication,
    TLA_DATA_confirm;

SIGNALLIST TLAU_DataIndications =
    TLA_UNITDATA_indication,
    TLA_UNITDATA_confirm;

SIGNALLIST TLA_DataRequests =
    (TLAA_DataRequests),
    (TLAU_DataRequests);

SIGNALLIST TLA_DataIndications =
    (TLAA_DataIndications),
    (TLAU_DataIndications);

SIGNALLIST TLA_Requests =
    (TLA_DataRequests),
    TLA_Cancel_request;

SIGNALLIST TLA_Indications =
    (TLA_DataIndications);
```

```
SIGNALLIST TLB_Requests =
    TLB_BROADCAST1_request,
    TLB_BROADCAST2_request;

SIGNALLIST TLB_Indications =
    TLB_BROADCAST1_indication,
    TLB_BROADCAST2_indication;

SIGNALLIST TLB_Confirms =
    TLB_BROADCAST1_confirm,
    TLB_BROADCAST2_confirm;
/*
SIGNALLIST TLC_Requests =
    TLC_CONFIGURE_request,
    TLC_MONITOR_request,
    TLC_SCAN_request,
    TLC_SELECT_request;
*/
SIGNALLIST TLC_Indications =
    TLC_CONFIGURE_confirm,
    TLC_SERVING_indication,
    TLC_MONITOR_indication,
    TLC_REPORT_indication,
    TLC_SCAN_confirm,
    TLC_SELECT_confirm;
```

```
USE AA_Common_Types;
USE BA_Layer2_Service_Primitives;
```

Package CA_Layer2_Private

1(6)



```
NEWTYPE BLOCK_ARRAY
    Array(Integer,Boolean)
ENDNEWTYPE BLOCK_ARRAY;
```

```
NEWTYPE BLOCK_NUMBER_ARRAY
    Array(Integer,Integer)
ENDNEWTYPE BLOCK_NUMBER_ARRAY;
```

```
NEWTYPE BAD_PRESIDING_BLOCK_MGMT STRUCT
    Enable Boolean;
    Pattern_BS BLOCK_ARRAY;
    Pattern_MS BLOCK_ARRAY;
    PatternSize Integer;
    Index Integer;
ENDNEWTYPE;
```

```
NEWTYPE BAD_DATA_BLOCK_MGMT STRUCT
    Enable Boolean;
    MaxVal Integer;
    Blk BLOCK_NUMBER_ARRAY;
ENDNEWTYPE;
```

```
/* SDU Array Management */
NEWTYPE SDU_ARRAY_MGMT STRUCT
    NumPending Integer;
    QueueSize Integer;
    TrailingIndex Integer;
    LeadingIndex Integer;
ENDNEWTYPE;
```

```
/* SLEEP PRIMITIVES */
NEWTYPE TLC_SleepRequestType STRUCT
    /* Supported fields */
    EnableSleep Boolean;
ENDNEWTYPE;
```

```
/* TEST SETUP PRIMITIVES */
NEWTYPE TLC_TestSetupRequestType STRUCT
    /* Supported fields */
    PresBlkMgmt BAD_PRESIDING_BLOCK_MGMT;
    DataBlkMgmt BAD_DATA_BLOCK_MGMT;
ENDNEWTYPE;
```

```
USE AA_Common_Types;
USE BA_Layer2_Service_Primitives;
```

Package CA_Layer2_Private

2(6)



```
NEWTYPE PDUTypeType
LITERALS
  PDU_TYPE_SYSINFO1,
  PDU_TYPE_SYSINFO2,
  PDU_TYPE_DOWNLINK_DATA1,
  PDU_TYPE_DOWNLINK_DATA2,
  PDU_TYPE_WAKEUP,
  PDU_TYPE_ACCESS_ANNOUNCE,
  PDU_TYPE_DOWNLINK_RESP1,
  PDU_TYPE_DOWNLINK_RESP2,
  PDU_TYPE_UPLINK_DATA1,
  PDU_TYPE_UPLINK_DATA2,
  PDU_TYPE_UPLINK_RESP;
ENDNEWTYPE;
```

```
NEWTYPE L2ProtocolType
LITERALS
  L2_PROTOCOL_TYPE_ACKNOWLEDGED,
  L2_PROTOCOL_TYPE_UNACKNOWLEDGED;
ENDNEWTYPE;
```

```
SYNTYPE BurstLengthType = /* 6 bits */
  0TO63
ENDSYNTYPE;

SYNTYPE SDUPriorityType = /* 3 bits */
  0TO7
ENDSYNTYPE;

SYNTYPE RelativeBlockNumberType = /* 8 bits */
  0TO255
ENDSYNTYPE;

SYNTYPE CarrierNumberType = /* 10 bits */
  0TO1023
ENDSYNTYPE;

SYNTYPE TrafficLabelType = /* 10 bits */
  0TO1023
ENDSYNTYPE;

SYNTYPE BroadcastLabelType = /* 2 bits */
  0TO3
ENDSYNTYPE;

SYNTYPE PDUNumberType = /* 3 bits */
  0TO7
ENDSYNTYPE;

SYNTYPE SegmentSizeType = /* 6 bits */
  0TO63
ENDSYNTYPE;

SYNTYPE BlocksReservedType = /* 8 bits */
  0TO255
ENDSYNTYPE;
```

```
USE AA_Common_Types;
USE BA_Layer2_Service_Primitives;
```

Package CA_Layer2_Private



```
/* MCCH */
NEWTYPE SYSINFO_1_PDU STRUCT
  PDUType PDUTypeType;
  TL_SDU TL_SDU_Type;
  TL_SDU_Length TL_SDU_LengthType;
ENDNEWTYPE;

NEWTYPE SYSINFO_2_PDU STRUCT
  PDUType PDUTypeType;
  BurstLength BurstLengthType;
  BRLabel BroadcastLabelType;
  TL_SDU TL_SDU_Type;
  TL_SDU_Length TL_SDU_LengthType;
ENDNEWTYPE;
```

```
/* ACCH */
NEWTYPE ACCESS_ANNOUNCE_PDU STRUCT
/* Supported */
  PDUType PDUTypeType;
  WindowMFNStart Integer;
  WindowFNStart Integer;
  WindowSize Integer;
  Priority1 SDUPriorityType;
  Priority2 SDUPriorityType;
  SubWindow1Start RelativeBlockNumber;
  SubWindow1End RelativeBlockNumber;
  SubWindow2Start RelativeBlockNumber;
  SubWindow2End RelativeBlockNumber;
  RACHCarrierNumber CarrierNumberType;
ENDNEWTYPE;

NEWTYPE WAKEUP_PDU STRUCT
/* New */
  NextWU_MultiframeNumber Integer;
  NextWU_FrameNumber Integer;
/* Supported */
  PDUType PDUTypeType;
ENDNEWTYPE;

NEWTYPE DOWNLINK_RESP_1_PDU STRUCT
/* New */
  UD2MultiframeNumber Integer;
  UD2FrameNumber Integer;
/* Supported */
  MainAddr MainAddrType;
  AddrType AddrTypeType;
  PDUType PDUTypeType;
  UTLabel TrafficLabelType;
  NumberBlocksReserved BlocksReserv;
  UplinkCarrierNumber CarrierNumber;
  DownlinkCarrierNumber CarrierNumber;
ENDNEWTYPE;
```

```
/* ACCH */
NEWTYPE DOWNLINK_DATA_1_PDU STRUCT
/* New */
  ProtocolSel L2ProtocolType;
  URMultiframeNumber Integer;
  URFrameNumber Integer;
  NumAccessAttempts AccessAttemptType;
/* Supported */
  PDUType PDUTypeType;
  DTLlabel TrafficLabelType;
  MainAddr MainAddrType;
  AddrType AddrTypeType;
  PDU_Priority PDU_PriorityType;
  UplinkCarrierNumber CarrierNumberType;
  DownlinkCarrierNumber CarrierNumberType;
/* For Testing Only */
  TL_SDU TL_SDU_Type;
  TL_SDU_Length TL_SDU_LengthType;
ENDNEWTYPE;
```

```
/* DTCH */
NEWTYPE DOWNLINK_DATA_2_PDU STRUCT
/* New */
  CloseSDU Boolean;
  NewSegment Boolean;
  URMultiframeNumber Integer;
  URFrameNumber Integer;
/* Supported */
  PDUType PDUTypeType;
  BurstLength BurstLengthType;
  DTLlabel TrafficLabelType;
  PDUNumber PDUNumberType;
  SegmentSize SegmentSizeType;
  BlockFlags BLOCK_ARRAY;
ENDNEWTYPE;

NEWTYPE DOWNLINK_RESP_2_PDU STRUCT
/* New */
  UD2MultiframeNumber Integer;
  UD2FrameNumber Integer;
/* Supported */
  PDUType PDUTypeType;
  UTLabel TrafficLabelType;
  PDUNumber PDUNumberType;
  BlockFlags BLOCK_ARRAY;
  NumberBlocksReserved BlocksReserv;
ENDNEWTYPE;
```

3(6)

```
/* MCCH */
NEWTYPE SYSINFO_1_PDU STRUCT
    PDUType PDUTypeType;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
ENDNEWTYPE;

NEWTYPE SYSINFO_2_PDU STRUCT
    PDUType PDUTypeType;
    BurstLength BurstLengthType;
    BRLLabel BroadcastLabelType;
    TL_SDU TL_SDU_Type;
    TL_SDU_Length TL_SDU_LengthType;
ENDNEWTYPE;
```

```

/* ACCH */
NEWTYPE ACCESS_ANNOUNCE_PDU STRUCT
/* Supported */
PDUType PDUTypeType;
WindowMFNStart Integer;
WindowFNStart Integer;
WindowSize Integer;
Priority1 SDUPriorityType;
Priority2 SDUPriorityType;
SubWindow1Start RelativeBlockNumberType;
SubWindow1End RelativeBlockNumberType;
SubWindow2Start RelativeBlockNumberType;
SubWindow2End RelativeBlockNumberType;
RACHCarrierNumber CarrierNumberType;
ENDNEWTYPE;

NEWTYPE WAKEUP_PDU STRUCT
/* New */
NextWU_MultiframeNumber Integer;
NextWU_FrameNumber Integer;
/* Supported */
PDUType PDUTypeType;
ENDNEWTYPE;

NEWTYPE DOWNLINK_RESP_1_PDU STRUCT
/* New */
UD2MultiframeNumber Integer;
UD2FrameNumber Integer;
/* Supported */
MainAddr MainAddrType;
AddrType AddrTypeType;
PDUType PDUTypeType;
UTLabel TrafficLabelType;
NumberBlocksReserved BlocksReservedType;
UplinkCarrierNumber CarrierNumberType;
DownlinkCarrierNumber CarrierNumberType;
ENDNEWTYPE;

```

```
/* DTCH */
NEWTYPE DOWNLINK_DATA_2_PDU STRUCT
/* New */
    CloseSDU Boolean;
    NewSegment Boolean;
    URMultiframeNumber Integer;
    URFframeNumber Integer;
/* Supported */
    PDUType PDUTypeType;
    BurstLength BurstLengthType;
    DTLlabel TrafficLabelType;
    PDUNumber PDUNumberType;
    SegmentSize SegmentSizeType;
    BlockFlags BLOCK_ARRAY;
ENDNEWTYPE;

NEWTYPE DOWNLINK_RESP_2_PDU STRUCT
/* New */
    UD2MultiframeNumber Integer;
    UD2FrameNumber Integer;
/* Supported */
    PDUType PDUTypeType;
    UTLabel TrafficLabelType;
    PDUNumber PDUNumberType;
    BlockFlags BLOCK_ARRAY;
    NumberBlocksReserved BlocksReservedType;
ENDNEWTYPE;
```

```
USE AA_Common_Types;  
USE BA_Layer2_Service_Primitives;
```

Package CA_Layer2_Private

4(6)



```
/* RACH */  
NEWTYPE UPLINK_DATA_1_PDU STRUCT  
/* New */  
    ProtocolSel L2ProtocolType;  
    CompensateReqForBadBlocks Boolean;  
    NumAccessAttempts AccessAttemptType;  
/* Supported */  
    PDUType PDUtypeType;  
    MainAddr MainAddrType;  
    AddrType AddrTypeType;  
    PDU_Priority PDU_PriorityType;  
    NumberDataBlocksRequested BlocksReservedType;  
/* For Testing Only */  
    TL_SDU TL_SDU_Type;  
    TL_SDU_Length TL_SDU_LengthType;  
ENDNEWTYPE;
```

```
/* UTCI */  
NEWTYPE UPLINK_DATA_2_PDU STRUCT  
/* New */  
    CloseSDU Boolean;  
    NewSegment Boolean;  
    SegmentSize SegmentSizeType;  
/* Supported */  
    PDUType PDUtypeType;  
    BurstLength BurstLengthType;  
    UTLabel TrafficLabelType;  
    PDUNumber PDUNumberType;  
    NumberDataBlocksRequested BlocksReservedType;  
    BlockFlags BLOCK_ARRAY;  
ENDNEWTYPE;
```

```
/* UTCI */  
NEWTYPE UPLINK_RESP_PDU STRUCT  
/* Supported */  
    PDUType PDUtypeType;  
    DTLabel TrafficLabelType;  
    PDUNumber PDUNumberType;  
    BlockFlags BLOCK_ARRAY;  
ENDNEWTYPE;
```

```
USE AA_Common_Types;
USE BA_Layer2_Service_Primitives;
```

Package CA_Layer2_Private

5(6)



```
***** Base-only constants *****
SYNONYM EVENT_LABEL_DT TrafficLabelType = 127;
SYNONYM EVENT_LABEL_UT TrafficLabelType = 33;

SYNONYM CARRIER_NUM_MCCH CarrierNumberType = 0;
SYNONYM CARRIER_NUM_ACCH CarrierNumberType = 1;
SYNONYM CARRIER_NUM_DTCH CarrierNumberType = 2;
SYNONYM CARRIER_NUM_UTCH CarrierNumberType = 3;
SYNONYM CARRIER_NUM_RACH CarrierNumberType = 4;

SYNONYM RACH_WINDOW_SIZE Integer = 20;

***** Mobile-only constants *****
SYNONYM SHORT_SUBSCRIBER_IDENTITY SSI_Type = 34;
/* max number of AA's between MS access attempts */
SYNONYM MAX_RETRY_WINDOW_COUNT Integer = 4;

***** Common constants *****
SYNONYM MULTIFRAMES_PER_SUPERFRAME Integer = 4096;
SYNONYM FRAMES_PER_MULTIFRAME Integer = 8;

SYNONYM MAX_DATA_BLOCKS_PER_BURST Integer = 3; /* spec: 40 */
SYNONYM MAX_DATA_BLOCKS_PER_SEG Integer = 7; /* spec: 40 */

SYNONYM NUM_FRAMES_PER_TEST Duration = 500;
SYNONYM NUM_FRAMES_PER_PDU_TEST Duration = 3000;

SYNONYM SDU_PRIORITY SDUPriorityType = 4;

SYNONYM SDU_QUEUE_SIZE Integer = 3;

SYNONYM MAX_RETRIES Integer = 2; /* N253 */
SYNONYM DR1_INTERVAL Duration = 72; /* T255 */
SYNONYM DL_INTERVAL Duration = 24; /* T256 */
SYNONYM DL_INTERVAL_2 Duration = 48;
SYNONYM DL_INTERVAL_3 Duration = 72;

SYNONYM STACK_TEST Boolean = false;
```

```
USE AA_Common_Types;
USE BA_Layer2_Service_Primitives;
```

Package CA_Layer2_Private

6 (6)

```

-----+
-----+ SIGNAL
-----+ /* System Timing */
-----+ FrameSync(Integer, Integer),
-----+ FrameSyncDL(Integer, Integer),
-----+ FrameSyncUL(Integer, Integer),

-----+ /* Broadcast PDUs - Downlink */
-----+ PBLK_SysInfo1(SYSINFO_1_PDU, CarrierNumberType),
-----+ PBLK_SysInfo2(SYSINFO_2_PDU, CarrierNumberType),
-----+ PBLK_AccessAnnounce(ACCESS_ANNOUNCE_PDU, CarrierNumberType),
-----+ PBLK_DownlinkData1(DOWNLINK_DATA_1_PDU, Boolean, CarrierNumberType),

-----+ /* Broadcast PDUs - Uplink */
-----+ PBLK_UplinkData1(UPLINK_DATA_1_PDU, Boolean, CarrierNumberType),

-----+ /* Base PDUs */
-----+ PBLK_DownlinkData2(DOWNLINK_DATA_2_PDU, Boolean, CarrierNumberType),

-----+ /* Base ACK PDUs */
-----+ PBLK_DownlinkResp1(DOWNLINK_RESP_1_PDU, Boolean, CarrierNumberType),
-----+ PBLK_DownlinkResp2(DOWNLINK_RESP_2_PDU, Boolean, CarrierNumberType),
-----+ PBLK_WakeUp(WAKEUP_PDU, CarrierNumberType),

-----+ /* Mobile PDUs */
-----+ PBLK_UplinkData2(UPLINK_DATA_2_PDU, Boolean, CarrierNumberType),

-----+ /* Mobile ACK PDUs */
-----+ PBLK_UplinkResp(UPLINK_RESP_PDU, Boolean, CarrierNumberType),

-----+ /* Generic Data Block */
-----+ FBLK_DataBlock(Integer, Boolean, CarrierNumberType),

-----+ /* MLE Primitives - Testing Only */
-----+ TLC_SLEEP_Request(TLC_SleepRequestType),
-----+ TLC_TESTSETUP_Request_DL(TLC_TestSetupRequestType),
-----+ TLC_TESTSETUP_Request_UL(TLC_TestSetupRequestType);

```

```
/*#INCLUDE 'random.pr' */
/*#INCLUDE 'file.pr' */
```

```
SIGNALLIST FrameSyncs =
FrameSync,
FrameSyncUL,
FrameSyncDL;
```

```

SIGNALLIST BroadcastPDUs =
PBLK_SysInfo1,
PBLK_SysInfo2,
PBLK_AccessAnnounce,
PBLK_WakeUp;

SIGNALLIST BasePDUs =
PBLK_DownlinkData1,
PBLK_DownlinkData2,
FBLK_DataBlock;

SIGNALLIST BaseAckPDUs =
PBLK_DownlinkResp1,
PBLK_DownlinkResp2;

SIGNALLIST MobilePDUs =
PBLK_UplinkData1,
PBLK_UplinkData2,
FBLK_DataBlock;

SIGNALLIST MobileAckPDUs =
PBLK_UplinkResp;

```

```
SIGNALLIST TLC_Requests =
TLC_SLEEP_Request,
TLC_TESTSETUP_Request_DL,
TLC_TESTSETUP_Request_UL,
TLC_CONFIGURE_request,
TLC_MONITOR_request,
TLC_SCAN_request,
TLC_SELECT_request;
```

```
SIGNALLIST TLC_Requests_DL =
TLC_TESTSETUP_Request_DL,
TLC_SLEEP_Request,
TLC_CONFIGURE_request,
TLC_MONITOR_request,
TLC_SCAN_request,
TLC_SELECT_request;
```

```

SIGNALLIST FrameSyncsMS =
FrameSyncUL,
FrameSyncDL;

SIGNALLIST TLA_Requests2 =
TLA_DATA_Request,
TLA_UNITDATA_Request,
TLA_CANCEL_Request;

SIGNALLIST TLA_Responses =
TLA_DATA_Indication,
TLA_UNITDATA_Indication,
TLA_DATA_Confirm,
TLA_UNITDATA_Confirm;

```

```
SIGNALLIST TLC_Requests_BS =
TLC_SLEEP_Request,
TLC_CONFIGURE_Request,
TLC_MONITOR_Request,
TLC_SCAN_Request,
TLC_SELECT_Request;
```

Use AA_Common_Types;

Package DA_MLE_Service_Primitives

1(7)



```
*****  
Parameter values for MLE protocol  
*****/  
  
***** Maximal index for MLE SDUs *****/  
SYNONYM MaxSDUIndex Natural = 100; /* Use 5 for MLE-L2; Use 100 for Layer 3 */  
  
SYNONYM PREDEFINED_QOS_1 QoS_Type =  
    (. (. 3, NetworkDependent, 256_0, 3 .), (. 3, 3 .) .);  
SYNONYM PREDEFINED_QOS_2 QoS_Type =  
    (. (. 2, NetworkDependent, 128_0, 2 .), (. 2, 2 .) .);
```

```
*****  
/* MLE Service primitives */  
*****/  
  
*****  
/* Definition of subtypes (fields) of the MLE SP */  
*****/  
  
SYNTYPE NoOfAddresses = Natural  
    CONSTANTS 0:7 /* Changed from Natural for Validation */  
ENDSYNTYPE;  
  
NEWTYPE TSI_array  
    ARRAY( NoOfAddresses, TSI_Type );  
ENDNEWTYPE TSI_array;  
  
NEWTYPE TSI_ListType STRUCT  
    Length Natural;  
    Element TSI_array;  
ENDNEWTYPE TSI_ListType;  
  
SYNTYPE SDUIndex = Natural  
    CONSTANTS 1: MaxSDUIndex  
ENDSYNTYPE;  
  
NEWTYPE SDU_ArrayType  
    ARRAY( SDUIndex, Natural );  
ENDNEWTYPE SDU_ArrayType;
```

Use AA_Common_Types;

Package DA_MLE_Service_Primitives

2(7)



```
NEWTYPE MLE_SDU_Type STRUCT
  SDU SDU_ArrayType;
  SDULength Natural;

  /* Code added for write the mle_sdu_type in plain text */
  /*#ADT(W(B))
  #BODY
  #ifdef XREADANDWRITEF
  extern char * yWri_#(mle_sdu_type)( void * Value)

  {
    static char xCharTmp[30];

    sprintf(xCharTmp, "(. MLE-SDU %d: %d %d %d %d .) ",
           (*(#(mle_sdu_type) *)Value).sdulength ,
           (*(#(mle_sdu_type) *)Value).sdu.A[0],
           (*(#(mle_sdu_type) *)Value).sdu.A[1],
           (*(#(mle_sdu_type) *)Value).sdu.A[2],
           (*(#(mle_sdu_type) *)Value).sdu.A[3],
           (*(#(mle_sdu_type) *)Value).sdu.A[4]);
    return xCharTmp;
  }
  #endif
  */
ENDNEWTYPE MLE_SDU_Type;

NEWTYPE Layer2ServiceType
  LITERALS
    ACKNOWLEDGED_REQS,
    UNACKNOWLEDGED
ENDNEWTYPE;

NEWTYPE TransferResultType
  LITERALS
    ENDPOINTID,
    SDU_CANCELED,
    SUCCESS,
    FAIL
ENDNEWTYPE;

NEWTYPE ReceivedAddressType
  LITERALS Individual, Group;
ENDNEWTYPE ReceivedAddressType;

NEWTYPE MLE_QoSType STRUCT
  QualOfServ QoS_Type;
  Prio PDU_PriorityType;
ENDNEWTYPE MLE_QoSType ;
```

Use AA_Common_Types;

Package DA_MLE_Service_Primitives

3(7)

```
/*********************  
/* Definition of MLE SP information elements */  
/*********************  
  
NEWTYPE MLE_ActivateReqType STRUCT  
    MobileCountryCode MCC_Type;  
    MobileNetworkCode MNC_Type;  
ENDNEWTYPE;  
  
NEWTYPE MLE_ActivateConType STRUCT  
    RegistrationRequired Boolean;  
    LocationArea LocationAreaType;  
ENDNEWTYPE;  
  
NEWTYPE MLE_CancelType STRUCT  
    EndpointID Endpoint_ID_Type;  
ENDNEWTYPE;  
  
NEWTYPE MLE_ConfigureReqType STRUCT  
    ChannelChangeAcc ChannelChangeAcceptedType;  
    CallRelease Boolean;  
    EncryptionFlag Boolean;  
    CommunicationType CommunicationTypeType;  
    Simplex_Duplex DuplexFlagType;  
    AddTempGSSI GSSI_Type2;  
    DeleteTempGSSI GSSI_Type2;  
    TxGrant Boolean;  
    Switch_U_Plane Boolean;  
ENDNEWTYPE MLE_ConfigureReqType;  
  
NEWTYPE MLE_MM_IdentitiesType STRUCT  
    ISSI SSI_Type2;  
    ASSI SSI_Type2;  
    Attached_GSSIlist TSI_ListType;  
    Detached_GSSIlist TSI_ListType;  
ENDNEWTYPE MLE_MM_IdentitiesType;  
  
NEWTYPE MLE_InfoReqType STRUCT  
    SubscriberClass SubscriberClassType;  
    EnergySavingMode EnergySavingModeType;  
ENDNEWTYPE MLE_InfoReqType;  
  
NEWTYPE MLE_LinkIndType STRUCT  
    MNC MNC_Type;  
    MCC MCC_Type;  
    LocationArea LocationAreaType;  
ENDNEWTYPE;
```

Use AA_Common_Types;

Package DA_MLE_Service_Primitives

4(7)



```
NEWTYPE MLE_ReportIndType STRUCT
    EndpointID      Endpoint_ID_Type;
    TransferResult  TransferResultType;
ENDNEWTYPE;

NEWTYPE MLE_MM_UnitdataReqType STRUCT
    /* For MM data requests */
    SDU MLE_SDU_Type;
    Layer2Service  Layer2ServiceType;
    PDU_Priority   PDU_PriorityType;
ENDNEWTYPE MLE_MM_UnitdataReqType;

NEWTYPE MLE_OpenReqType STRUCT
    Cell_Id CellIdentifierType;
ENDNEWTYPE MLE_OpenReqType;
```

```
NEWTYPE MLE_SC_UnitdataReqType STRUCT
    /* For SCLNP and CONP data reqs */
    SDU MLE_SDU_Type;
    Layer2Service  Layer2ServiceType;
    PDU_Priority   PDU_PriorityType;
ENDNEWTYPE;

NEWTYPE MLE_MSC_UnitdataIndType STRUCT
    SDU MLE_SDU_Type;
ENDNEWTYPE;

NEWTYPE MLE_UpdateReqType STRUCT
    MCC MCC_Type;
    MNC MNC_Type;
    RegistrationSuccess Boolean;
ENDNEWTYPE;
```

Use AA_Common_Types;

Package DA_MLE_Service_Primitives

5(7)



```
/**************************************************************************/  
/*          Signal declarations  
/**************************************************************************/  
  
SIGNAL  
  MLE_ACTIVATE_request(MLE_ActivateReqType),  
  MLE_ACTIVATE_confirm(MLE_ActivateConType),  
  MLE_ACTIVATE_indication,  
  MLE_CANCEL_request_MM(MLE_CancelType),  
  MLE_CONFIGURE_request( MLE_ConfigureReqType ),  
  MLE_BREAK_indication,  
  MLE_CLOSE_request,  
  MLE_CLOSE_indication,  
  MLE_DEACTIVATE_request,  
  MLE_IDENTITIES_request_MM( MLE_MM_IdentitiesType ),  
  MLE_INFO_request( MLE_InfoReqType ),  
  MLE_LINK_indication(MLE_LinkIndType),  
  MLE_OPEN_request,  
  MLE_OPEN_indication,  
  MLE_REOPEN_indication,  
  MLE_REPORT_indication( MLE_ReportIndType ),  
  MLE_RESET_request,  
  MLE_RESET_indication,  
  MLE_RESET_response,  
  MLE_RESET_confirm,  
  MLE_RESUME_indication,  
  MLE_UNITDATA_request_MM( MLE_MM_UnitdataReqType ),  
  MLE_UNITDATA_request_SCLNP( MLE_SC_UnitdataReqType ),  
  MLE_UNITDATA_request_CONP( MLE_SC_UnitdataReqType ),  
  MLE_UNITDATA_indication_MSC( MLE_MSC_UnitdataIndType ),  
  MLE_UPDATE_request( MLE_UpdateReqType );
```

Use AA_Common_Types;

Package DA_MLE_Service_Primitives

6(7)

```
/*
 *      Signal list declarations
 */
SIGNALLIST MLE_m_MM_indications =
    MLE_ACTIVATE_confirm,
    MLE_ACTIVATE_indication,
    MLE_LINK_indication,
    MLE_REPORT_indication,
    MLE_UNITDATA_indication_MSC;

SIGNALLIST MLE_m_MM_requests =
    MLE_ACTIVATE_request,
    MLE_CANCEL_request_MM,
    MLE_CLOSE_request,
    MLE_DEACTIVATE_request,
    MLE_IDENTITIES_request_MM,
    MLE_INFO_request,
    MLE_OPEN_request,
    MLE_UNITDATA_request_MM,
    MLE_UPDATE_request;

SIGNALLIST MLE_m_CONP_indications =
    MLE_BREAK_indication,
    MLE_CLOSE_indication,
    MLE_OPEN_indication,
    MLE_REPORT_indication,
    MLE_RESET_indication,
    MLE_RESET_confirm,
    MLE_RESUME_indication,
    MLE_UNITDATA_indication_MSC;

SIGNALLIST MLE_m_CONP_requests =
    MLE_RESET_request,
    MLE_RESET_response,
    MLE_UNITDATA_request_CONP;
```

Use AA_Common_Types;

Package DA_MLE_Service_Primitives

7(7)



```
SIGNALLIST MLE_m_SCLNS_indications =
    MLE_CLOSE_indication,
    MLE_OPEN_indication,
    MLE_REPORT_indication,
    MLE_UNITDATA_indication_MSC;

SIGNALLIST MLE_m_SCLNS_requests =
    MLE_UNITDATA_request_SCLNP;

/* Signal lists for the Base Station
   interface */

SIGNALLIST MLE_i_MM_indications =
    MLE_ACTIVATE_confirm,
    MLE_LINK_indication,
    MLE_REPORT_indication,
    MLE_UNITDATA_indication_MSC;

SIGNALLIST MLE_i_MM_requests =
    MLE_ACTIVATE_request,
    MLE_CANCEL_request_MM,
    MLE_CLOSE_request,
    MLE_DEACTIVATE_request,
    MLE_IDENTITIES_request_MM,
    MLE_INFO_request,
    MLE_OPEN_request,
    MLE_UNITDATA_request_MM,
    MLE_UPDATE_request;
```

```
SIGNALLIST MLE_i_CONP_indications =
    MLE_BREAK_indication,
    MLE_CLOSE_indication,
    MLE_OPEN_indication,
    MLE_REPORT_indication,
    MLE_RESET_indication,
    MLE_RESET_confirm,
    MLE_RESUME_indication,
    MLE_UNITDATA_indication_MSC;

SIGNALLIST MLE_i_CONP_requests =
    MLE_RESET_request,
    MLE_RESET_response,
    MLE_UNITDATA_request_CONP;

SIGNALLIST MLE_i_SCLNS_indications =
    MLE_CLOSE_indication,
    MLE_OPEN_indication,
    MLE_REPORT_indication,
    MLE_UNITDATA_indication_MSC;

SIGNALLIST MLE_i_SCLNS_requests =
    MLE_UNITDATA_request_SCLNP;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

1(13)



```
/* Signal NoHandle indicates to the MLE protocol that no LLC handle was
   received resulting from the sending of a data transfer. Then no further
   information on the transfer is preserved as there should not be any
   confirm to such a transmission.
*/
SIGNAL
  NoHandle;
```

```
/*********************************************
/* MLE Protocol Data Units and associated data types */
/********************************************

/*********************************************
/* Definitions of the PDU information elements */
/********************************************

/* Clause 18.5.1 */
SYNTYPE MLE_AnnouncedCellReselSupportedType =
  0TO1
ENDSYNTYPE;

SYNONYM CR_TYPE_3_SUPPORTED MLE_AnnouncedCellReselSupportedType = 0;
SYNONYM CR_TYPE_2_3_SUPPORTED MLE_AnnouncedCellReselSupportedType = 1;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

2(13)



```
/* Clause 18.5.2 */
NEWTYPE BS_ServiceDetailsType STRUCT
    Registration 0TO1;
    DeRegistration 0TO1;
    PriorityCell 0TO1;
    MinimumModeService 0TO1;
    Migration 0TO1;
    Roaming 0TO1;
    CONP_Service 0TO1;
    SCLNP_Service 0TO1;
ENDNEWTYPE ;

NEWTYPE BS_ServiceDetailsType2 STRUCT
    Present Boolean;
    BS_ServiceDetails BS_ServiceDetailsType
ENDNEWTYPE BS_ServiceDetailsType2;

SYNONYM NULL_CELLID CellIdentifierType = 0;

NEWTYPE CellIdentifierType2 STRUCT
    Present Boolean;
    CellIdentifier CellIdentifierType;
ENDNEWTYPE CellIdentifierType2;

/* Clause 18.5.4 */
SYNTYPE SlowReselectThresholdType =
    0TO15
ENDSYNTYPE;

SYNTYPE FastReselectThresholdType =
    0TO15
ENDSYNTYPE;

SYNTYPE SlowReselectHysteresisType =
    0TO15
ENDSYNTYPE;

SYNTYPE FastReselectHysteresisType =
    0TO15
ENDSYNTYPE;

NEWTYPE MLE_CellReselectParmsType STRUCT
    SlowReselectThreshold SlowReselectThresholdType;
    FastReselectThreshold FastReselectThresholdType;
    SlowReselectHysteresis SlowReselectHysteresisType;
    FastReselectHysteresis FastReselectHysteresisType;
ENDNEWTYPE MLE_CellReselectParmsType;

/* Clause 18.5.5 */
SYNTYPE MLE_CellServiceLevelType =
    0TO3
ENDSYNTYPE;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

3(13)



```
/* Clause 18.5.6 */
SYNTYPE MLE_ChanCommValidType =
    0TO3
ENDSYNTYPE;

SYNONYM FOLLOW_MAC_CHANNEL_CHANGE MLE_ChanCommValidType = 0;
SYNONYM CHANGE_CHANNEL_IMMEDIATELY MLE_ChanCommValidType = 1;
SYNONYM NO_CHANNEL_CHANGE MLE_ChanCommValidType = 2;

/* Clause 18.5.7 */
SYNTYPE MLE_FailCauseType =
    0TO3
ENDSYNTYPE;

SYNONYM NEIGHBOUR_CELL_ENQ_NOT_AVAILABLE MLE_FailCauseType = 0;
SYNONYM CELL_RESELECTION_NOT_SUPPORTED MLE_FailCauseType = 1;
SYNONYM SUBSCRIBER_CLASS_NOT_ALLOWED MLE_FailCauseType = 2;
SYNONYM RESTORATION_CAN_NOT_BE_DONE_ON_CELL MLE_FailCauseType = 3;

/* Clause 18.5.9 */
/* LocationAreaType defined in Package
   Commontype
*/
NEWTYPE LocationAreaType2 STRUCT
    Present Boolean;
    LocationArea LocationAreaType;
ENDNEWTYPE LocationAreaType2;

/* Clause 18.5.12 */
SYNTYPE MaxMSTransmitPowerType =
    0TO7
ENDSYNTYPE;

NEWTYPE MaxMSTransmitPowerType2 STRUCT
    Present Boolean;
    MaxMSTransmitPower MaxMSTransmitPowerType;
ENDNEWTYPE MaxMSTransmitPowerType2;

/* Clause 18.5.13 */
SYNTYPE MinRXAccessLevelType =
    0TO15
ENDSYNTYPE;

NEWTYPE MinRXAccessLevelType2 STRUCT
    Present Boolean;
    MinRXAccessLevel MinRXAccessLevelType;
ENDNEWTYPE MinRXAccessLevelType2;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

4(13)



```
/* Clause 18.5.14 */
/* Type mobile countrycode defined
   as MCC_Type in package AAcommontype
*/
NEWTYPE MCC_Type2 STRUCT
  Present Boolean;
  MCC MCC_Type;
ENDNEWTYPE MCC_Type2;

/* Clause 18.5.15 */
/* Type Mobile network code defined
   as MNC_Type in Package AAcommontype
*/
NEWTYPE MNC_Type2 STRUCT
  Present Boolean;
  MNC MNC_Type;
ENDNEWTYPE MNC_Type2;

/* Clause 18.5.16 */
NEWTYPE NeighbourCellBroadcastType STRUCT
  DNWRK_Broadcast 0TO1;
  DNWRK_Enquiry 0TO1;
ENDNEWTYPE NeighbourCellBroadcastType;

/* Clause 18.5.17 */
NEWTYPE MLE_NeighbourCellInfoType STRUCT
  CellIdentifier CellIdentifierType;
  AnnounCellReselSupported MLE_AnnouncedCellReselSupportedType;
  NeighbourCellSync NeighbourCellSyncType;
  CellServiceLevel MLE_CellServiceLevelType;
  MainCarrierNo MainCarrierNoType;
  O_bit Boolean; /* Type 2 elements */
  MainCarrierNoExt MainCarrierNoExtType2;
  MobileCountryCode MCC_Type2;
  MobileNetworkCode MNC_Type2;
  LocationArea LocationAreaType2;
  MaxMSTransmitPower MaxMSTransmitPowerType2;
  MinRXAccessLevel MinRXAccessLevelType2;
  SubscriberClass SubscriberClassType2;
  BS_ServiceDetails BS_ServiceDetailsType2;
ENDNEWTYPE MLE_NeighbourCellInfoType;

SYNTYPE MLE_NoofNeighbourCellsType =
  0TO7
ENDSYNTYPE MLE_NoofNeighbourCellsType;

SYNTYPE MLE_NeighbourCellsIndexType = Natural
  Constants 0:1
ENDSYNTYPE MLE_NeighbourCellsIndexType;

NEWTYPE MLE_NeighbourCellInfoListType
  ARRAY(MLE_NeighbourCellsIndexType, MLE_NeighbourCellInfoType)
ENDNEWTYPE MLE_NeighbourCellInfoListType;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

5(13)



```
/* Clause 18.5.18 */
SYNTYPE NeighbourCellSyncType =
  0TO1
ENDSYNTYPE;

/* Clause 18.5.19 */
SYNTYPE NoofNeighbourCellsType =
  0TO7
ENDSYNTYPE;

NEWTYPE NoofNeighbourCellsType2 STRUCT
  Present Boolean;
  NoofNeighbourCells NoofNeighbourCellsType;
ENDNEWTYPE NoofNeighbourCellsType2;

/* Clause 18.5.20 */
NEWTYPE MLE_D_PduType
  LITERALS
    D_NEW_CELL_PDU, /* = 0 */
    D_PREPARE_FAIL_PDU, /* = 1 */
    D_NWRK_BROADCAST_PDU, /* = 2 */
    RESERVED3;
  OPERATORS
    retr_D_PDU_Kind: TL_SDU_ArrayType -> MLE_D_PduType;
/*#ADT (B)
#BODY
#(MLE_D_PduType) #(retr_D_PDU_Kind) (#(TL_SDU_ArrayType) sdu)
{
  return *((#(MLE_D_PduType) *)(&sdu.A[1]));
}
*/
ENDNEWTYPE;

/* Clause 18.5.20 */
NEWTYPE MLE_U_PduType
  LITERALS
    U_PREPARE_PDU, /* = 0 */
    UNUSED_U_PDU1, /* = 1 */
    UNUSED_U_PDU2, /* = 2 */
    UNUSED_U_PDU3; /* = 3 */
  OPERATORS
    retr_U_PDU_Kind: TL_SDU_ArrayType -> MLE_U_PduType;
/*#ADT (B)
#BODY
#(MLE_U_PduType) #(retr_U_PDU_Kind) (#(TL_SDU_ArrayType) sdu)
{
  return *((#(MLE_U_PduType) *)(&sdu.A[1]));
}
*/
ENDNEWTYPE;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

6(13)



```
/* Clause 18.5.21 */
NEWTYPE MLE_ProtocolDiscriminatorType
LITERALS
  RESERVED_DISCR_VALUE, /* = 0 */
  MM_PROTOCOL,           /* = 1 */
  CONP_PROTOCOL,         /* = 3 */
  SCLNP_PROTOCOL,        /* = 4 */
  MLE_PROTOCOL,          /* = 5 */
  TME_PROTOCOL;          /* = 6 */
OPERATORS
  retr_ProtocolDiscr: TL_SDU_Type -> MLE_ProtocolDiscriminatorType;
/*#ADT (B)
#BODY
#(MLE_ProtocolDiscriminatorType) #(retr_ProtocolDiscr) (#(TL_SDU_Type) sdu)
{
  return *((#(MLE_ProtocolDiscriminatorType) *)(&sdu));
}
*/
ENDNEWTYPE;

/* Clause 18.5.23 */
SYNTYPE TETRA_NetworkTimeType =
  0TO16777215 /* 24 bit */
ENDSYNTYPE;

SYNTYPE LocalTimeOffsetType =
  0TO16777215 /* 24 bit */
ENDSYNTYPE;

NEWTYPE TETRA_NetworkTimeElementType STRUCT
  NetworkTime TETRA_NetworkTimeType;
  LocalTimeOffset LocalTimeOffsetType;
ENDNEWTYPE TETRA_NetworkTimeElementType;

NEWTYPE TETRA_NetworkTimeElementType2 STRUCT
  Present Boolean;
  TETRA_NetworkTimeElement TETRA_NetworkTimeElementType;
ENDNEWTYPE TETRA_NetworkTimeElementType2;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

7(13)



```
/*
 * PDUs
 */
/* Clause 18.4.1.3 */
NEWTYPE MLE_DataTransferPDUType STRUCT
    ProtocolDiscriminator MLE_ProtocolDiscriminatorType;
    O_bit Boolean; /* Type 2 elements */
    SDU MLE_SDU_Type;
ADDING
    OPERATORS
        tc_DataTransferPDU: MLE_DataTransferPDUType -> TL_SDU_Type;
        retr_DataTransferPDU: TL_SDU_Type -> MLE_DataTransferPDUType;
/*#ADT (B)
#BODY
#(TL_SDU_Type) #(tc_DataTransferPDU) (#(MLE_DataTransferPDUType) pdu)
{
    return *((#(TL_SDU_Type) *)(&pdu));
}
#(MLE_DataTransferPDUType) #(retr_DataTransferPDU) (#(TL_SDU_Type) sdu)
{
    return *((#(MLE_DataTransferPDUType) *)(&sdu));
}
*/
ENDNEWTYPE MLE_DataTransferPDUType;

/* Clause 18.4.1.4.1 */
NEWTYPE MLE_D_NWRK_BroadcastType STRUCT
    ProtocolDiscriminator MLE_ProtocolDiscriminatorType;
    PduType MLE_D_PduType;
    CellReselectParms MLE_CellReselectParmsType;
    CellServiceLevel MLE_CellServiceLevelType;
    O_bit Boolean; /* Type 2 elements */
    TETRA_NetworkTime TETRA_NetworkTimeElementType2;
    NoofNeighbourCells NoofNeighbourCellsType2;
    NeighbourCellInfoList MLE_NeighbourCellInfoListType;
ADDING
    OPERATORS
        tc_D_NWRK_BroadcastPDU: MLE_D_NWRK_BroadcastType -> TL_SDU_Type;
        retr_D_NWRK_BroadcastPDU: TL_SDU_Type -> MLE_D_NWRK_BroadcastType;
/*#ADT (B)
#BODY
#(TL_SDU_Type) #(tc_D_NWRK_BroadcastPDU) (#(MLE_D_NWRK_BroadcastType) pdu)
{
    return *((#(TL_SDU_Type) *)(&pdu));
}
#(MLE_D_NWRK_BroadcastType) #(retr_D_NWRK_BroadcastPDU) (#(TL_SDU_Type) sdu)
{
    return *((#(MLE_D_NWRK_BroadcastType) *)(&sdu));
}
*/
ENDNEWTYPE MLE_D_NWRK_BroadcastType;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

8(13)



```
/* Clause 18.4.1.4.2 */
NEWTYPE MLE_D_NewCellType STRUCT
ProtocolDiscriminator MLE_ProtocolDiscriminatorType;
PduType MLE_D_PduType;
ChanCommValid MLE_ChanCommValidType;
O_bit Boolean; /* Type 2 elements */
MM_Sdu MLE_Sdu_Type;
ADDING
OPERATORS
    tc_D_NewCellPDU: MLE_D_NewCellType -> TL_SDU_Type;
    retr_D_NewCellPDU: TL_SDU_Type -> MLE_D_NewCellType;
/*#ADT (B)
#BODY
#(TL_SDU_Type) #(tc_D_NewCellPDU) (#(MLE_D_NewCellType) pdu)
{
    return *((#(TL_SDU_Type) *)(&pdu));
}
#(MLE_D_NewCellType) #(retr_D_NewCellPDU) (#(TL_SDU_Type) sdu)
{
    return *((#(MLE_D_NewCellType) *)(&sdu));
}
*/
ENDNEWTYPE MLE_D_NewCellType;

/* Clause 18.4.1.4.3 */
NEWTYPE MLE_D_PrepFailType STRUCT
ProtocolDiscriminator MLE_ProtocolDiscriminatorType;
PduType MLE_D_PduType;
FailCause MLE_FailCauseType;
O_bit Boolean; /* Type 2 elements */
MM_Sdu MLE_Sdu_Type; /* NOTE: This field is only introduced in the validation model. It is not described in the MLE PDU definition Clause 18.4.1.4.3. */
ADDING
OPERATORS
    tc_D_PrepFailPDU: MLE_D_PrepFailType -> TL_SDU_Type;
    retr_D_PrepFailPDU: TL_SDU_Type -> MLE_D_PrepFailType;
/*#ADT (B)
#BODY
#(TL_SDU_Type) #(tc_D_PrepFailPDU) (#(MLE_D_PrepFailType) pdu)
{
    return *((#(TL_SDU_Type) *)(&pdu));
}
#(MLE_D_PrepFailType) #(retr_D_PrepFailPDU) (#(TL_SDU_Type) sdu)
{
    return *((#(MLE_D_PrepFailType) *)(&sdu));
}
*/
ENDNEWTYPE MLE_D_PrepFailType;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

9(13)



```
/* Clause 18.4.1.4.6 */
NEWTYPE MLE_U_PrepareType STRUCT
ProtocolDiscriminator MLE_ProtocolDiscriminatorType;
PduType MLE_U_PduType;
O_bit Boolean; /* Type 2 elements */
CellIdentifier CellIdentifierType2;
MM_Sdu MLE_SDU_Type;
ADDING
OPERATORS
    tc_U_PreparePDU: MLE_U_PrepareType -> TL_SDU_Type;
    retr_U_PreparePDU: TL_SDU_Type -> MLE_U_PrepareType;
/*#ADT (B)
#BODY
#(TL_SDU_Type) #(tc_U_PreparePDU) (#(MLE_U_PrepareType) pdu)
{
    return *((#(TL_SDU_Type) *)(&pdu));
}
#(MLE_U_PrepareType) #(retr_U_PreparePDU) (#(TL_SDU_Type) sdu)
{
    return *((#(MLE_U_PrepareType) *)(&sdu));
}
*/
ENDNEWTYPE MLE_U_PrepareType;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

10(13)



```
/* Clause 18.4.2.2 */
NEWTYPE MLE_D_SIN1Type STRUCT
  LocationArea LocationAreaType;
  SubscriberClass SubscriberClassType;
  BS_ServiceDetails BS_ServiceDetailsType;
ADDING
  OPERATORS
    tc_D_SIN1PDU: MLE_D_SIN1Type -> TL_SDU_Type;
    retr_D_SIN1PDU: TL_SDU_Type -> MLE_D_SIN1Type;
/*#ADT (B)
#BODY
#(TL_SDU_Type) #(tc_D_SIN1PDU) (#(MLE_D_SIN1Type) pdu)
{
  return *((#(TL_SDU_Type) *)(&pdu));
}
#(MLE_D_SIN1Type) #(retr_D_SIN1PDU) (#(TL_SDU_Type) sdu)
{
  return *((#(MLE_D_SIN1Type) *)(&sdu));
}
*/
ENDNEWTYPE MLE_D_SIN1Type;

/* Clause 18.4.2.1 */
NEWTYPE MLE_D_SIN2Type STRUCT
  MobileCountryCode MCC_Type;
  MobileNetworkCode MNC_Type;
  NeighbourCellBroadcast NeighbourCellBroadcastType;
  CellServiceLevel MLE_CellServiceLevelType;
ADDING
  OPERATORS
    tc_D_SIN2PDU: MLE_D_SIN2Type -> TL_SDU_Type;
    retr_D_SIN2PDU: TL_SDU_Type -> MLE_D_SIN2Type;
/*#ADT (B)
#BODY
#(TL_SDU_Type) #(tc_D_SIN2PDU) (#(MLE_D_SIN2Type) pdu)
{
  return *((#(TL_SDU_Type) *)(&pdu));
}
#(MLE_D_SIN2Type) #(retr_D_SIN2PDU) (#(TL_SDU_Type) sdu)
{
  return *((#(MLE_D_SIN2Type) *)(&sdu));
}
*/
ENDNEWTYPE MLE_D_SIN2Type;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

11(13)



```
/*
 * Signal declarations
 */
/* Down link PDUs at the TLA SAP */
SIGNAL
    MLE_D_NWRK_BROADCAST( MLE_D_NWRK_BroadcastType ),
    MLE_D_NEW_CELL( MLE_D_NewCellType ),
    MLE_D_PREPARE_FAIL( MLE_D_PrepareFailType );

/* Up link PDUs at the TLA SAP */
SIGNAL
    MLE_U_PREPARE( MLE_U_PrepareType );

/* Down link Broadcast PDUs at the TLB SAP */
SIGNAL
    MLE_D_SIN1( MLE_D_SIN1Type ),
    MLE_D_SIN2( MLE_D_SIN2Type );

/* Clause 18.4.1.3 */
/* Data Transfer PDUs for MLE service users */
SIGNAL
    MLE_U_DataTransfer_PDU( MLE_DataTransferPDUType ),
    MLE_D_DataTransfer_PDU( MLE_DataTransferPDUType );

/*
 * Signal list declarations
 */
SIGNALLIST TLA_DownLink_PDUs =
    MLE_D_NWRK_BROADCAST,
    MLE_D_NEW_CELL,
    MLE_D_PREPARE_FAIL,
    MLE_D_DataTransfer_PDU;

SIGNALLIST TLA_Uplink_PDUs =
    MLE_U_PREPARE,
    MLE_U_DataTransfer_PDU;

SIGNALLIST TLB_Downlink_PDUs =
    MLE_D_SIN1,
    MLE_D_SIN2;

SIGNALLIST MLE_D_PDUs =
    (TLA_DownLink_PDUs),
    (TLB_Downlink_PDUs);

SIGNALLIST MLE_U_PDUs =
    (TLA_Uplink_PDUs);
```

```
SIGNALLIST U_TLAB_SPs =
    TLA_DATA_confirm,
    TLA_UNITDATA_confirm,
    NoHandle;

SIGNALLIST D_TLAB_SPs =
    MLE_CANCEL_request_MM;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

12(13)



```
/****************************************************************************
 *          MLE Protocol Data Units and associated data types      */
/****************************************************************************
 /* Internal data types used by different entities in the MLE system */
/* Data type for transfer of additional data PDU information */

NEWTYPE DataReqInfoType STRUCT
    AddrType    AddrTypeType;
    MainAddr   MainAddrType;
    EndpointID Endpoint_ID_Type;
    ScCode     ScramblingCodeType;
    SubscrClass SubscriberClassType;
    QoS        MLE_QoSType;
    Layer2Service Layer2ServiceType;
    PDU_Priority PDU_PriorityType;
    FCS_Flag Boolean;
ENDNEWTYPE DataReqInfoType;
```

```
Use AA_Common_Types;
USE BA_Layer2_Service_Primitives;
Use DA_MLE_Service_Primitives;
```

Package DB_MLE_Protocol_Primitives

13(13)



```
/* Set according to the sort of validation procedure to perform */
SYNONYM ValidationOption Natural = 1; /* EXTERNAL; */
```

```
/* External synonyms for Selectable options, which are used if
ValidationOption = 2
*/
SYNONYM Ext_MS_ServiceReqSatisfied Boolean = TRUE; /* EXTERNAL; */
SYNONYM Ext_NB_Service_OK Boolean = TRUE; /* EXTERNAL; */
SYNONYM Ext_FCS_FlagSet Boolean = TRUE; /* EXTERNAL; */
SYNONYM Ext_Background_Scanning_Supported Boolean = TRUE; /* EXTERNAL; */
SYNONYM Ext_NeighbourCellService Boolean = TRUE; /* EXTERNAL */
```

```
***** Parameters used *****
```

```
/* SSI value */
SYNONYM SSI_Value SSI_Type = 4545;
/* USSI value associated to MS (24bit, 0:16777215) */
SYNONYM USSI_Value USSI_Type = SSI_Value; /* Home_ITSI!SSI; */ /* EXTERNAL; */
```

```
***** COMMON CONSTANTS *****
```

```
/* The max size in bits of the PDU sent via a basic link connection.

NOTE: This value can be set different in other MS models as the
standard only specifies that it should be approx. 3 TDMA time
slots worth of data.
*/
SYNONYM MAX_BL_SIZE Natural = 750;
```

```
/* Timer duration value for Wait Handle timer */
SYNONYM WAIT_ENDPOINTID_TMDURATION Duration = 2;
```

```
USE AA_Common_Types;
USE BA_Layer2_Service_Primitives;
USE CA_Layer2_Private;
USE DA_MLE_Service_Primitives;
USE DB_MLE_Protocol_Primitives;
```

Package DC_MLE

1(2)

MLE_MS_Protocol_Type

MLE_MS_Formatter_type

```
USE AA_Common_Types;
USE BA_Layer2_Service_Primitives;
USE CA_Layer2_Private;
USE DA_MLE_Service_Primitives;
USE DB_MLE_Protocol_Primitives;
```

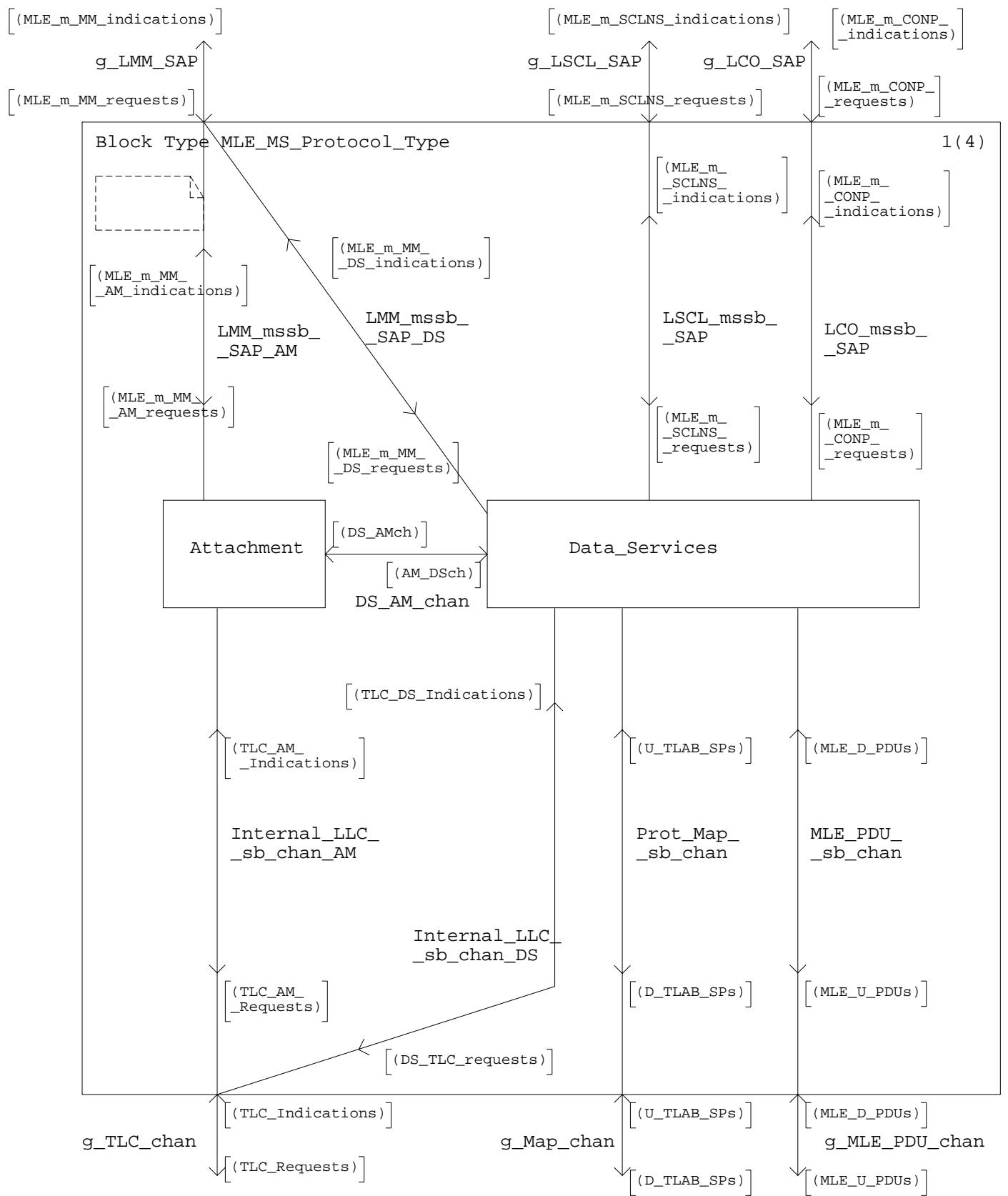
Package DC_MLE

2(2)



```
/* Remote procedure definition for transfer of additional data parameters
   at the MS side used by the PDU to LLC Service primitive formatter. When
   a data PDU is transferred the remote procedure is called to transfer
   the additional parameters.
*/
REMOTE PROCEDURE GetSendDataParms;
FPAR
  IN/OUT DataReqInfoType,
  IN MLE_ProtocolDiscriminatorType;
```

```
/* Remote procedure for getting a EndpointID */
REMOTE PROCEDURE GetEndpointID;
RETURNS Endpoint_ID_TYPE;
```



Block Type MLE_MS_Protocol_Type



```
/* The different types of cell
   reselections */

NEWTYPE CellReselectionType
LITERALS
  NO_CR,
  INITIAL_CR,
  UNDECLARED_CR,
  UNANNOUNCED_CR,
  ANNOUNCED_T3_CR,
  ANNOUNCED_T2_CR;
ENDNEWTYPE;
```

```
/* Type to transfer information
   from the attachment process to
   the data transfer process on
   the MCC, MNC, and Location area
   of a newly selected cell.
   The fields are present only
   if the values differ from
   those the previous selected
   cell.
*/
NEWTYPE NewCellInfoType STRUCT
  NewMCC MCC_Type2;
  NewMNC MNC_Type2;
  NewLocArea LocationAreaType2;
ENDNEWTYPE NewCellInfoType;
```

```
/* Signals exchanged between the
   Attachment block and the
   Data Service block
*/
SIGNAL
  SCANSTATUS( Boolean ),
  SCANSTATUS_ACK,
  NEWC_INFO( NewCellInfoType );
```

2 (4)

```
SIGNALLIST AM_DSCh =
  CR_BREAK_indication,
  CR_RESUME_indication,
  MLE_BREAK_indication,
  MLE_REOPEN_indication,
  MLE_U_PREPARE,
  NEWC_INFO,
  SCANSTATUS;

SIGNALLIST DS_AMch =
  MLE_D_NWRK_BROADCAST,
  MLE_D_NEW_CELL,
  MLE_D_SIN2,
  MLE_D_PREPARE_FAIL,
  MLE_D_SIN1,
  MLE_OPEN_request,
  SCANSTATUS_ACK;

SIGNALLIST MLE_m_MM_AM_indications =
  MLE_ACTIVATE_confirm,
  MLE_ACTIVATE_indication,
  MLE_LINK_indication;

SIGNALLIST MLE_m_MM_AM_requests =
  MLE_ACTIVATE_request,
  MLE_DEACTIVATE_request,
  MLE_UPDATE_request;

SIGNALLIST MLE_m_MM_DS_indications =
  MLE_UNITDATA_indication_MSC,
  MLE_REPORT_indication;

SIGNALLIST MLE_m_MM_DS_requests =
  MLE_CANCEL_request_MM,
  MLE_CLOSE_request,
  MLE_IDENTITIES_request_MM,
  MLE_INFO_request,
  MLE_OPEN_request,
  MLE_UNITDATA_request_MM;

SIGNALLIST DS_TLC_requests =
  TLC_CONFIGURE_request;

SIGNALLIST TLC_DS_Indications =
  TLC_CONFIGURE_confirm;

SIGNALLIST TLC_AM_Indications =
  TLC_SERVING_indication,
  TLC_MONITOR_indication,
  TLC_REPORT_indication,
  TLC_SCAN_confirm,
  TLC_SELECT_confirm;

SIGNALLIST TLC_AM_Requests =
  TLC_MONITOR_request,
  TLC_SCAN_request,
  TLC_SELECT_request;
```

Block Type MLE_MS_Protocol_Type

3(4)

```
/* Remote procedure used to transfer
information from process
Attachment_management to
process Data_transfer
*/
REMOTE PROCEDURE GetScramblingCode;
RETURNS ScramblingCodeType;
```

```
/* Remote variable used by the attachment
process to check if there is any CONP
transfer in progress. The information
is held by the data transfer process.
*/
REMOTE is_CONP_InProgress Boolean;
```

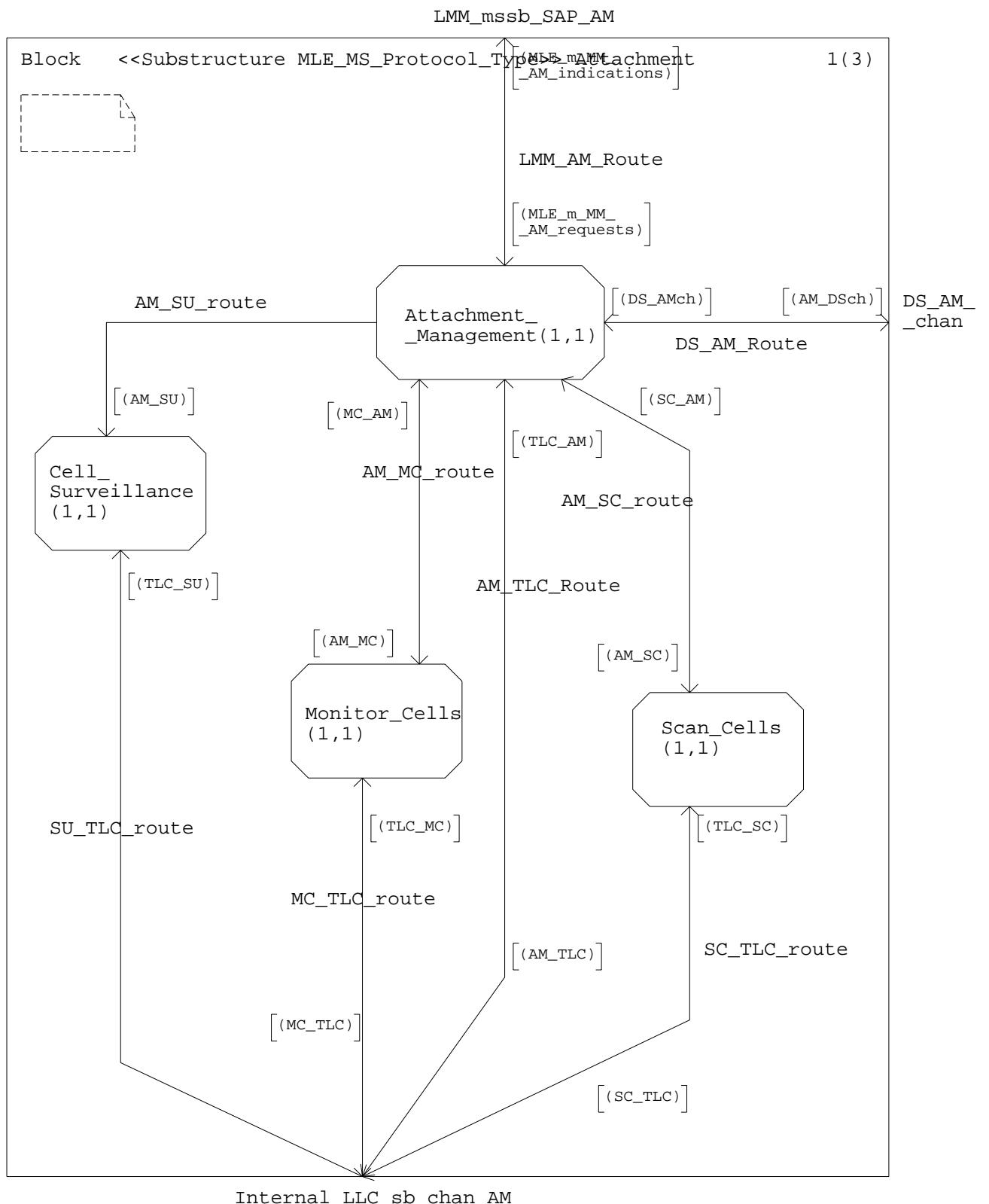
Block Type MLE_MS_Protocol_Type

4(4)



```
/* Signal from Attachment Management to Data Transfer
   to indicate the type of cell reselection and make
   the Data Transfer inform CMCE, CONP and SCLNP by
   Break- and Close-indications.
   Note: this signal replaces MLE-INDICATION as
   described in 18.3.5.2.1 d).
*/
SIGNAL
  CR_BREAK_indication( CellReselectionType );
```

```
/* Signal from the Attachment Management process to
   the Data Transfer process to let it inform the
   the MLE service users CMCE, CONP and SCLNP that
   they may resume connections.
*/
SIGNAL
  CR_RESUME_indication( CellReselectionType );
```



Block <<Substructure MLE_MS_Protocol_Type>> Attachment

2(3)



/* ** DATA TYPE DEFINITIONS **/

```
/* Types for monitoring and
   ranking of monitored cells
*/
NEWTYPE MonCellDescrType STRUCT
  CellIndex Natural;
  Chan ChanType;
  C2 PathLossC2Type;
  MonTime Time;
  Monitored Boolean;
ENDNEWTYPE;

NEWTYPE MonCellDescrArrayType
  ARRAY(CellIndexType, MonCellDescrType);
ENDNEWTYPE;

NEWTYPE MonListType STRUCT
  Length Natural;
  MonDescr MonCellDescrArrayType;
ENDNEWTYPE;

/* Types for scan and
   rank list of scanned
   cells
*/
SYNTYPE CellIndexType =
  Natural CONSTANTS 0:7
ENDSYNTYPE;

NEWTYPE ScanCellDescrType STRUCT
  CellIndex CellIndexType;
  Chan ChanType;
  C1 PathLossC1Type;
  ScanTime Time;
  Exclude Boolean;
ENDNEWTYPE;

NEWTYPE ScanCellDescrArrayType
  ARRAY(CellIndexType, ScanCellDescrType);
ENDNEWTYPE;

NEWTYPE ScanListType STRUCT
  Length Natural;
  ScanDescr ScanCellDescrArrayType;
ENDNEWTYPE;
```

/* ** SIGNAL DEFINITIONS **/

```
/* Signals exchanged between Attachment
   Management process and Surveillance
   process
*/
SIGNAL
  START_SURVEILLANCE,
  STOP_SURVEILLANCE;

/* Signals exchanged between Attachment
   Management process and Monitoring
   process
*/
SIGNAL
  STOP_MONITORING,
  START_MONITORING( ChanType,
                     MonListType ),
  SUSPEND_MON_Ind,
  RESUME_MON_Ind,
  RANKED_MONLIST( MonListType,
                  Boolean );
```

```
/* Signals used between Attachment
   Management process and Scanning
   process
*/
SIGNAL
  RANKED_SCAN( ScanListType,
                Boolean ),
  SCANLIST( ScanningMeasurementMethodType,
            ScanListType ),
  SCAN_ALL( ScanListType ),
  SCAN_NEXT_CELL_req( CellIndexType ),
  SCAN_NEXT_CELL_ACK,
  SCAN_RESUME,
  SGL_SCAN_req( ScanCellDescrType ),
  SGL_SCAN_conf( ScanCellDescrType ),
  STOP_SCAN;
```

Block <<Substructure MLE_MS_Protocol_Type>> Attachment 3(3)

```
/* SIGNAL LISTS DEFINITIONS */  
  
SIGNALLIST TLC_AM =  
    TLC_REPORT_indication,  
    TLC_SELECT_confirm;  
  
SIGNALLIST AM_TLC =  
    TLC_SELECT_request;  
  
SIGNALLIST SC_TLC =  
    TLC_SCAN_request;  
  
SIGNALLIST TLC_SC =  
    TLC_SCAN_confirm;  
  
SIGNALLIST SC_AM =  
    SGL_SCAN_conf,  
    SCAN_NEXT_CELL_req,  
    RANKED_SCAN;  
  
SIGNALLIST AM_SC =  
    SGL_SCAN_req,  
    STOP_SCAN,  
    SCANLIST,  
    SCAN_RESUME,  
    SCAN_NEXT_CELL_ACK,  
    SCAN_ALL;  
  
SIGNALLIST MC_TLC =  
    TLC_MONITOR_request;  
  
SIGNALLIST TLC_MC =  
    TLC_MONITOR_Indication;  
  
SIGNALLIST MC_AM =  
    RANKED_MONLIST;  
  
SIGNALLIST AM_MC =  
    RESUME_MON_Ind,  
    START_MONITORING,  
    STOP_MONITORING,  
    SUSPEND_MON_Ind;  
  
SIGNALLIST TLC_SU =  
    TLC_SERVING_indication;  
  
SIGNALLIST AM_SU =  
    START_SURVEILLANCE,  
    STOP_SURVEILLANCE;  
  
/* Remote procedure to update  
surveillance information on  
the serving cell between the  
surveillance process and the  
attachment management process.  
*/  
REMOTE PROCEDURE SurveilInfo:  
FPAR  
    IN TLC_ServingIndicationType,  
    IN Time;
```

Process CellSurveillance

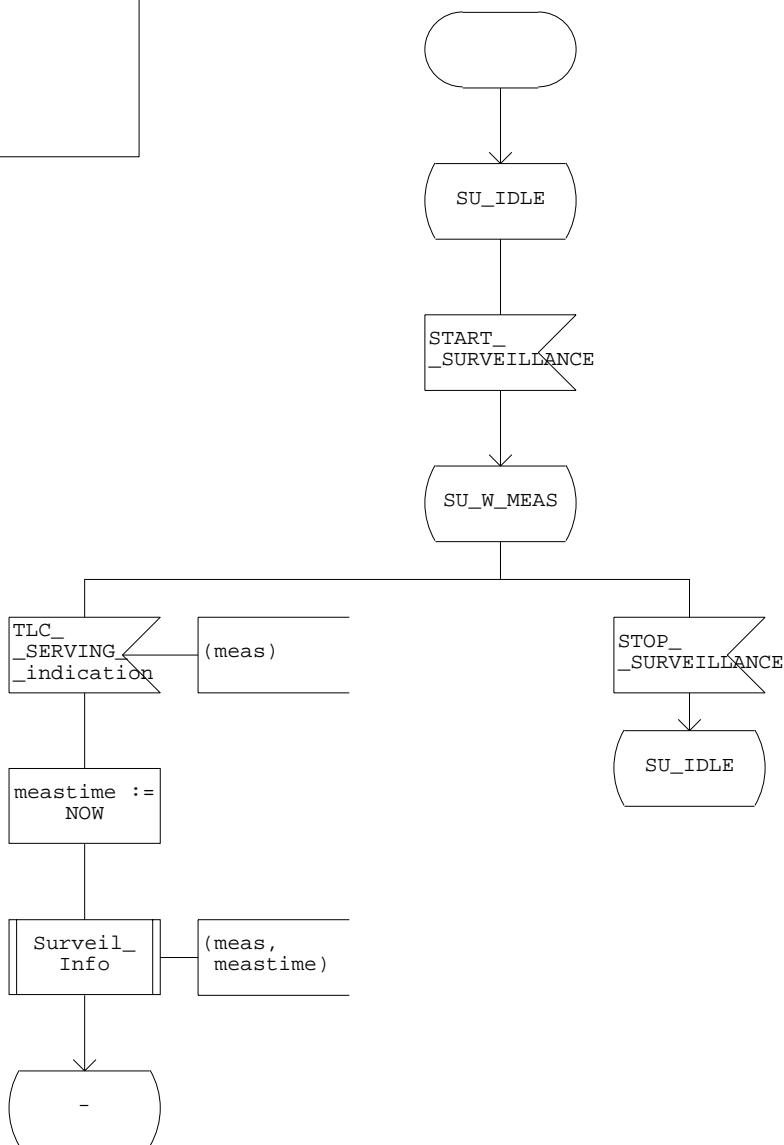
1(1)



```
DCL
  meas  TLC_ServingIndicationType,
  meastime Time;
```

```
/* Remote procedure to update surveillance
   information on the serving cell descriptor
   of the attachment management process.
```

```
IMPORTED PROCEDURE SurveilInfo;
  FPAR
    IN TLC_ServingIndicationType,
    IN Time;
```



```
/** Cell descriptor data type definition **/
```

```
NEWTYPE CellElemType STRUCT
    Suspended          Boolean;
    CellId             CellIdentifierType;
    MCC                MCC_Type2;
    MNC                MNC_Type2;
    NeighbourCellBroadcast NeighbourCellBroadcastType;
    CellReselectParms MLE_CellReselectParmsType;
    LocationArea       LocationAreaType2;
    SubscriClass       SubscriberClassType;
    BSServiceDetails  BS_ServiceDetailsType;
    CellServiceLevel  MLE_CellServiceLevelType;
    AnnounReselect    MLE_AnnouncedCellReselSupportedType;
    NeighbourSync     NeighbourCellSyncType;
    MainCarrierNo     MainCarrierNoType;
    MainCarrierNoExt  MainCarrierNoExtType2;
    MaxMSPow          MaxMSTransmitPowerType2;
    MinRXAccLev      MinRXAccessLevelType2;
    C1                 PathLossC1Type;
    C2                 PathLossC2Type;
    QualityIndication QualityIndicationType;
    C1Below_FRT        Boolean; /* FRT - Fast Reselect Threshold */
    C1BelowFRTSince   Time;
    C1Below_SRT        Boolean;
    C1BelowSRTSince   Time; /* SRT Slow Reselect Threshold */
    ExceedC1pFRH      Boolean; /* Fast Reselect Hysteresis */
    ExceedC1pFRHSince Time;
    ExceedC1pSRH      Boolean; /* Slow Reselect Hysteresis */
    ExceedC1pSRHSince Time;
    ExceedC2pFRH      Boolean; /* Fast Reselect Hysteresis */
    ExceedC2pFRHSince Time;
    ExceedC2pSRH      Boolean; /* Slow Reselect Hysteresis */
    ExceedC2pSRHSince Time;
    C1gtFRTpFRH       Boolean;
    C1gtFRTpFRHSince  Time;
    C2gtFRTpFRH       Boolean;
    C2gtFRTpFRHSince  Time;
    RadioLinkFailure  Boolean;
    RelinquishableCell Boolean;
    RadioImprovableCell Boolean;
    RadioUsableCell   Boolean;
ENDNEWTYPE CellElemType;

SYNTYPE CellListIndexType =
    Natural CONSTANTS 0:7
ENDSYNTYPE;

NEWTYPE CellArrayType
    ARRAY (CellListIndexType, CellElemType)
ENDNEWTYPE CellArrayType;

NEWTYPE CellListType STRUCT
    Length Natural;
    Element CellArrayType;
ENDNEWTYPE CellListType;
```

Process Attachment_management

2(14)



/* *** LOCAL SYNONYM DEFINITIONS ***/

```
SYNONYM InitialCellIndex Natural = 0;
```

```
/* Time constants used for criteria evaluation */
```

```
SYNONYM 5seconds Duration = 5;
SYNONYM 15seconds Duration = 15;
SYNONYM 60seconds Duration = 60;
```

```
/* Channels supported by the MS for initial scanning according to Clause 18.3.4.1.
   NOTE: this definition is only informative. */
```

```
SYNONYM MaxChanIndex Natural = 3;
```

/* *** TIMER DEFINITIONS ***/

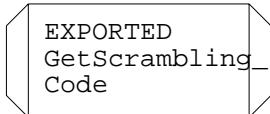
```
TIMER
  AM_T370 := T_370; /* Timer used for announced type 2 reselection */
```

/* *** EXPORT OF REMOTE PROCEDURES ***/

```
/* Remote procedure to update the serving cell
   descriptor with surveillance information
*/
```



```
/* Remote procedure to transfer scrambling code
   information from the data transfer process
*/
```



Process Attachment_management

3(14)

```
/* *** LOCAL VARIABLE DEFINITIONS ***/
```

```
/** LOCAL VARIABLES **/
```

```
DCL  
CellList CellListType; /* The selected and neighbour cell descriptor list */
```

```
/* Variables used for scanning and monitoring */  
  
DCL  
MonitoringComplete Boolean,  
MonitoringList MonListType, /* List of cells to be monitored */  
  
RankedMonitorList MonListType, /* List of currently ranked cells by monitoring(C2) */  
RankedScanList ScanListType, /* List of currently ranked cells by scanning (C1) */  
ScanCellIndex Natural, /* Cell list index of the cell currently being scanned. */  
Scanning Boolean := FALSE,  
ScanningComplete Boolean, /* Indicate whether all in the list has been scanned. */  
  
ScanningList ScanListType; /* List of cells to scan */
```

```
DCL  
MS_SupportedChannels ScanListType; /* The list of channels supported by the MS */
```

```
/* Variables for the cell reselection */  
  
DCL  
CellSelectedIndex Natural,  
CellSelectKind CellReselectionType,  
CR_Status Boolean, /* Status of the last performed cell  
reselection attempt. */  
CR_StayOnCell Boolean, /* If Type 1 or 2 cell reselection fails this variable  
indicate whether to continuously stay on the current cell. */  
  
NB_CellInfo_Available Boolean := FALSE,  
OldCellIndex Natural,  
PrefNbCellIndex Natural, /* Cell array index of the preferred neighbour  
cell to select next */  
PrefNbCellFound Boolean,  
LastReselectionTime Time; /* Timestamp for last cell reselection */
```

Process Attachment_management

4(14)



```
/** MLE PDU DATA VARIABLES **/
```

```
DCL
  D_SIN2_PDU      MLE_D_SIN2Type,
  D_NWRK_PDU      MLE_D_NWRK_BroadcastType,
  D_SIN1_PDU      MLE_D_SIN1Type;
```

```
/** MLE SP DATA VARIABLES **/
```

```
DCL
  ActivateReq     MLE_ActivateReqType,
  UpdateReq       MLE_UpdateReqType;
```

```
/** LLC SP DATA VARIABLES **/
```

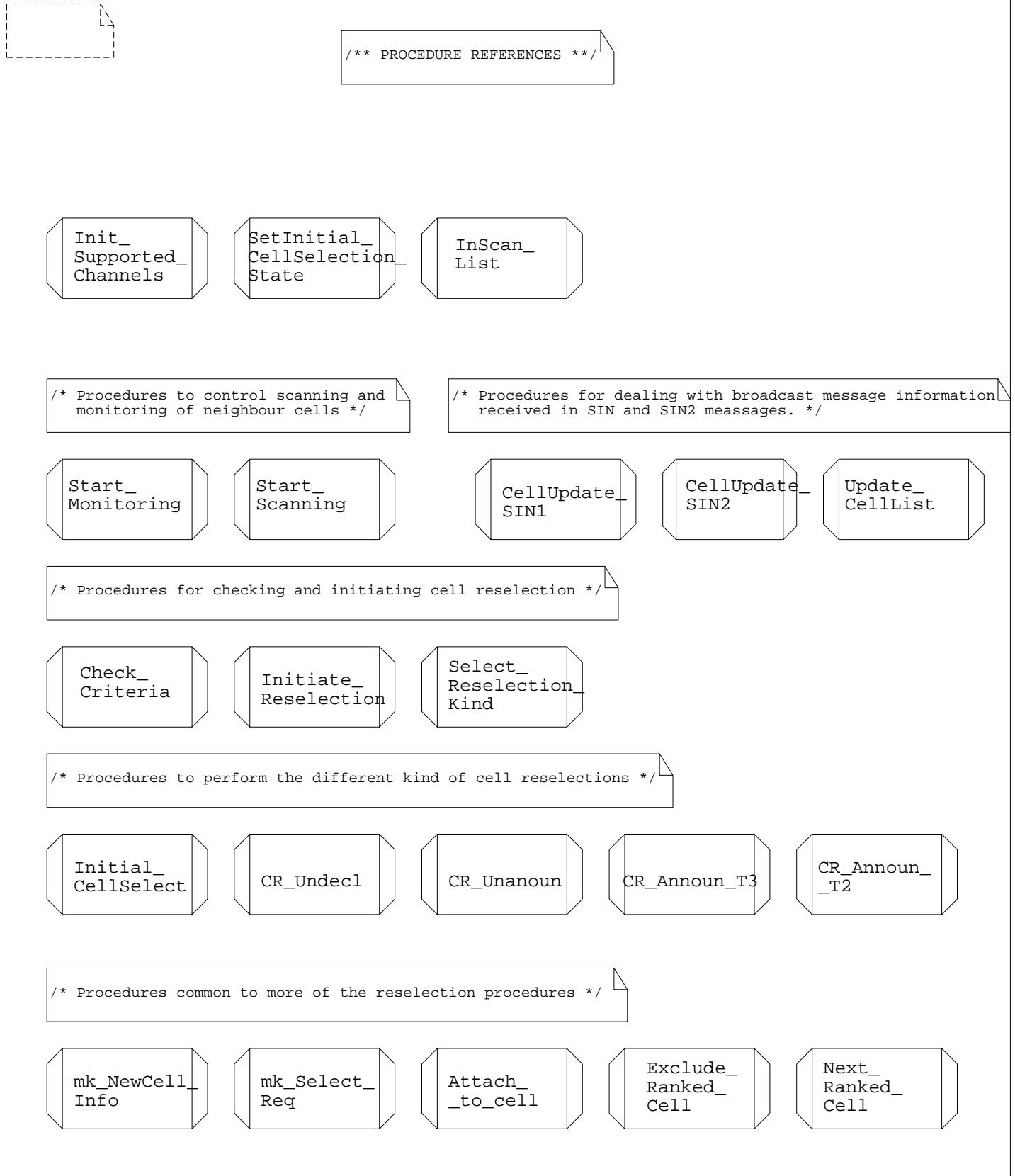
```
DCL
  RepIndication   TLC_ReportIndicationType;
```

```
*** IMPORT OF REMOTE VARIABLES AND PROCEDURES ***/
```

```
IMPORTED is_CONP_InProgress Boolean; /* Check if any CONP transfer in progress */
```

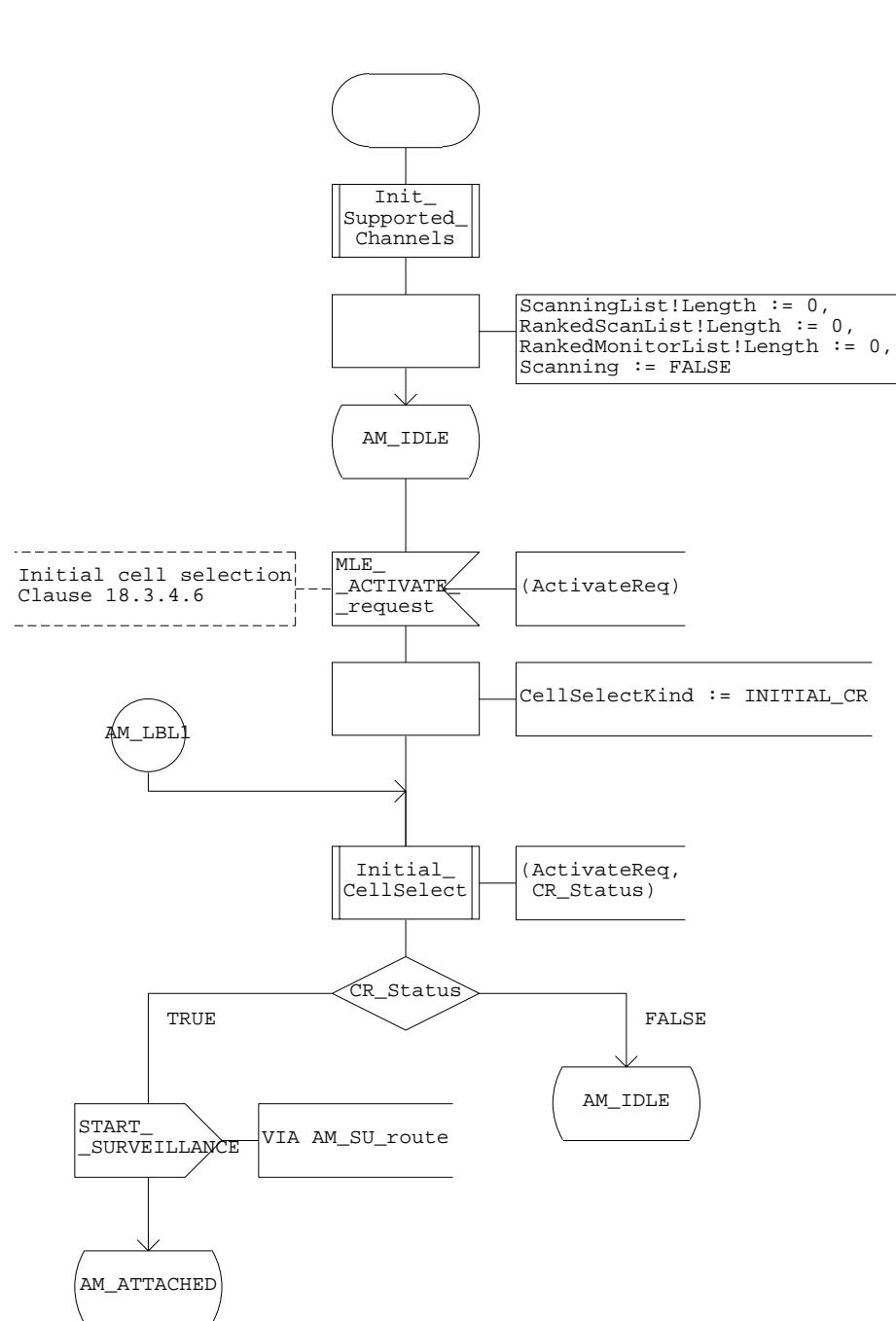
Process Attachment_management

5 (14)



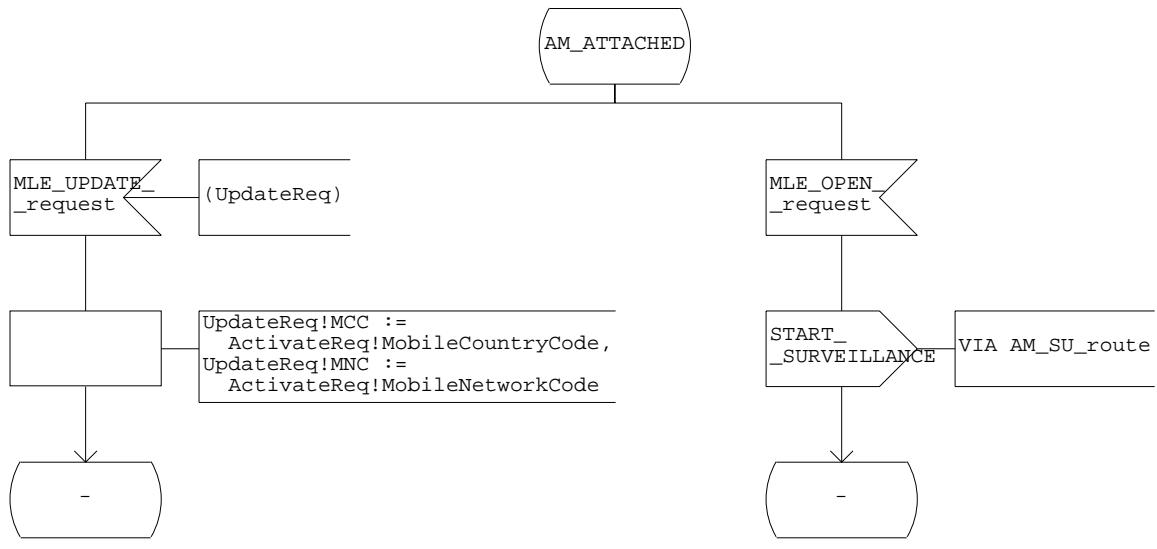
Process Attachment_management

6 (14)



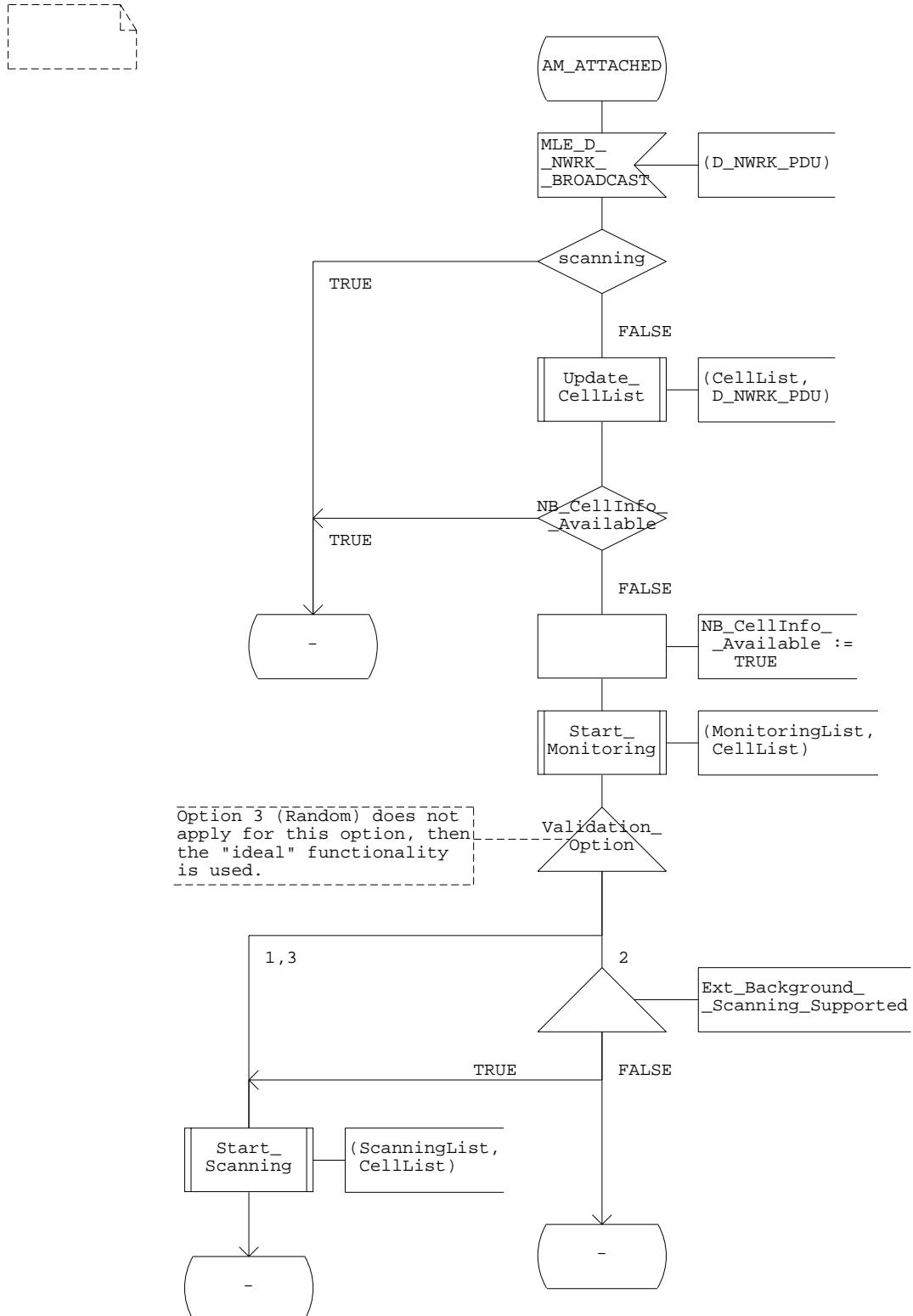
Process Attachment_management

7 (14)



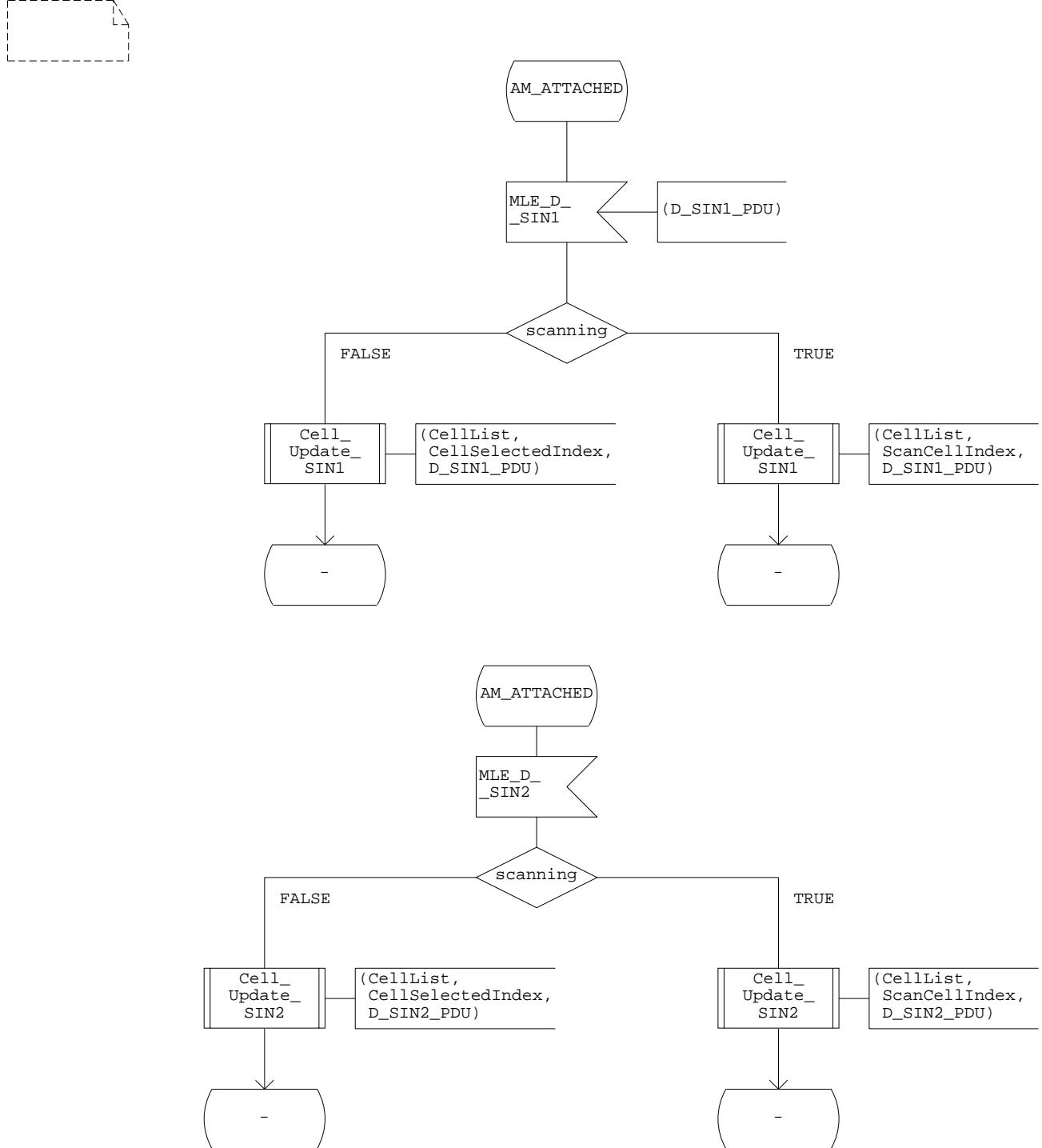
Process Attachment_management

8 (14)



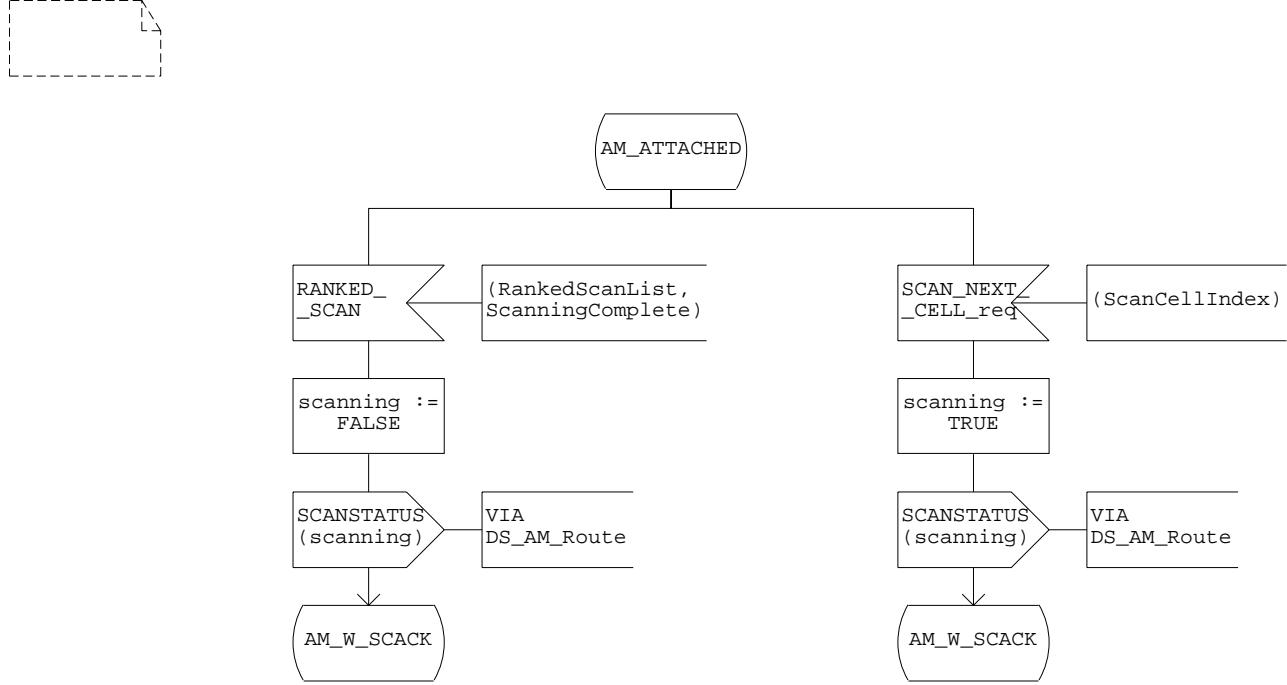
Process Attachment_management

9 (14)



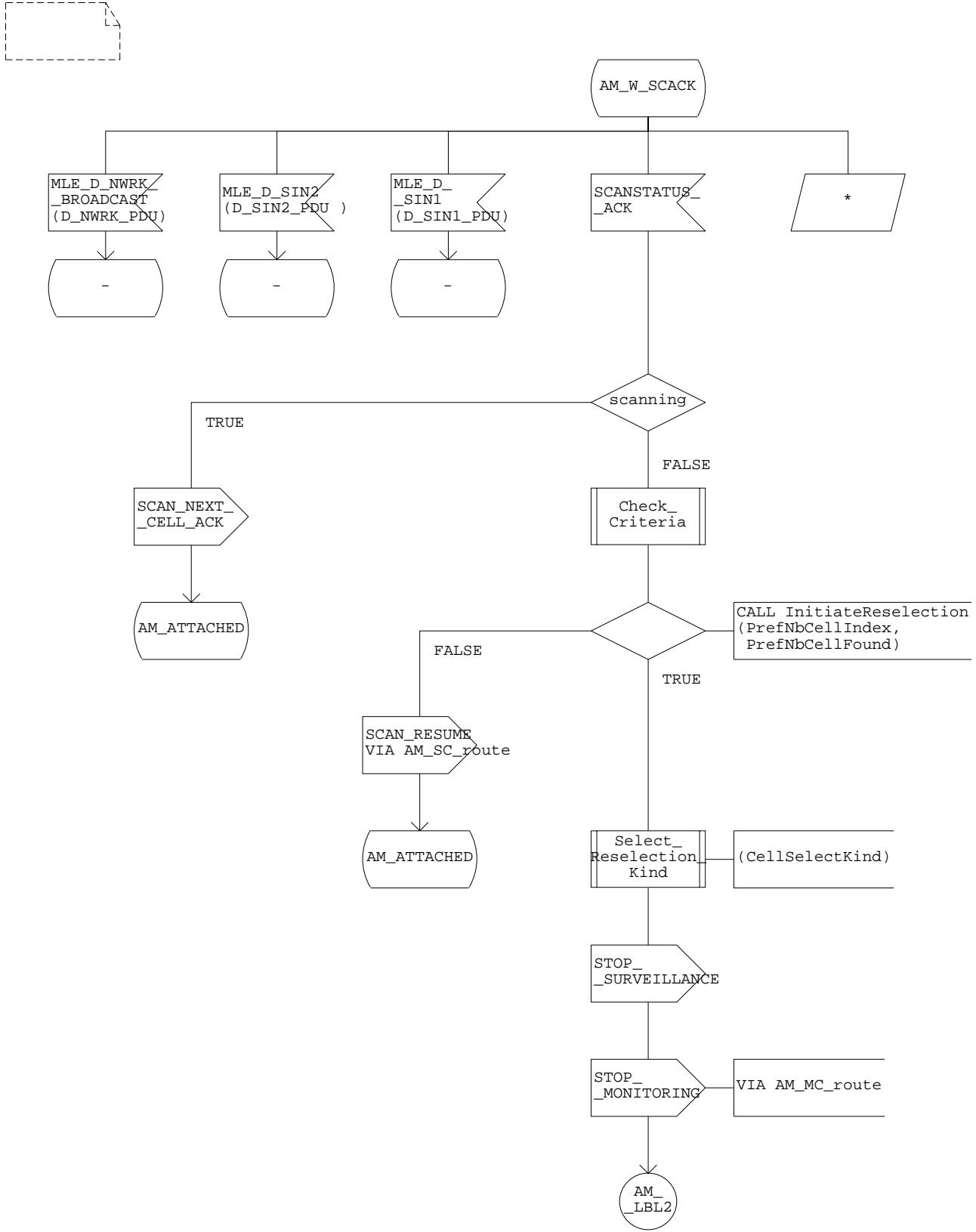
Process Attachment_management

10 (14)



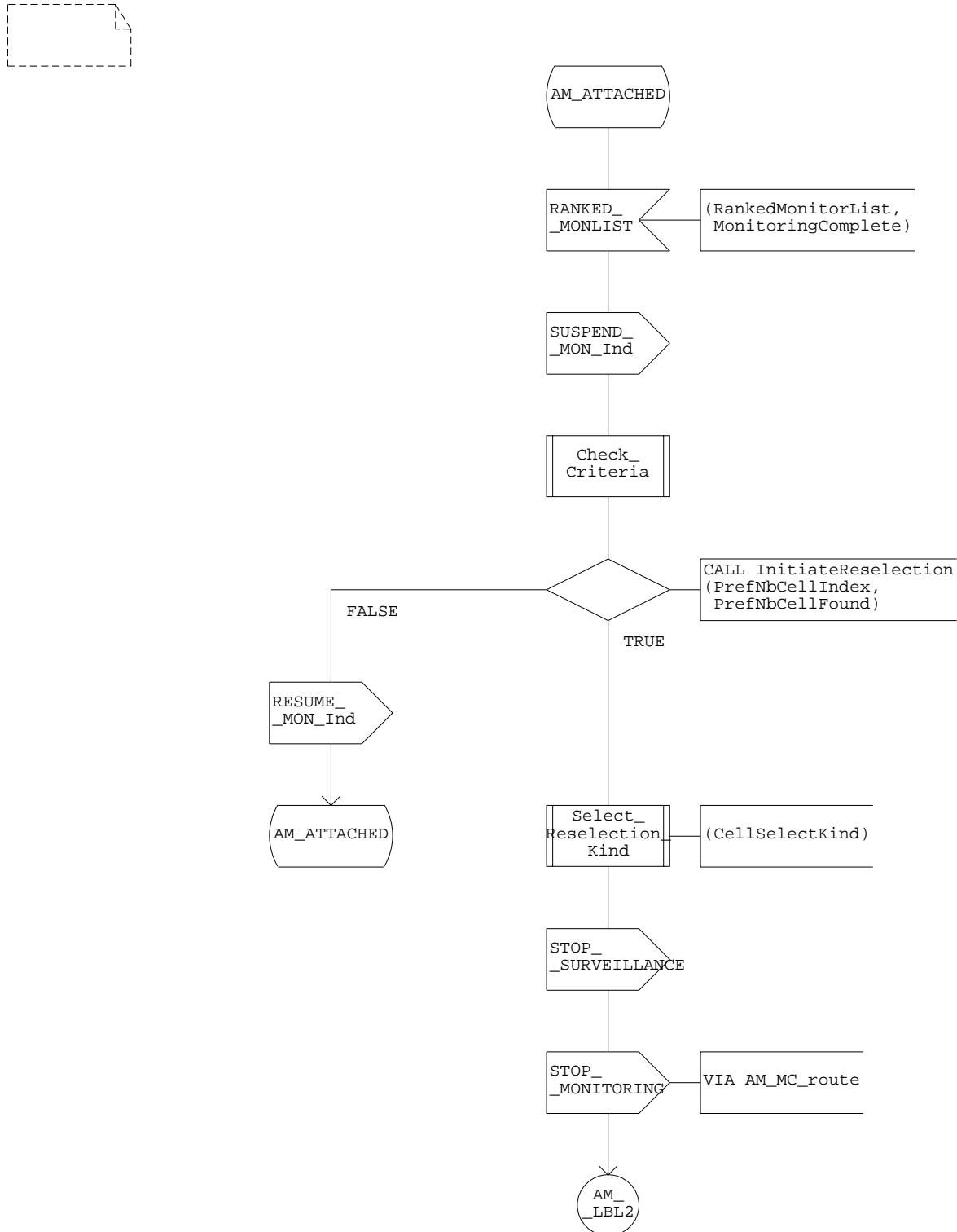
Process Attachment_management

11(14)



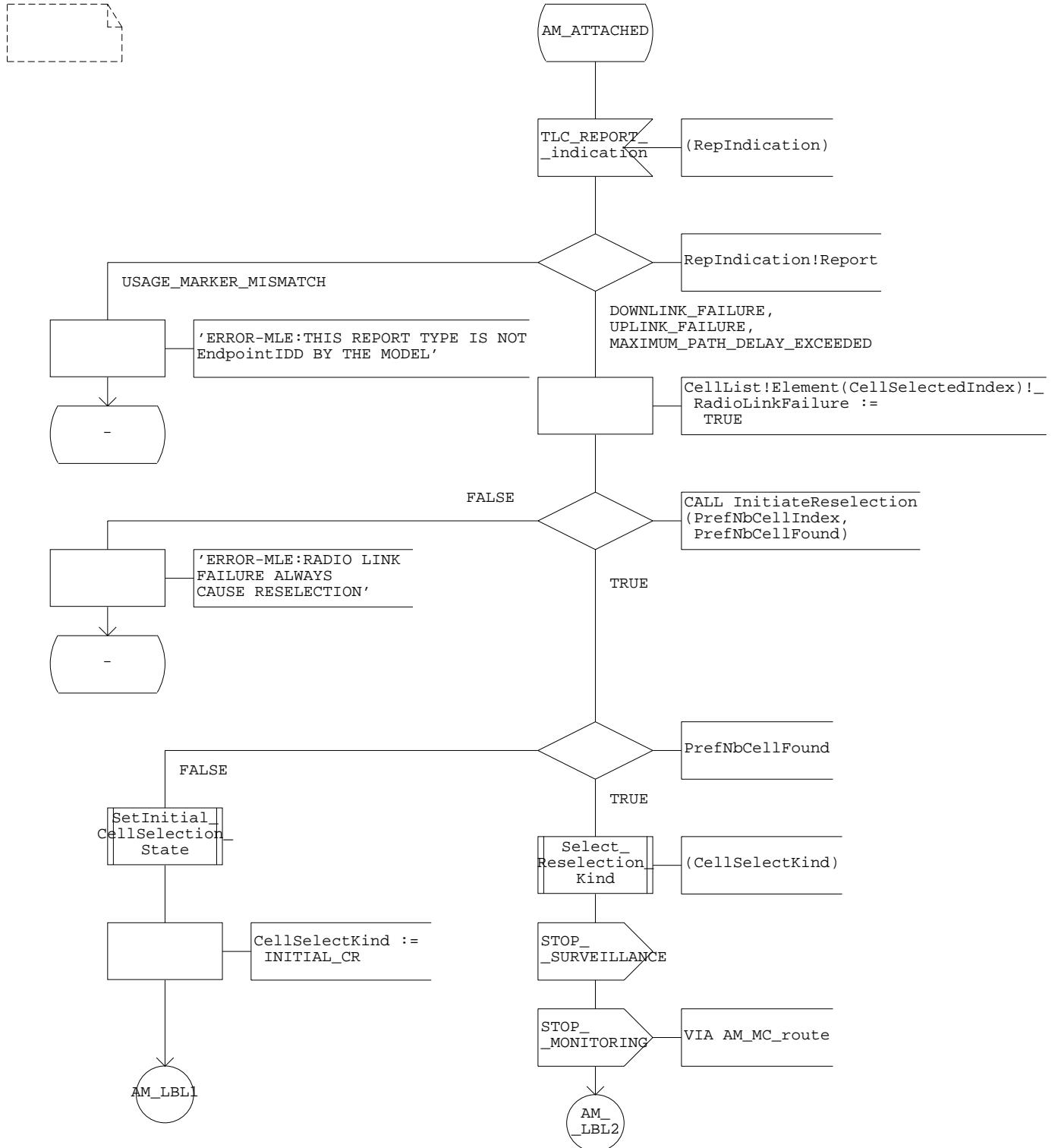
Process Attachment_management

12(14)



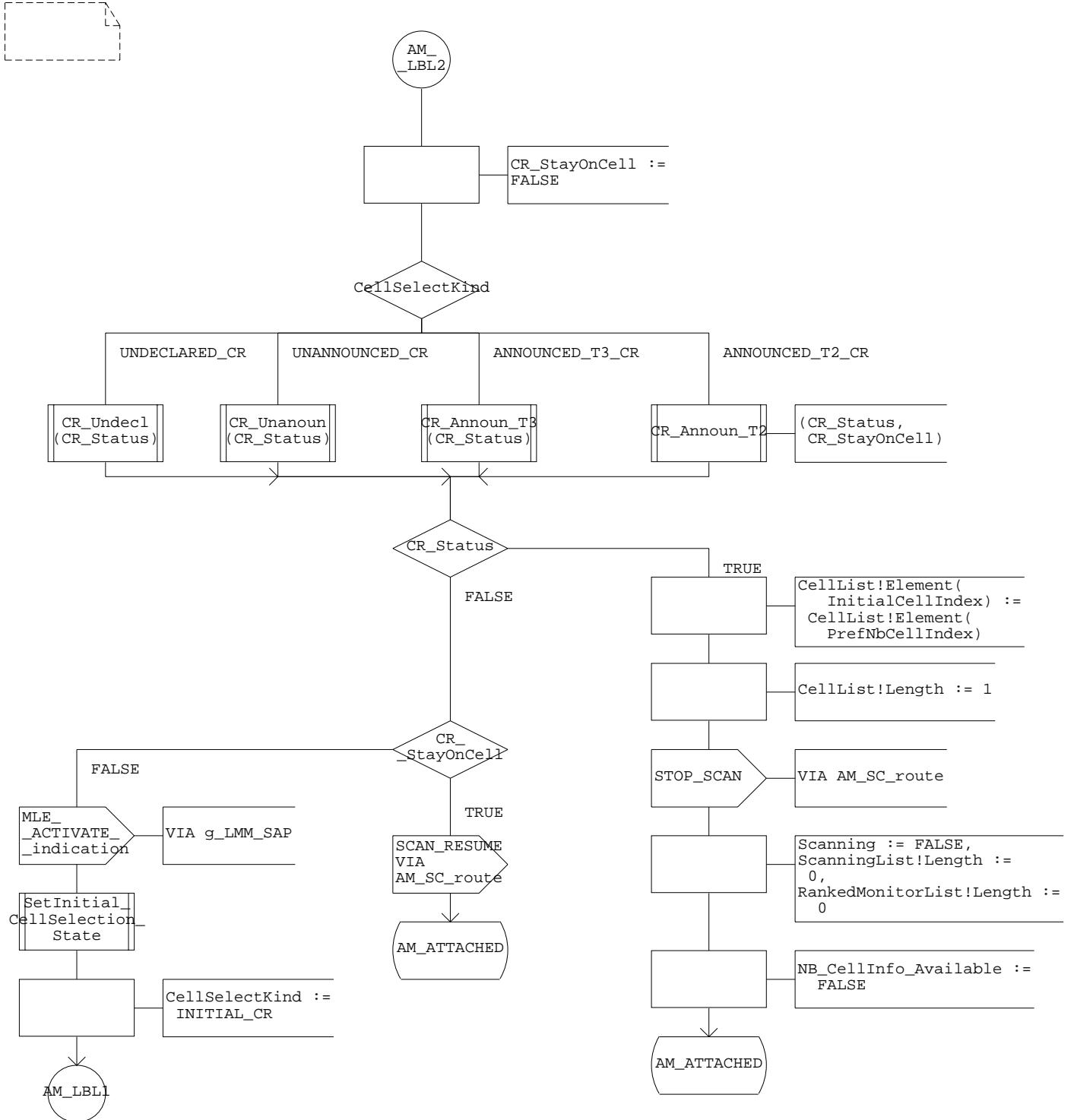
Process Attachment_management

13(14)



Process Attachment_management

14 (14)

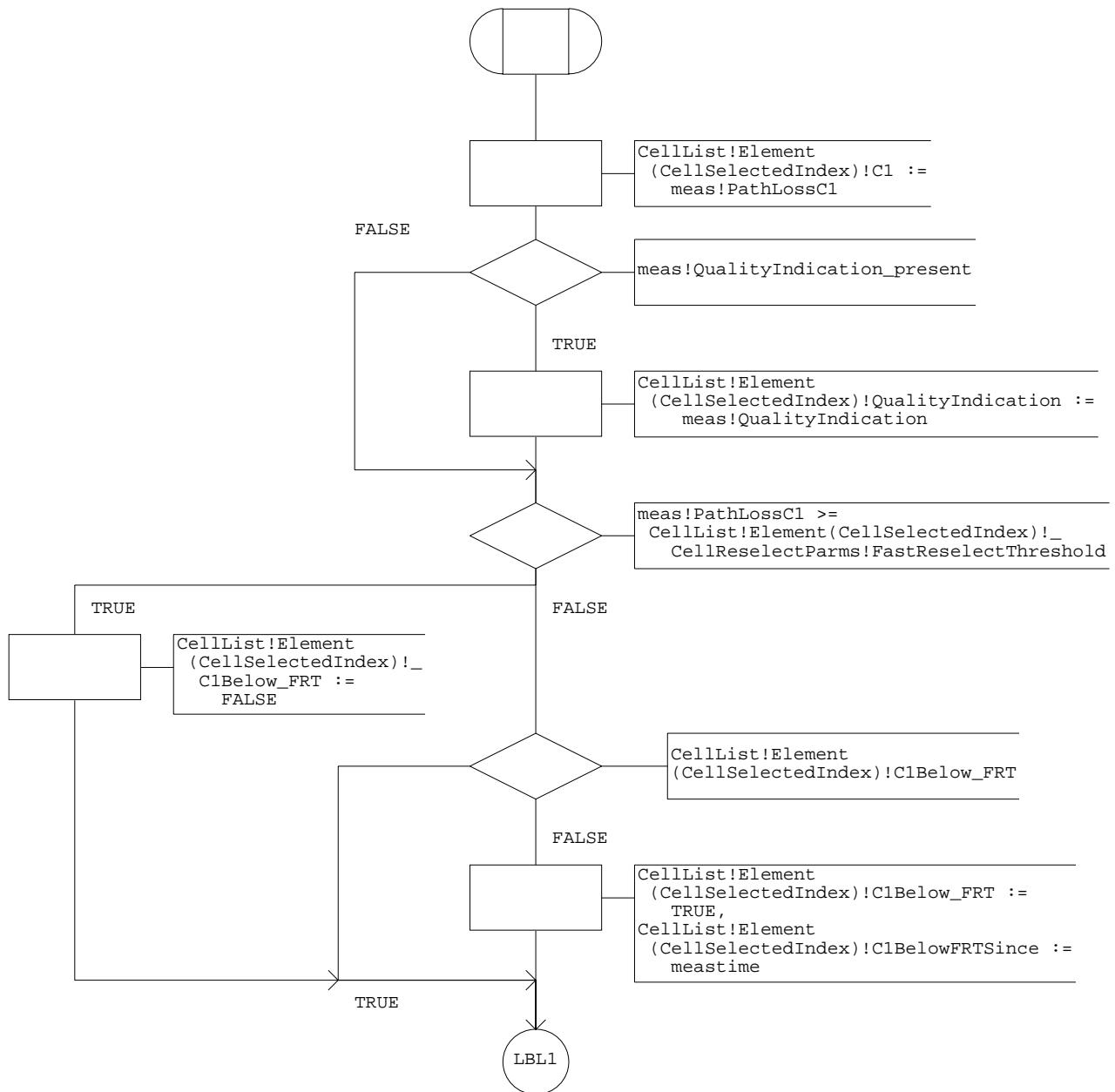


EXPORTED Procedure SurveilInfo

1(2)

```
; FPAR
  IN meas TLC_ServingIndicationType;
  IN meastime Time;
```

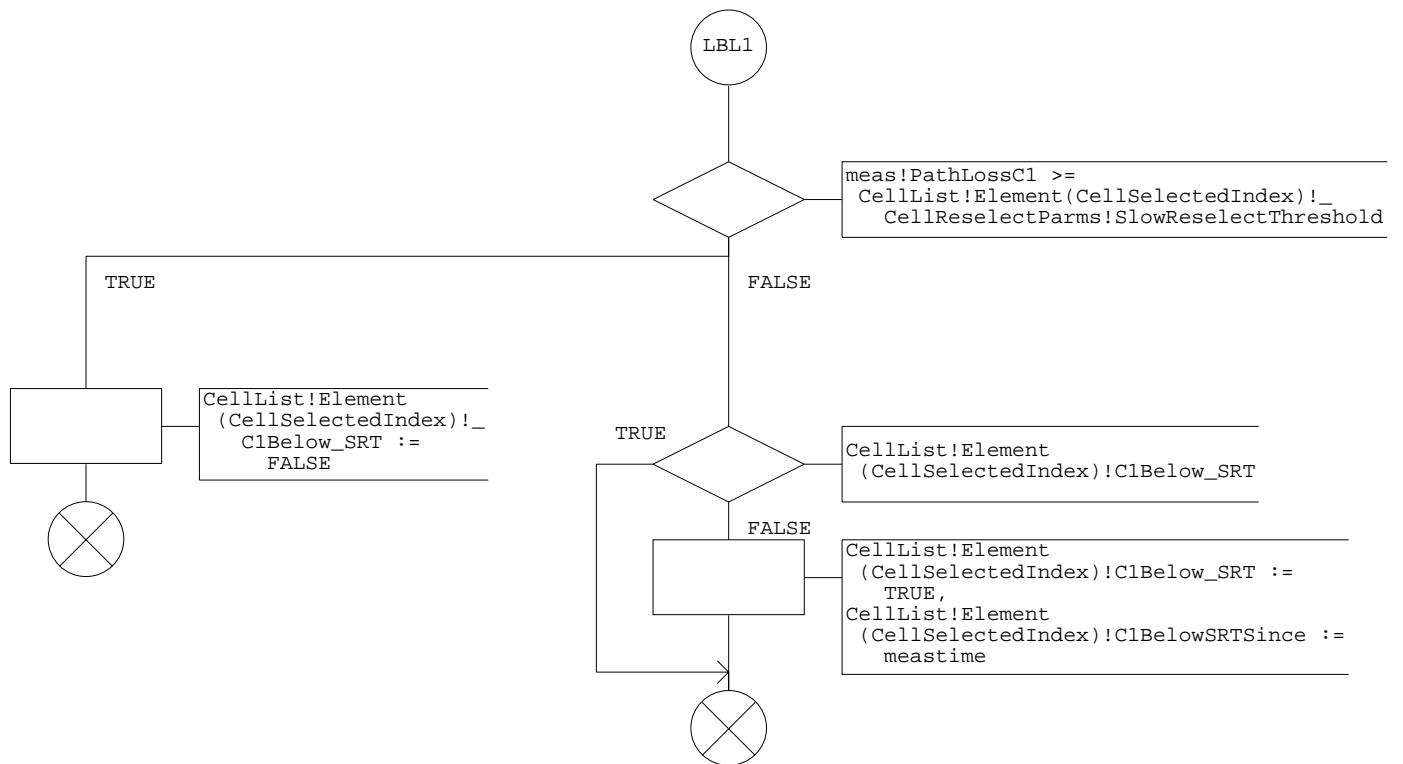
```
/* Update the serving cell descriptor
   with surveillance information
*/
```



EXPORTED Procedure SurveilInfo

2(2)

```
; FPAR
  IN meas TLC_ServingIndicationType;
  IN meastime Time;
```

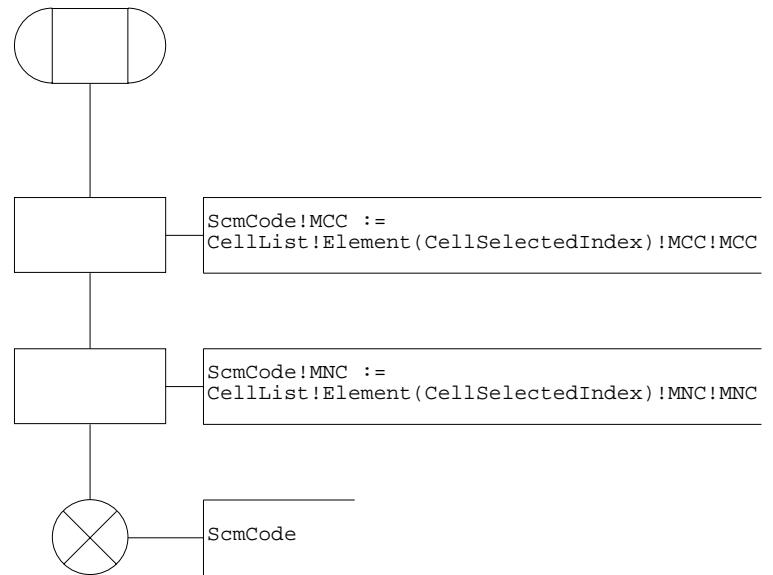


```
EXPORTED Procedure GetScramblingCode
```

```
1(1)
```

```
; RETURNS ScramblingCodeType;
```

```
DCL  
  ScmCode ScramblingCodeType;
```



Procedure InitSupportedChannels

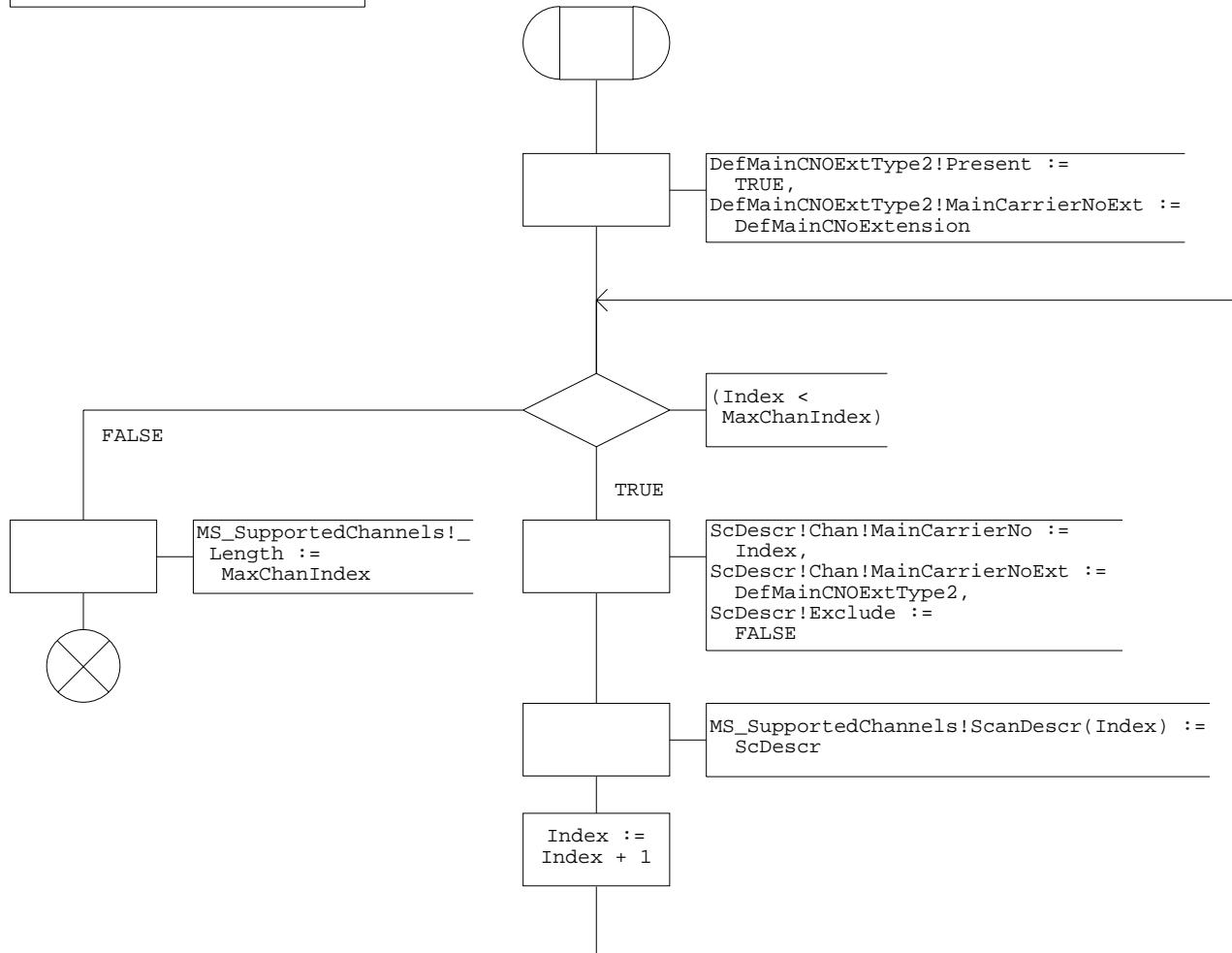
1(1)

```
/* Initialise the variable MS_SupportedChannels
   that defines the channels supported by the MS.

   NOTE: The values defined is only informative,
   they are not required according to the ETS.
*/
```

```
DCL DefMainCNoExtension MainCarrierNoExtType := (. 0,0,0,0 .);
DCL DefMainCNOExtType2 MainCarrierNoExtType2;
```

```
DCL
  ScDescr ScanCellDescrType,
  Index Natural := 0;
```

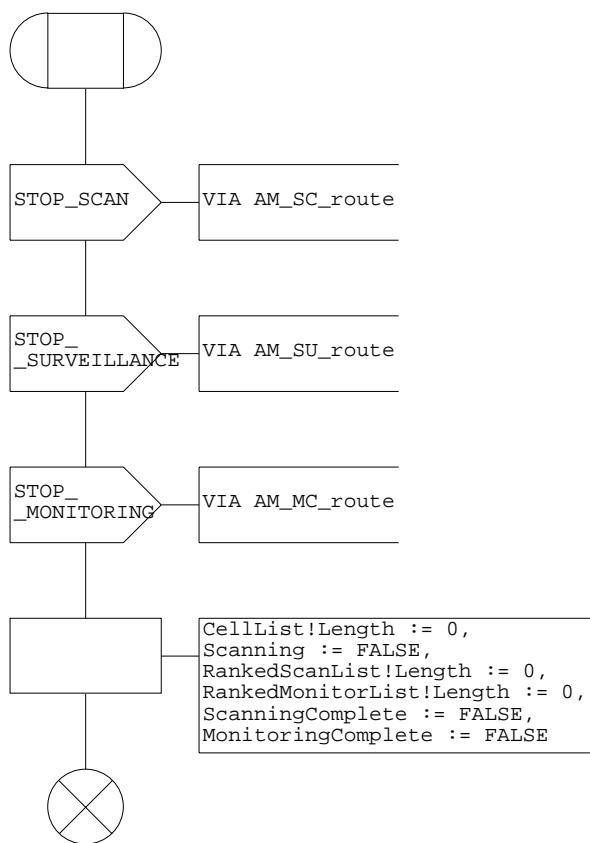


Procedure SetInitialCellSelectionState

1(1)



```
/* Set/reset the parameters to  
allow for a new forced initial  
cell selection procedure to be  
performed.  
*/
```

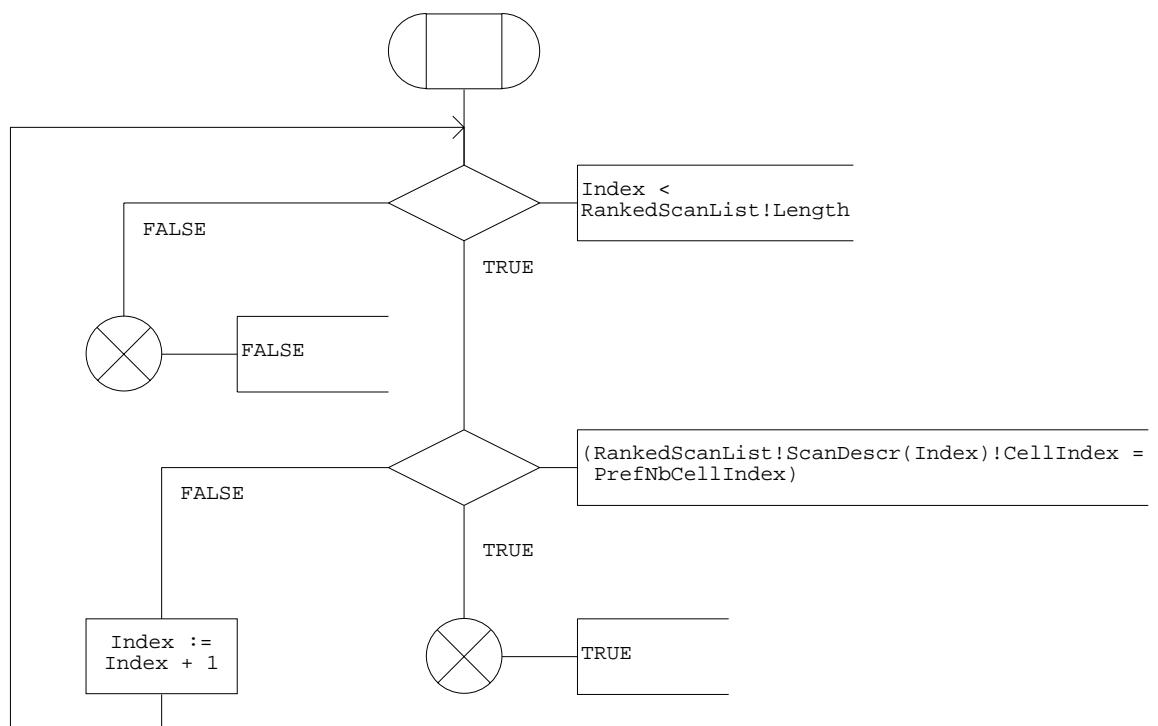


Procedure InScanList

1(1)

```
; FPAR
  IN PrefNbCellIndex Natural;
  RETURNS Boolean;
  /* Check if the preferred neighbourcell
   has been scanned, i.e. is in the
   list of ranked cells.
  */
```

```
DCL
  Index Natural := 0;
```



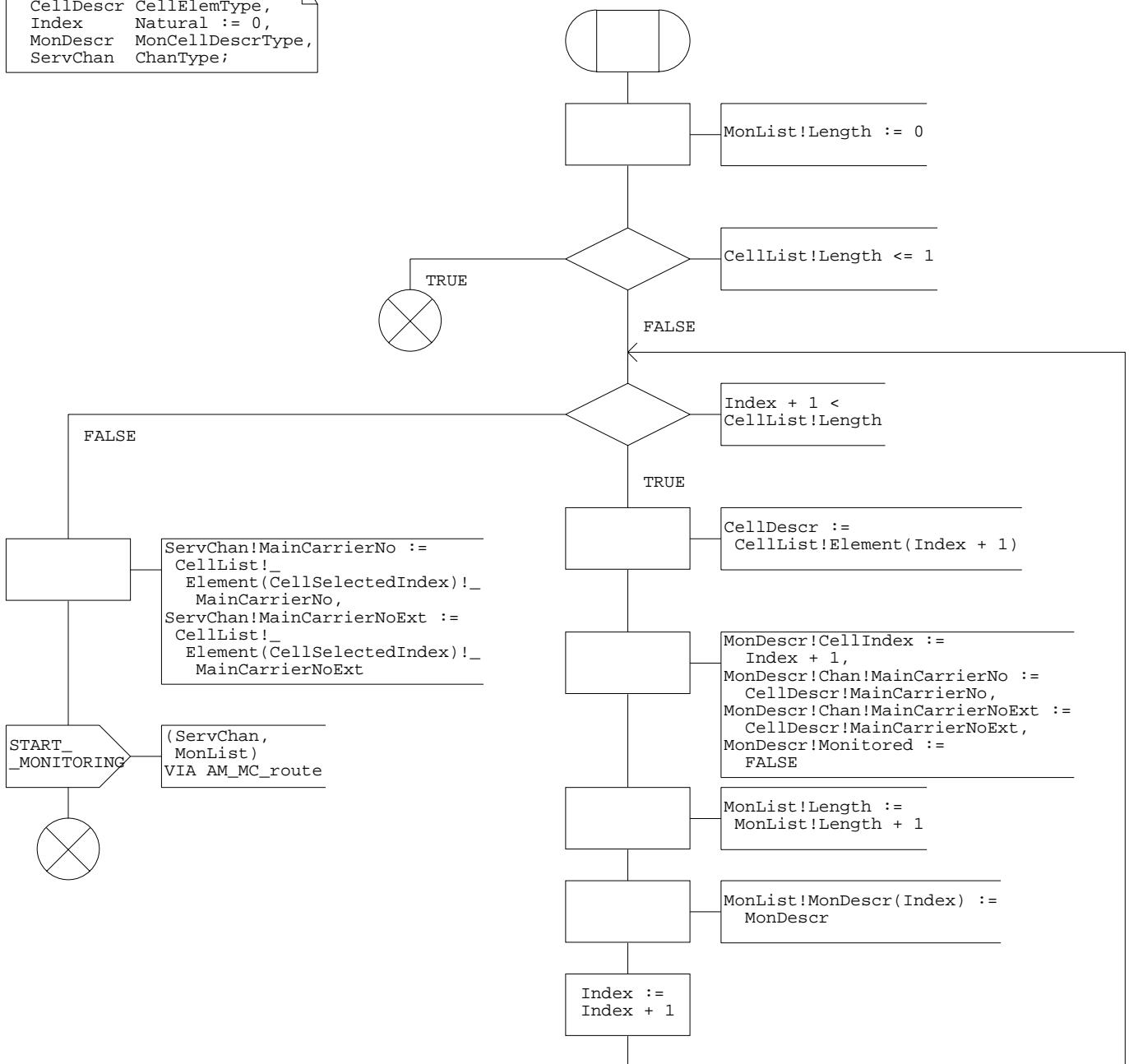
Procedure StartMonitoring

1(1)

```
; FPAR
  IN/OUT MonList MonListType;
  IN CellList CellListType;
```

```
/* Build monitor list from the cell
list descriptor and initiate
monitoring.
```

```
DCL
CellDescr CellElemType,
Index Natural := 0,
MonDescr MonCellDescrType,
ServChan ChanType;
```



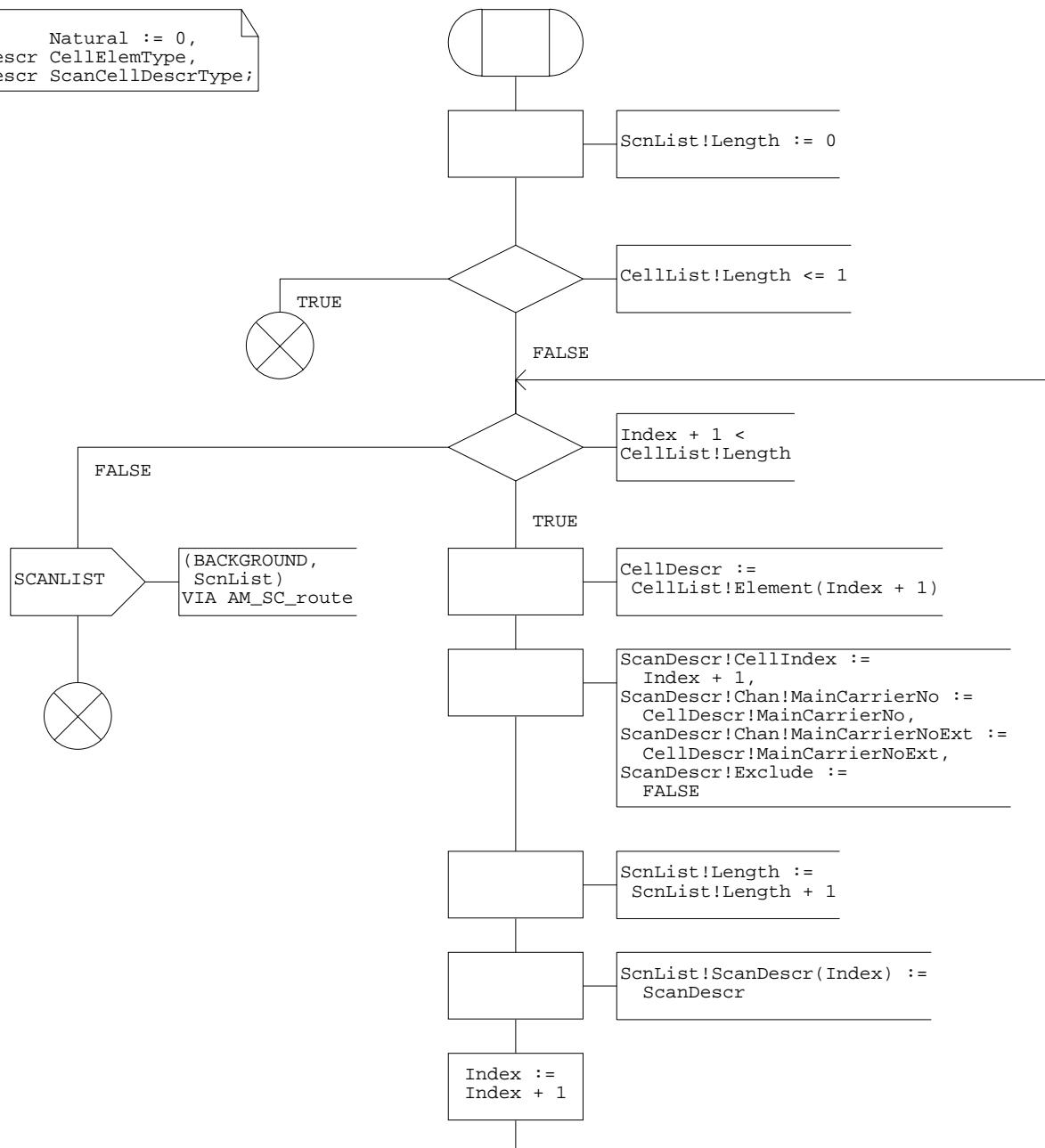
Procedure StartScanning

1(1)

```
; FPAR
  IN/OUT ScnList ScanListType;
  IN CellList CellListType;
```

```
/* Build scan list from the cell list descriptor and initiate
   background scanning of the neighbour cells in the scan list.
*/
```

```
DCL
Index Natural := 0,
CellDescr CellElemType,
ScanDescr ScanCellDescrType;
```

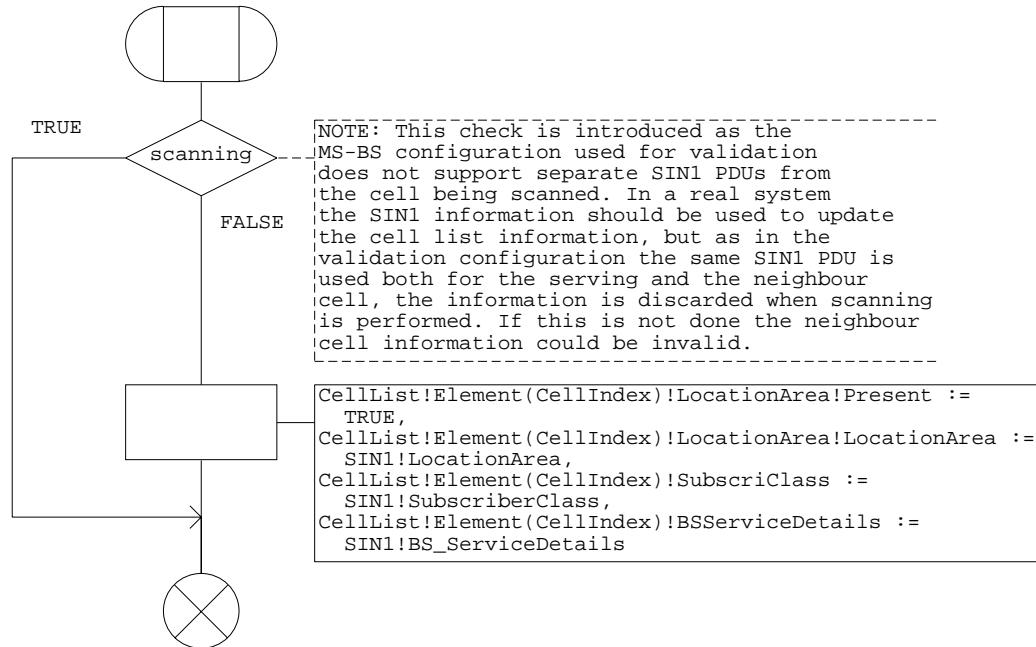


Procedure CellUpdateSIN1

1(1)

```
; FPAR
  IN/OUT CellList CellListType,
  IN CellIndex Natural,
  IN SIN1 MLE_D_SIN1Type;
```

```
/* Update the cell list
   element with the information
   received in the system
   information PDU.
*/
```

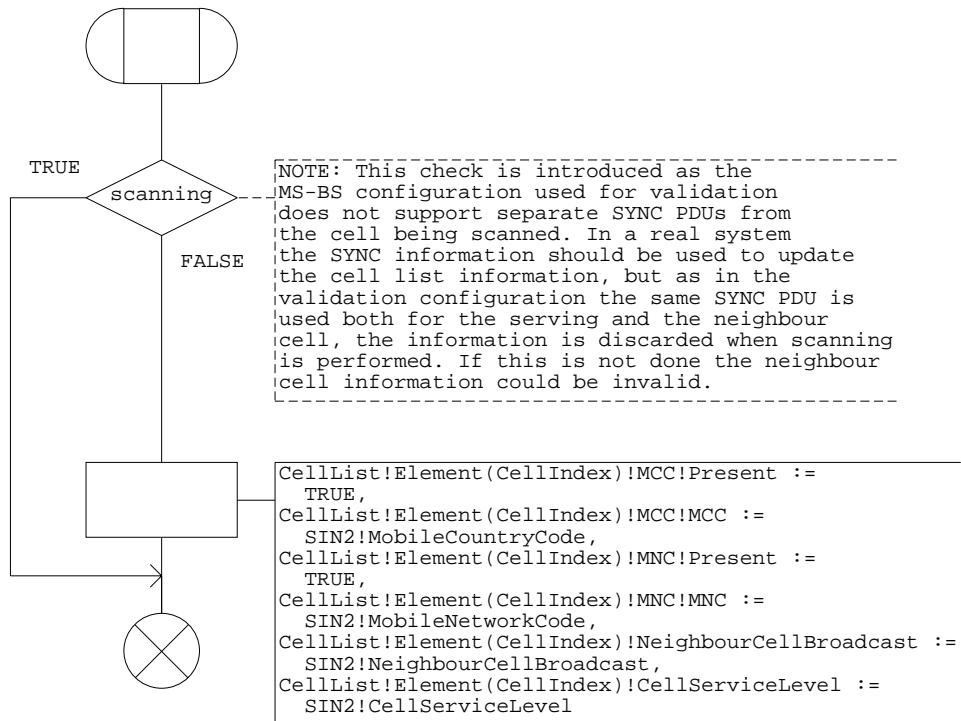


Procedure CellUpdateSIN2

1(1)

```
; FPAR
  IN/OUT CellList CellListType;
  IN CellIndex Natural,
  IN SIN2 MLE_D_SIN2Type;
```

```
/* Update the cell descriptor with the
   information received in the SIN2 PDU.
*/
```



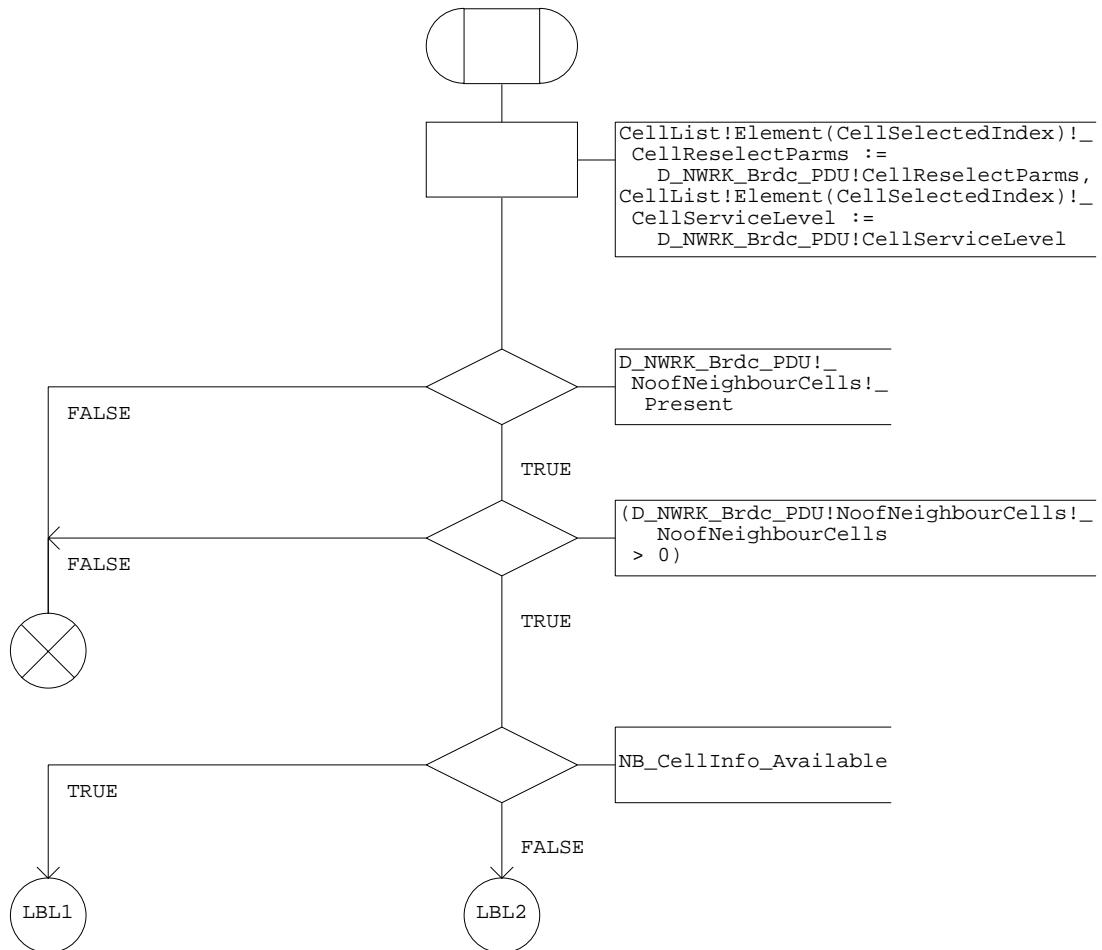
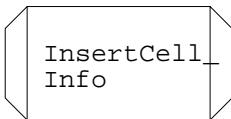
Procedure UpdateCellList

1 (2)

```
; FPAR
  IN/OUT CellList CellListType,
  IN D_NWRK_Brdc_PDU MLE_D_NWRK_BroadcastType;
```

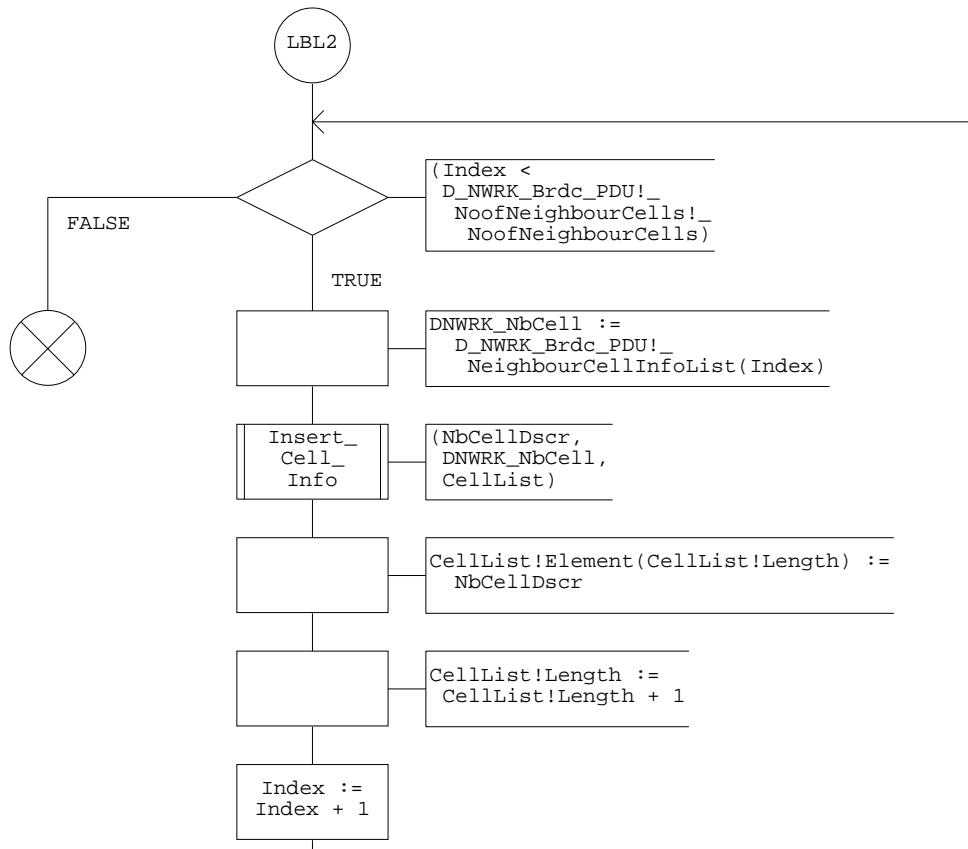
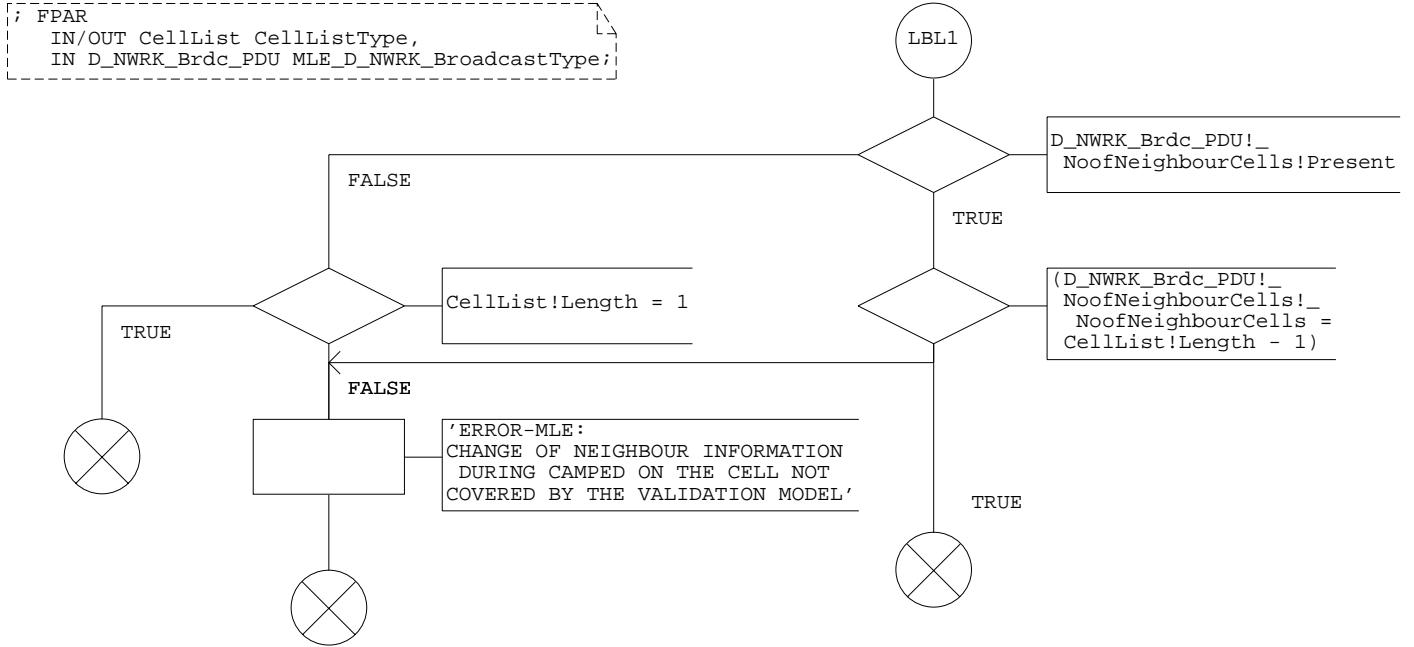
```
/* Update the serving cell descriptor with the
information received in the SIN2
PDU. Add the neighbour cell descriptions if
not already in the cell list. If the
information is already there check that it
is consistent with the information received
in the SIN2 PDU.
```

```
DCL
  DNWRK_NbCell MLE_NeighbourCellInfoType,
  Index Natural := 0,
  NbCellDscr CellElemType;
```



Procedure UpdateCellList

2 (2)

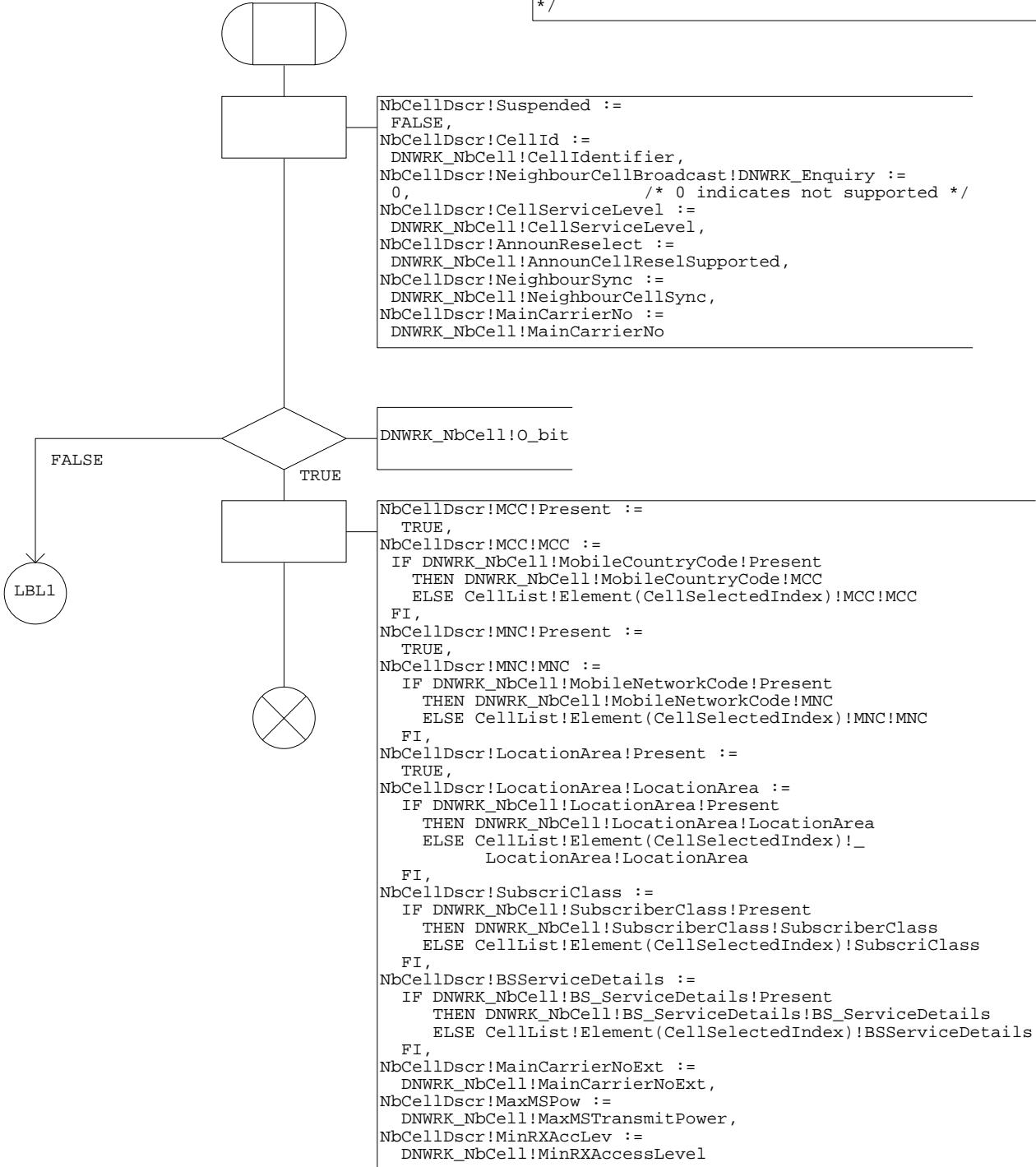


Procedure InsertCellInfo

1 (2)

```
; FPAR
  IN/OUT NbCellDscr CellElemType,
  IN DNWRK_NbCell MLE_NeighbourCellInfoType,
  IN/OUT CellList CellListType;
```

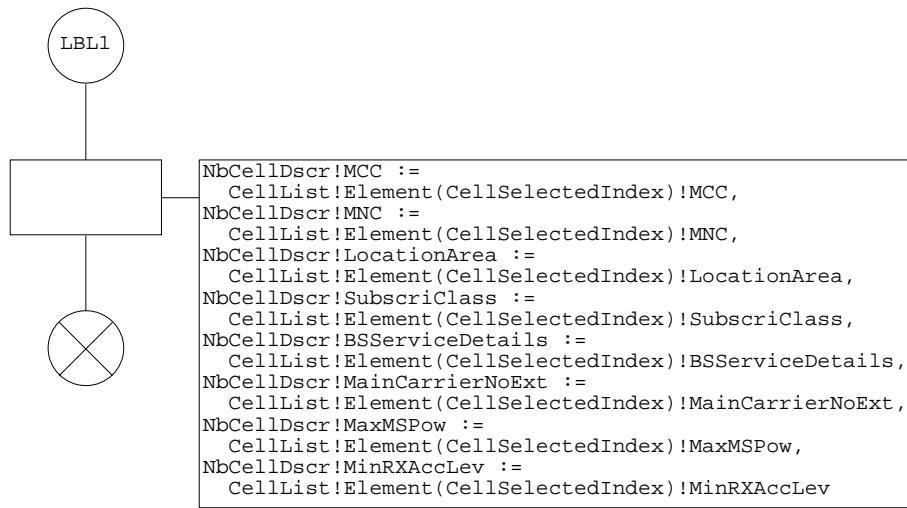
```
/* Insert the information from the neighbour cell
   information in a D_SIN2 PDU in a new
   cell list element. If no type2 elements are
   supported the parameters of the serving cell are
   used for the neighbour cell descriptor
*/
```



Procedure InsertCellInfo

2(2)

```
; FPAR
  IN/OUT NbCellDscr CellElemType,
  IN DNWRK_NbCell MLE_NeighbourCellInfoType,
  IN/OUT CellList CellListType;
```



Procedure CheckCriteria

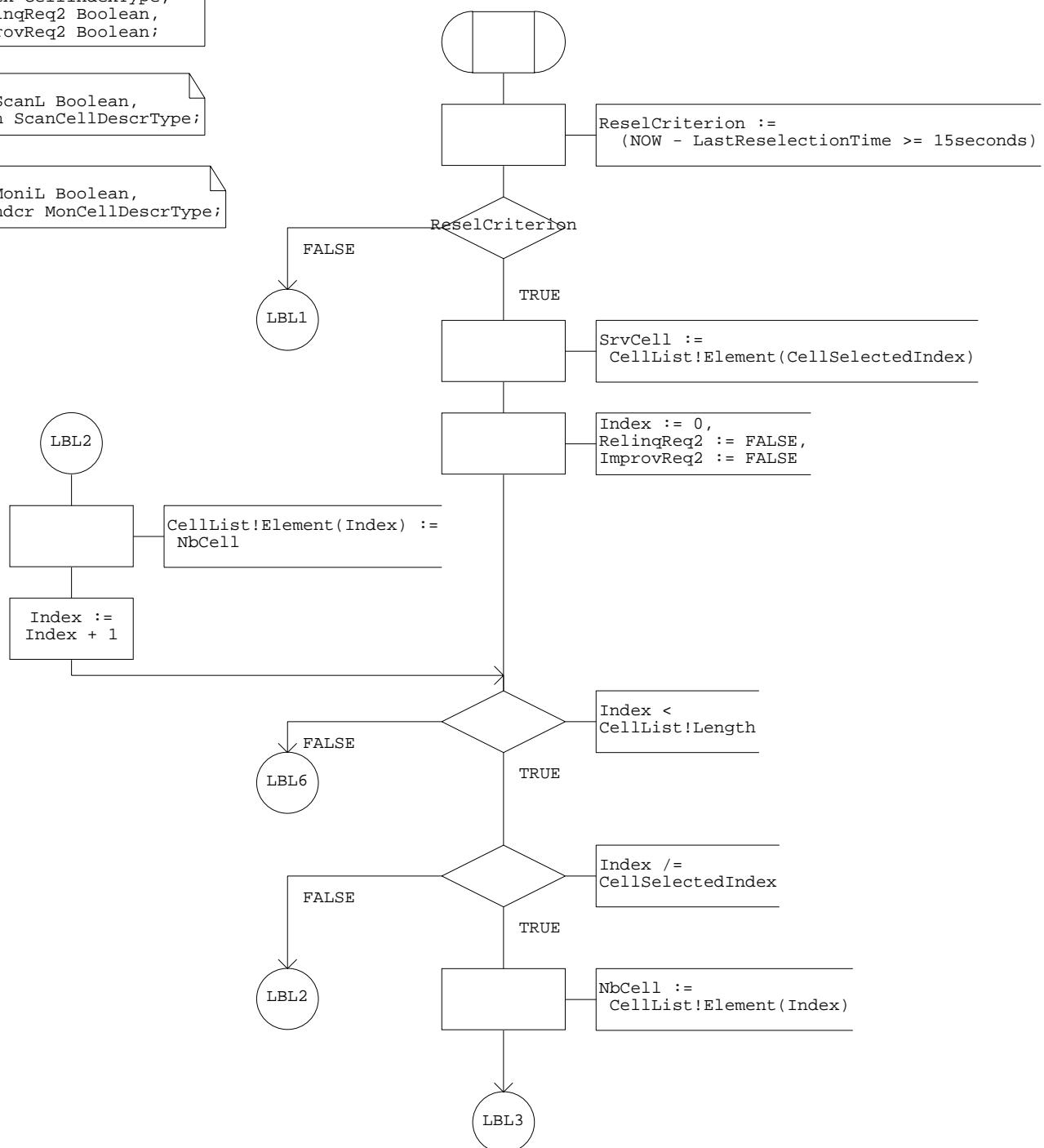
1(6)

```
/* Check the criteria for
 - Radio relinquishable cell; 18.3.4.5.4
 - Radio improvable cell; 18.3.4.5.5
 - Radio usable cell; 18.3.4.5.6
 And update the cell list with the scanned
 and monitored information.
 */
```

```
DCL
 SrvCell CellElemType,
 NbCell CellElemType,
 ReselCriterion Boolean,
 Index CellIndexType,
 RelinqReq2 Boolean,
 ImprovReq2 Boolean;
```

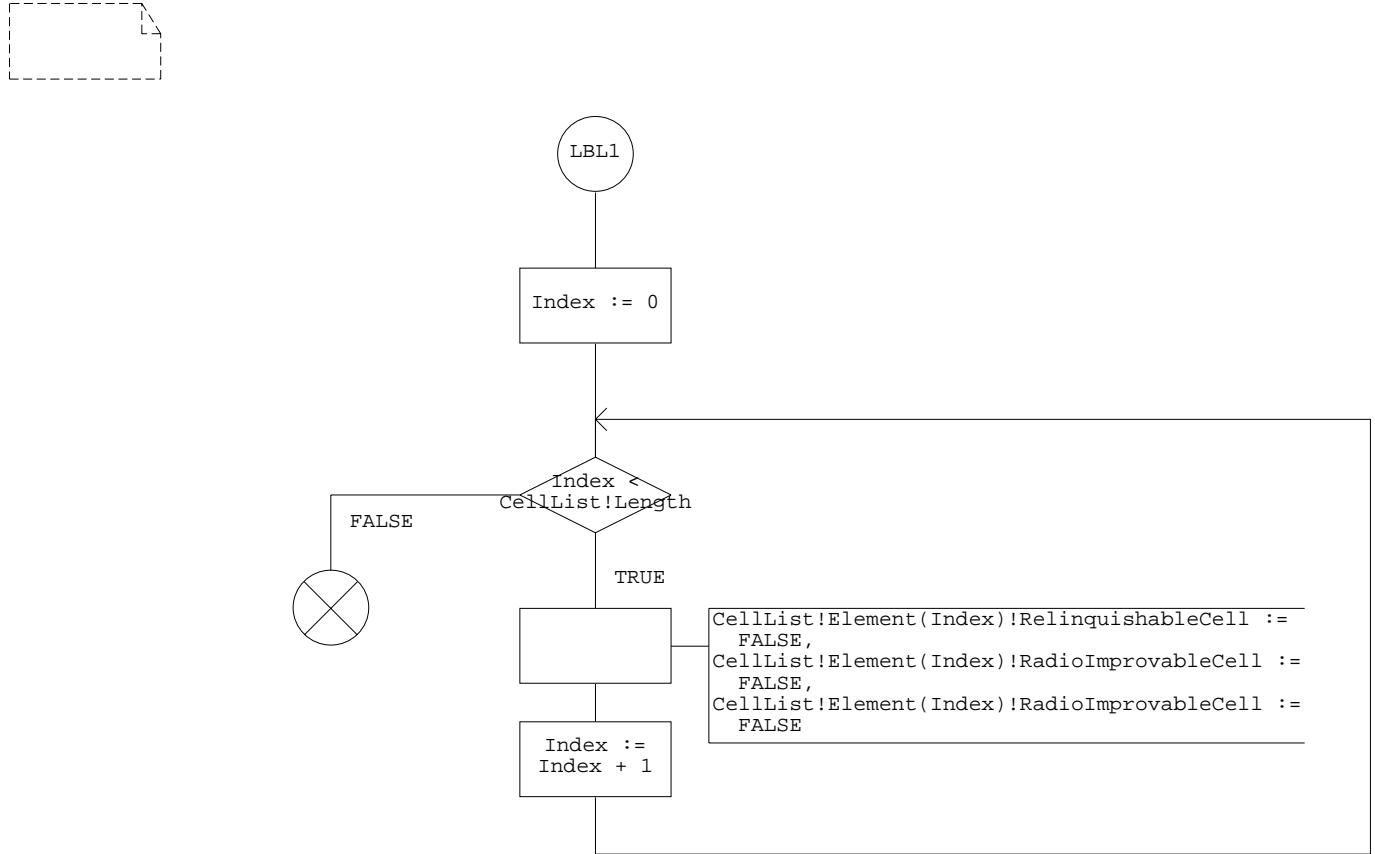
```
DCL
 InScanL Boolean,
 scn ScanCellDescrType;
```

```
DCL
 InMoniL Boolean,
 mondcr MonCellDescrType;
```



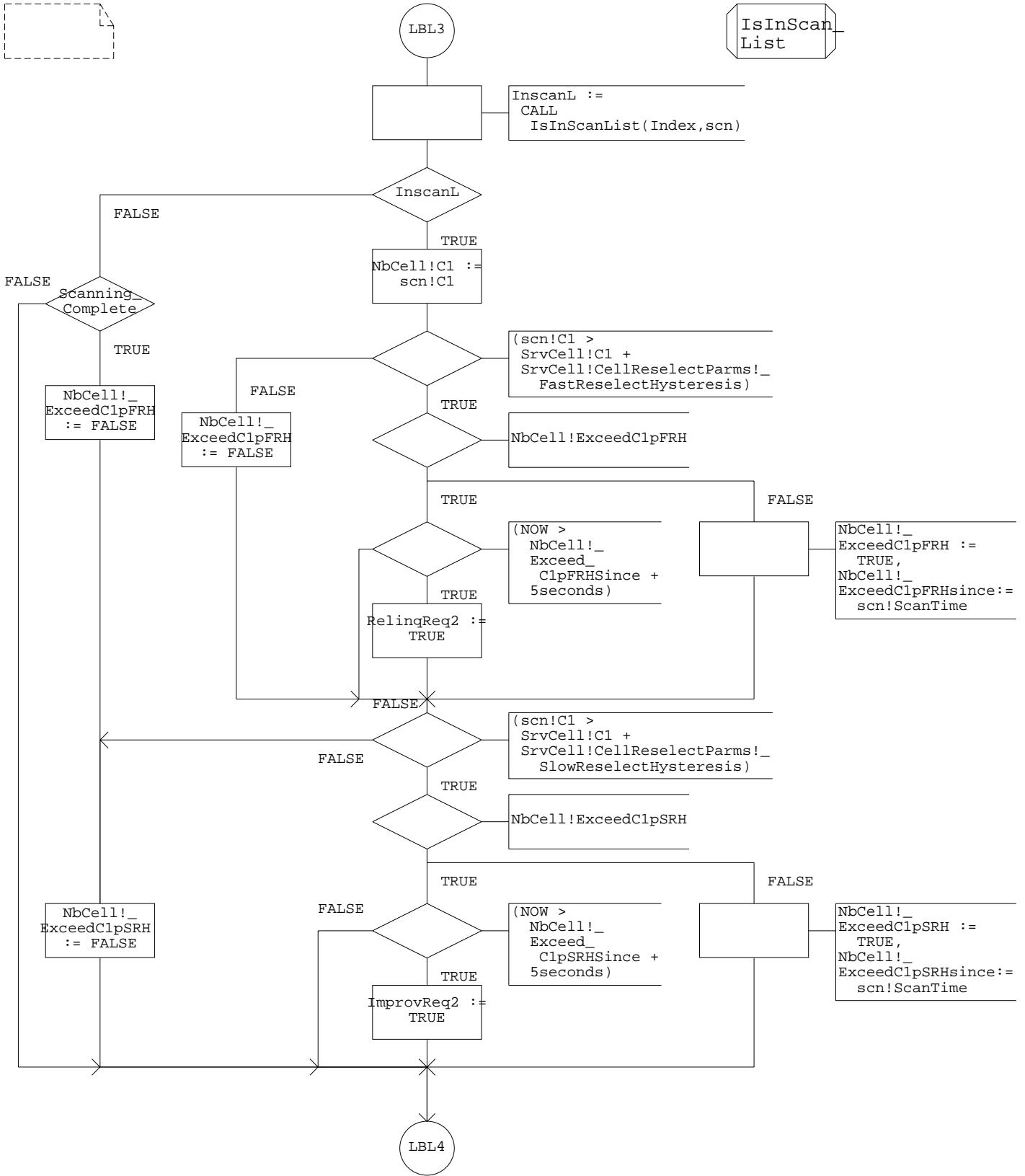
Procedure CheckCriteria

2(6)



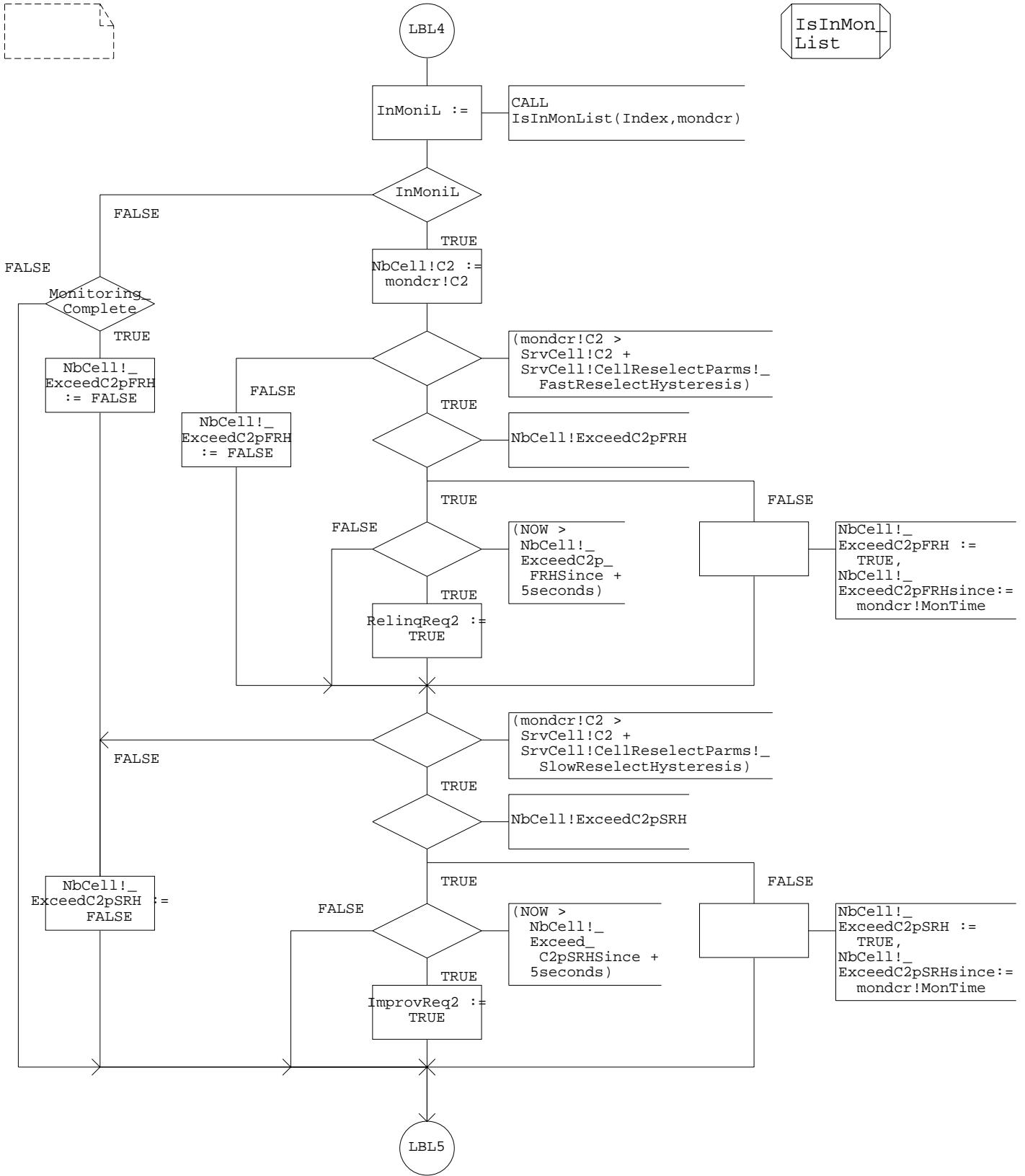
Procedure CheckCriteria

3 (6)



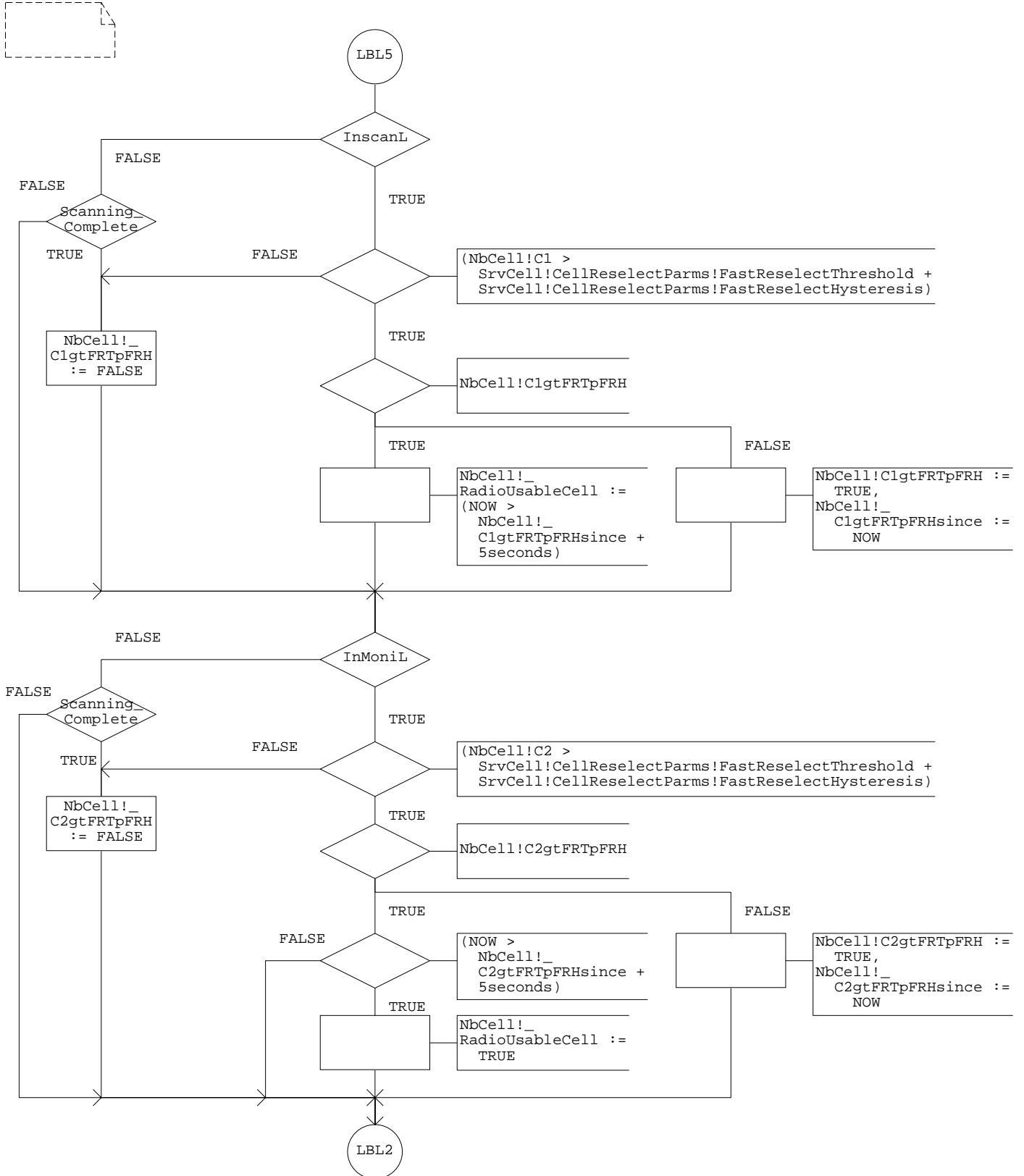
Procedure CheckCriteria

4 (6)



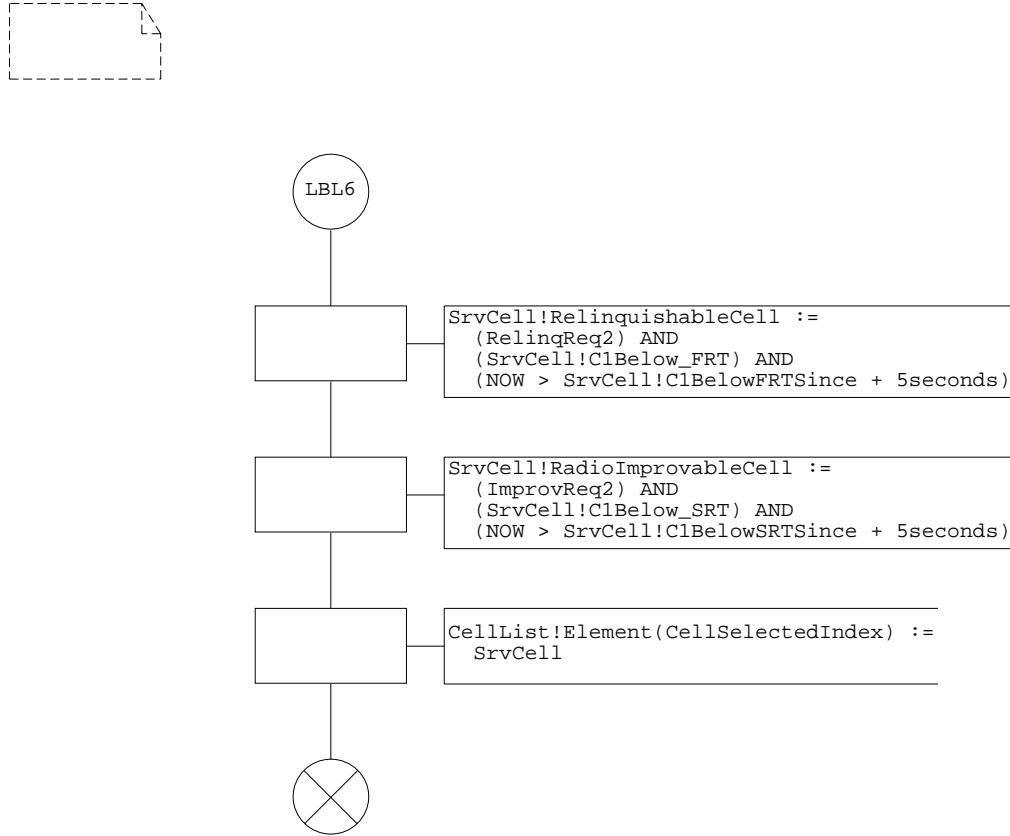
Procedure CheckCriteria

5 (6)



Procedure CheckCriteria

6(6)



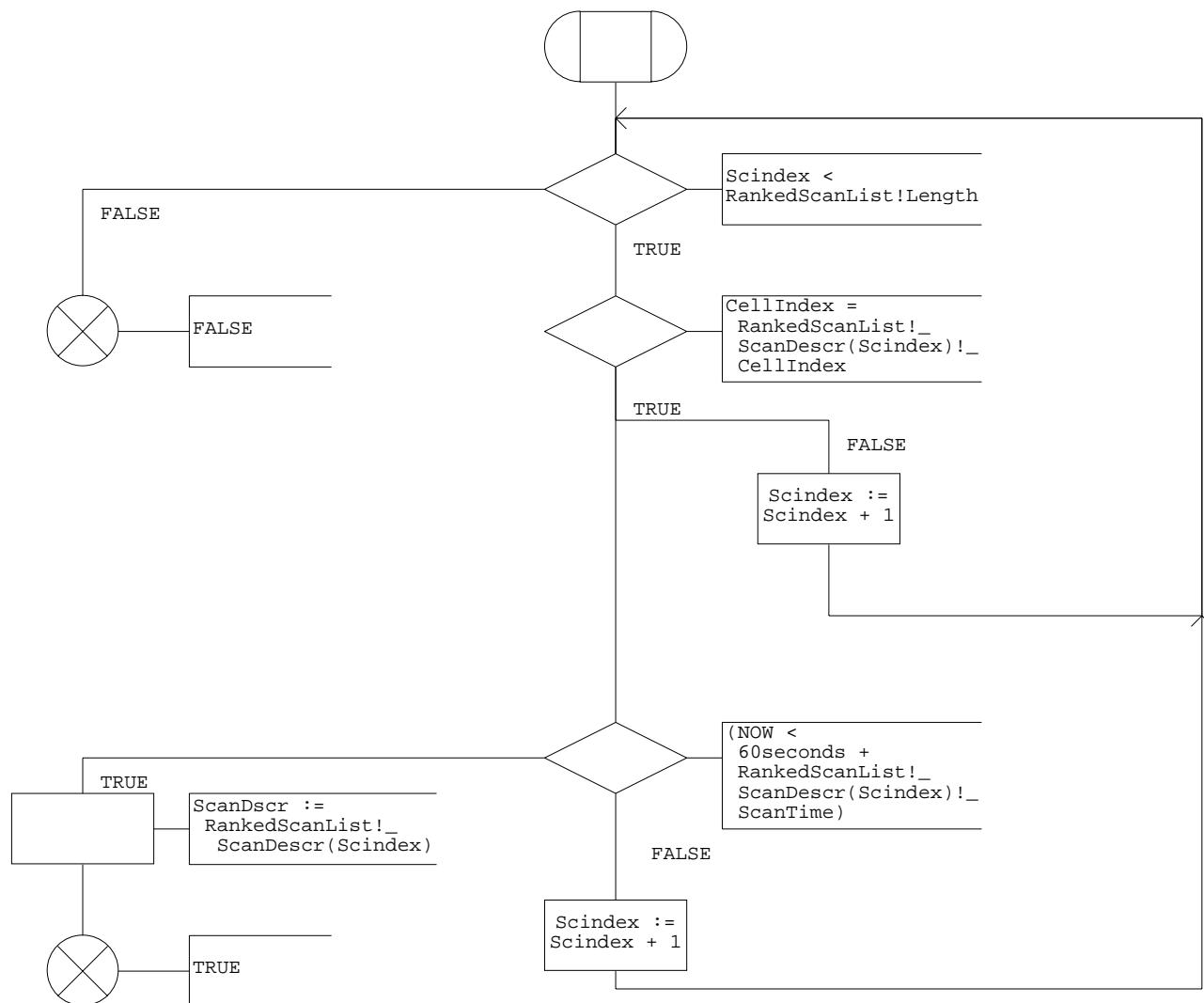
Procedure IsInScanList

```
/* FPAR
   IN cellIndex CellIndexType,
   IN/OUT ScanDscr ScanCellDescrType;
RETURNS Boolean;
```

```
DCL
Scindex Natural := 0;
```

/* Check if the specified cell (cellindex) has been scanned and if this scanning is still valid. If this is so it returns TRUE and initialise the parameter ScanDscr with the information on the scanning.

1(1)



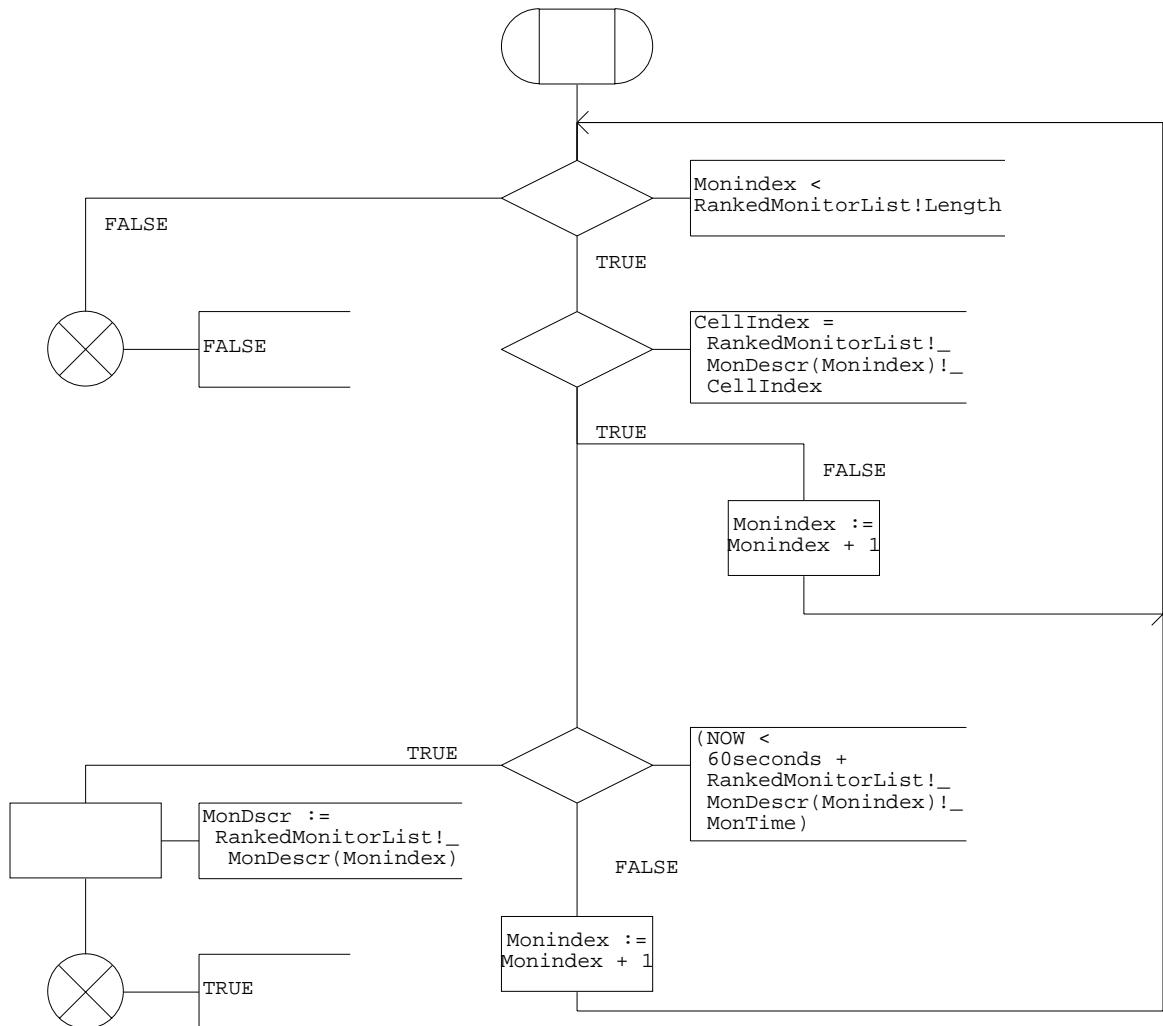
Procedure IsInMonList

1(1)

```
!; FPAR
  IN cellIndex CellIndexType,
  IN/OUT MonDscr MonCellDescrType;
RETURNS Boolean;
```

```
/* Check if the specified cell (cellindex) has been
monitored and if this monitoring is still valid.
If this is so it returns TRUE and initialise the
parameter MonDscr with the information on the
monitoring.
```

```
DCL
Monindex Natural := 0;
```



Procedure InitiateReselection

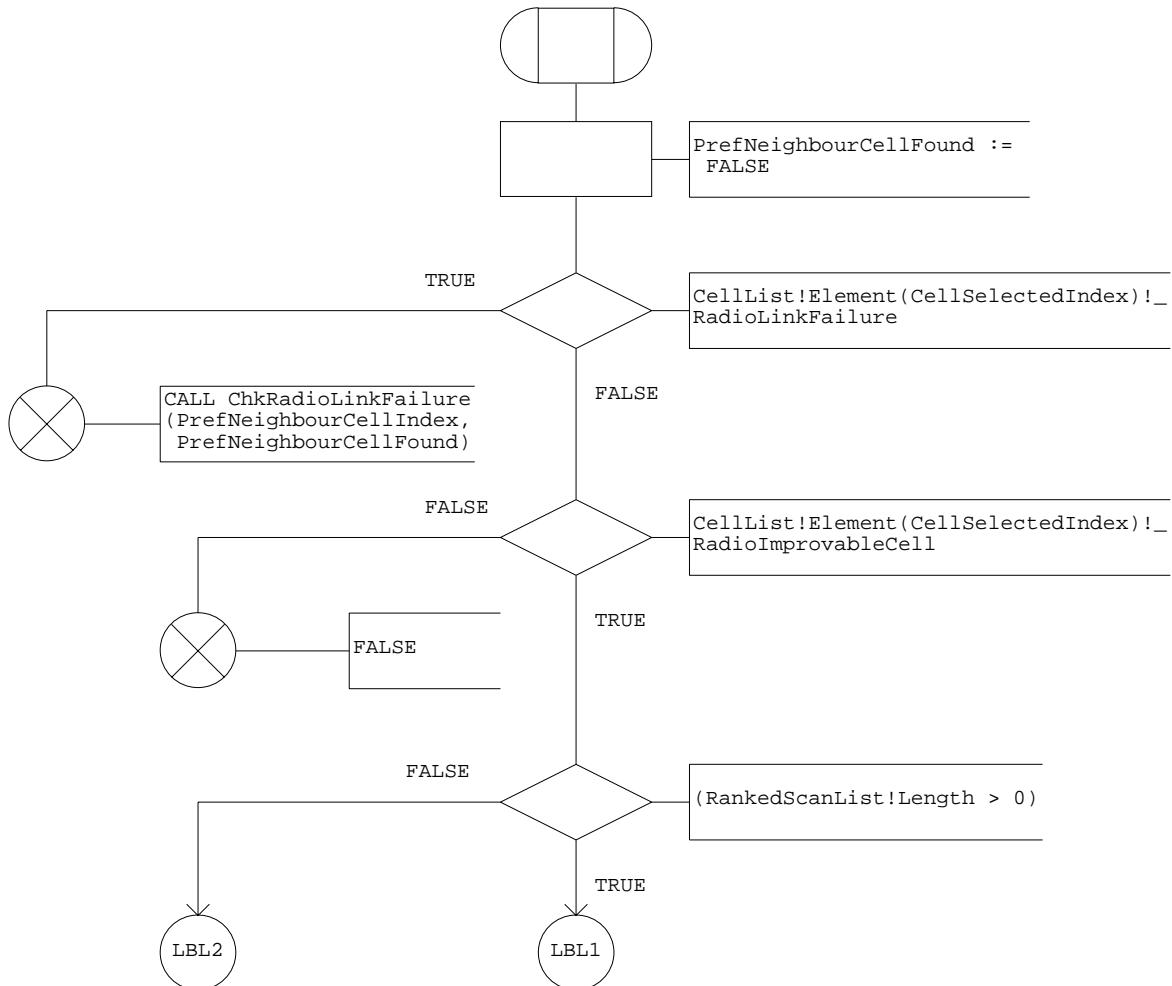
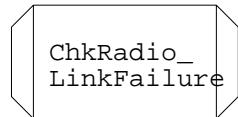
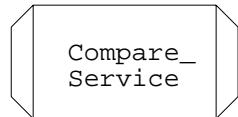
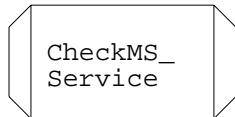
1 (3)

```
; FPAR
  IN/OUT PrefNeighbourCellIndex Natural;
  IN/OUT PrefNeighbourCellFound Boolean;
RETURNS Boolean;
```

/* Return TRUE if cell reselection is to be performed
and then indicates in parameter PrefNeighbourCellIndex
which new cell to select. The check performed is
defined in Clause 18.3.4.5.7.
*/

```
DCL
NbServiceGeq      Boolean,
RankedMonListIndex Natural := 0,
RankedScListIndex  Natural := 0;
```

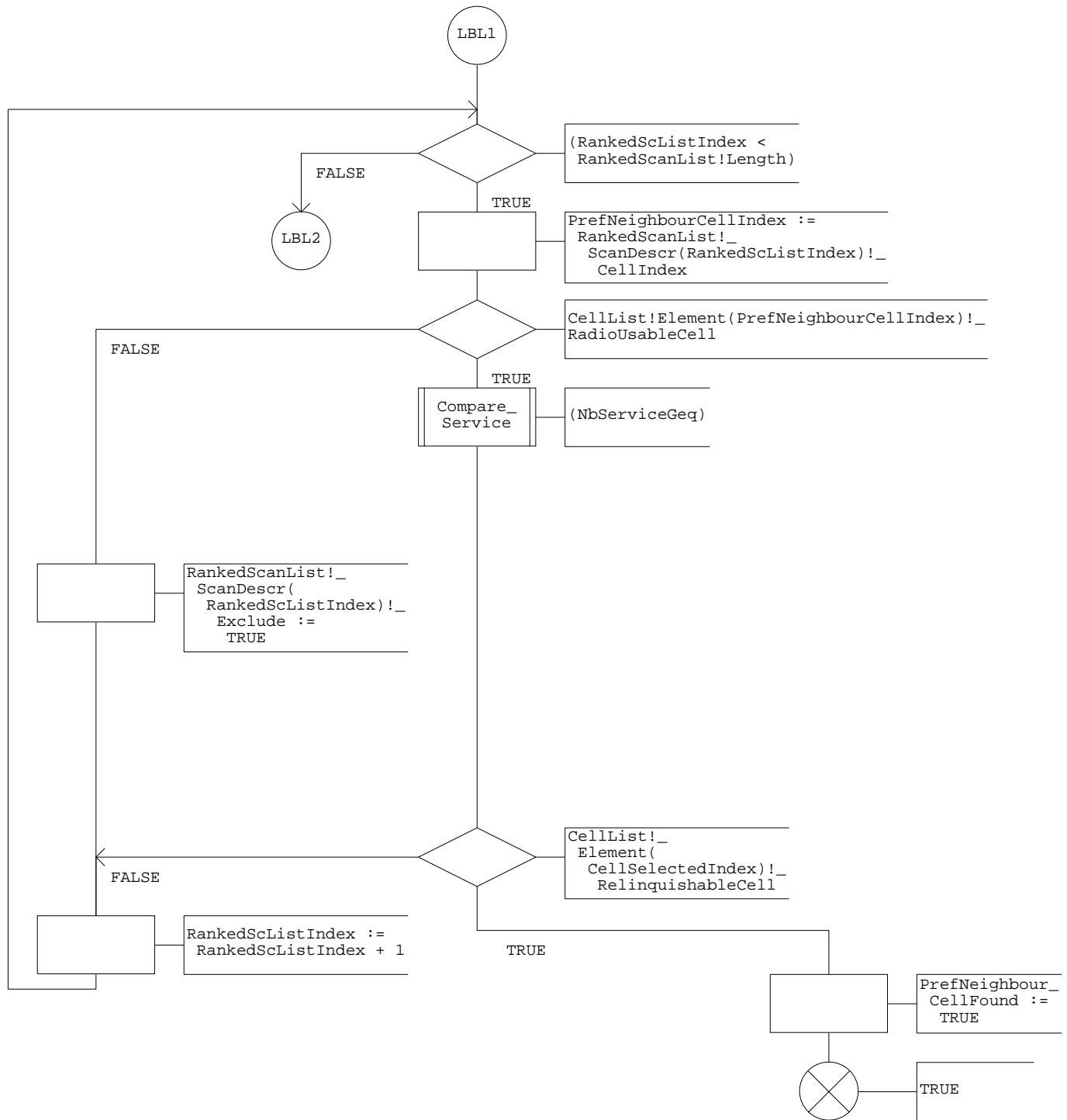
/** LOCAL PROCEDURES **/



Procedure InitiateReselection

2 (3)

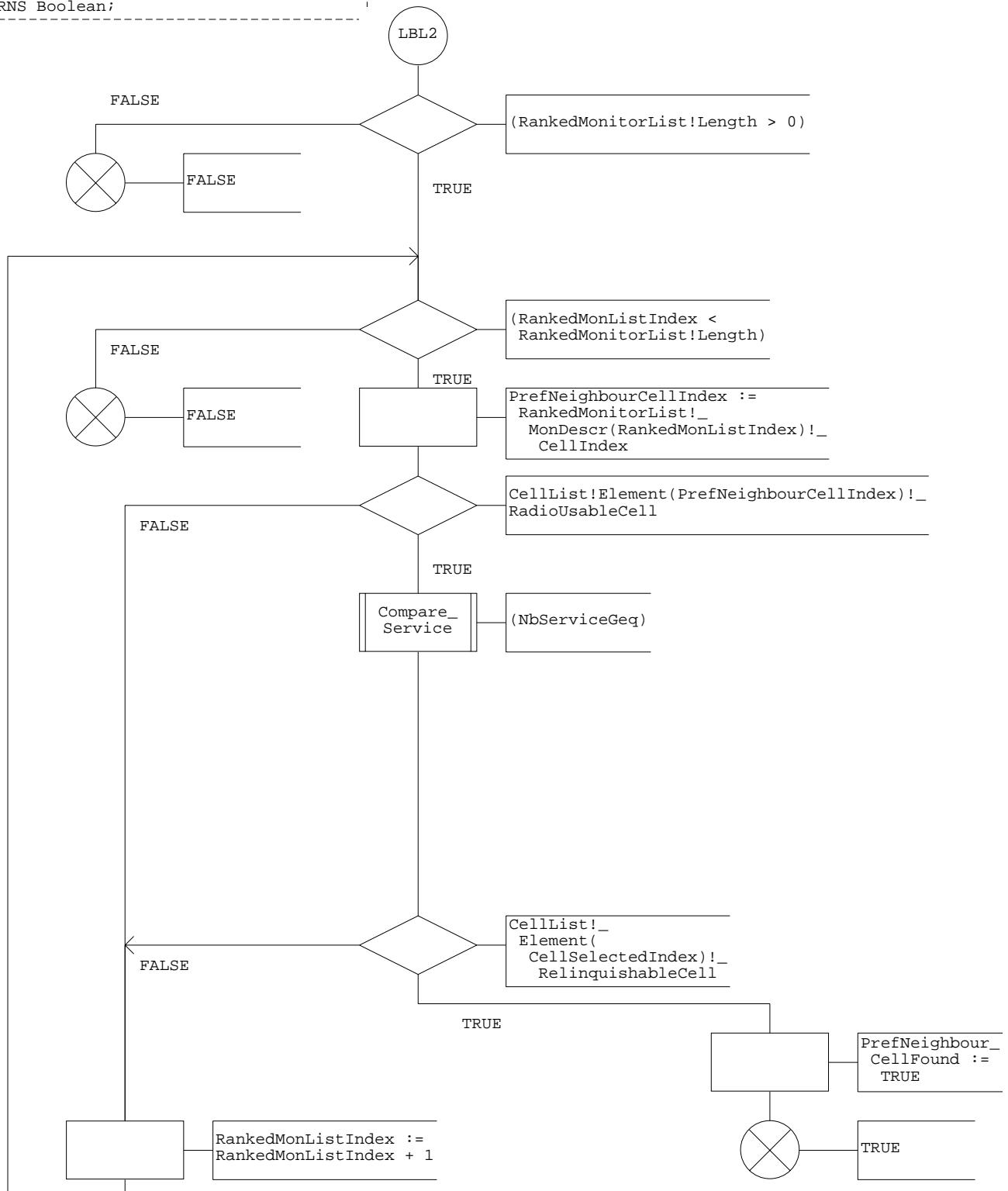
```
; FPAR
  IN/OUT PrefNeighbourCellIndex Natural;
  IN/OUT PrefNeighbourCellFound Boolean;
  RETURNS Boolean;
```



Procedure InitiateReselection

3 (3)

```
; FPAR
  IN/OUT PrefNeighbourCellIndex Natural;
  IN/OUT PrefNeighbourCellFound Boolean;
  RETURNS Boolean;
```

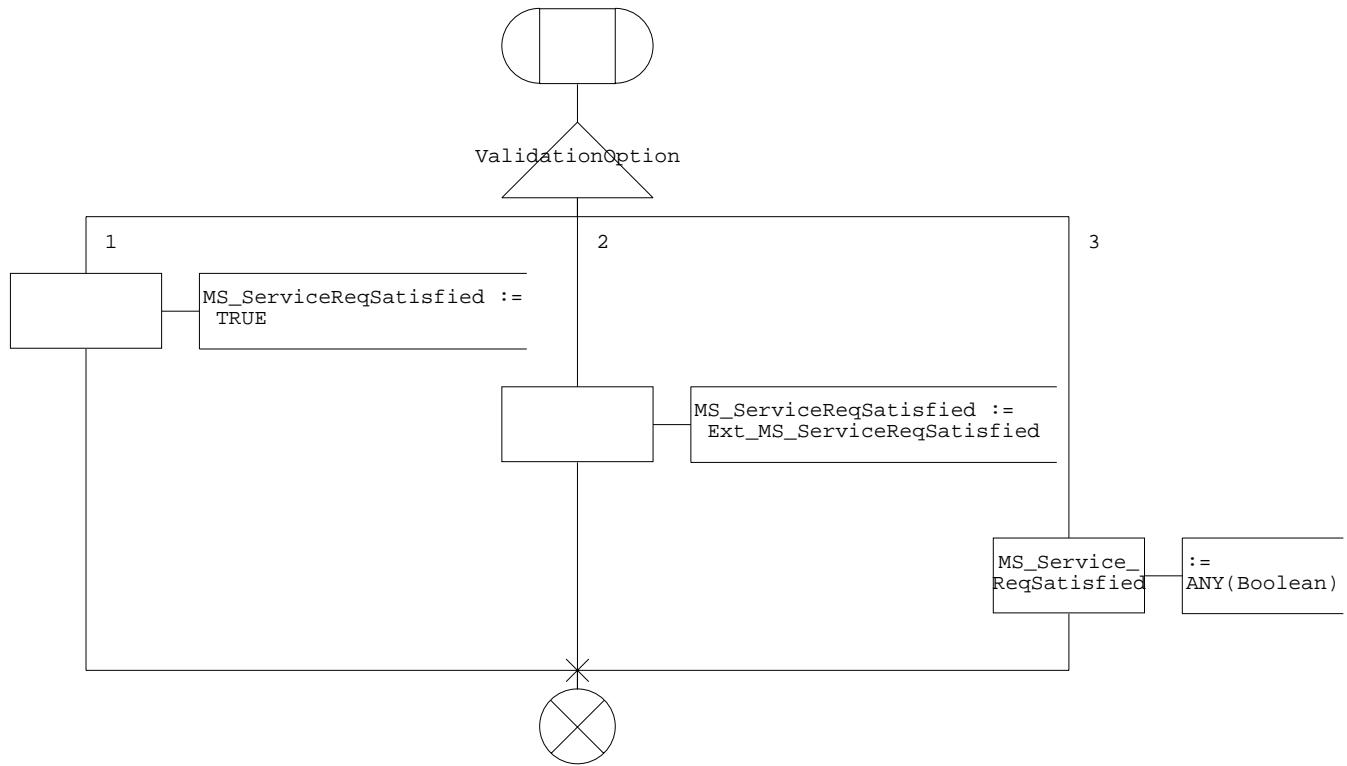


Procedure CheckMSService

1(1)

```
; FPAR
IN/OUT MS_ServiceReqSatisfied Boolean;
```

```
/* Check if the MS service requirements are satisfied by the preferred neighbour cell when a radio link failure has occurred Clause 18.3.4.5.7. This criterion is dependent on the MS and in this model is controlled through the use of options.
```

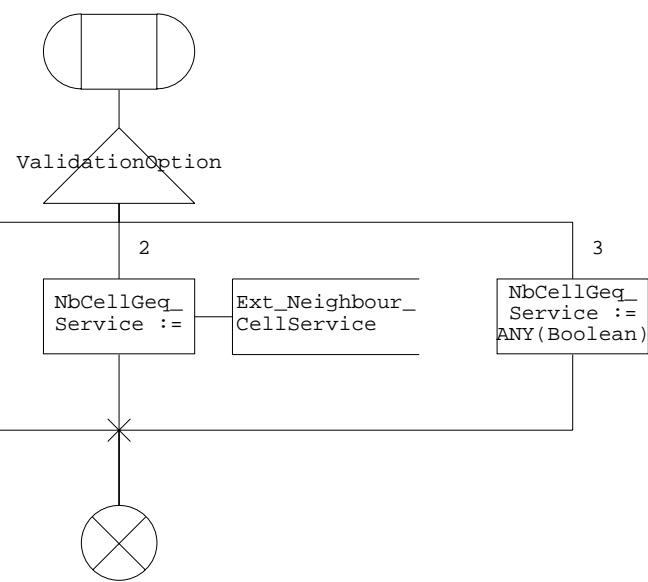


Procedure CompareService

1(1)

```
/* FPAR
   IN/OUT NbCellGeqService Boolean */
```

```
/* Compare the service offered by the
   serving cell and the preferred
   neighbour cell according to Clause
   18.3.4.5.7 and if the neighbour
   cell offers better or equal service
   the NbCellGeqService is set to True.
*/
```



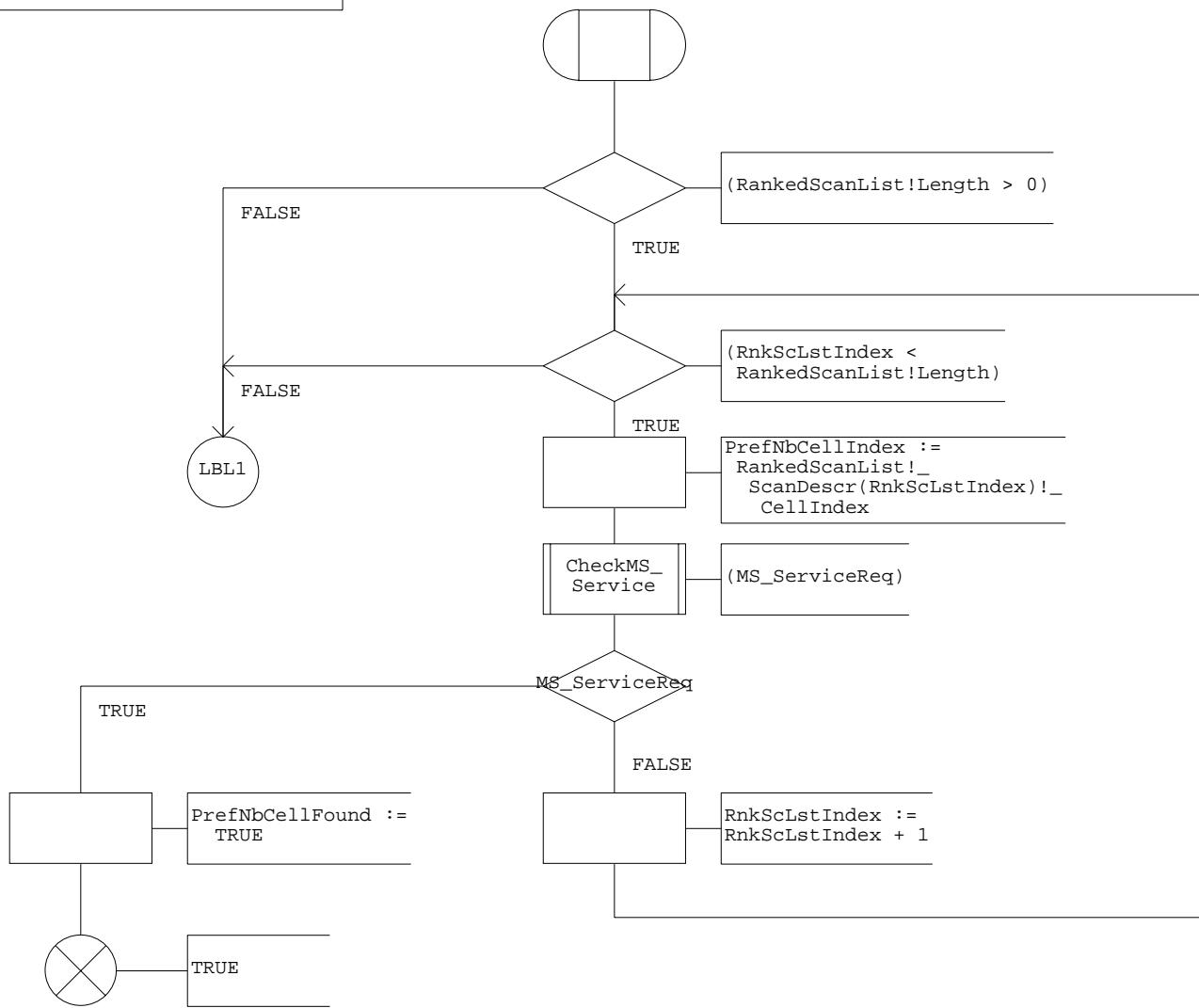
Procedure ChkRadioLinkFailure

1 (2)

```
!; FPAR
  IN/OUT PrefNbCellIndex Natural;
  IN/OUT PrefNbCellFound Boolean;
  RETURNS Boolean;
```

```
/* In the cell reselection check this function
  checks if the criteria "radio link failure"
  is satisfied and hence a cell reselection
  shall be initiated. If so TRUE is returned.
```

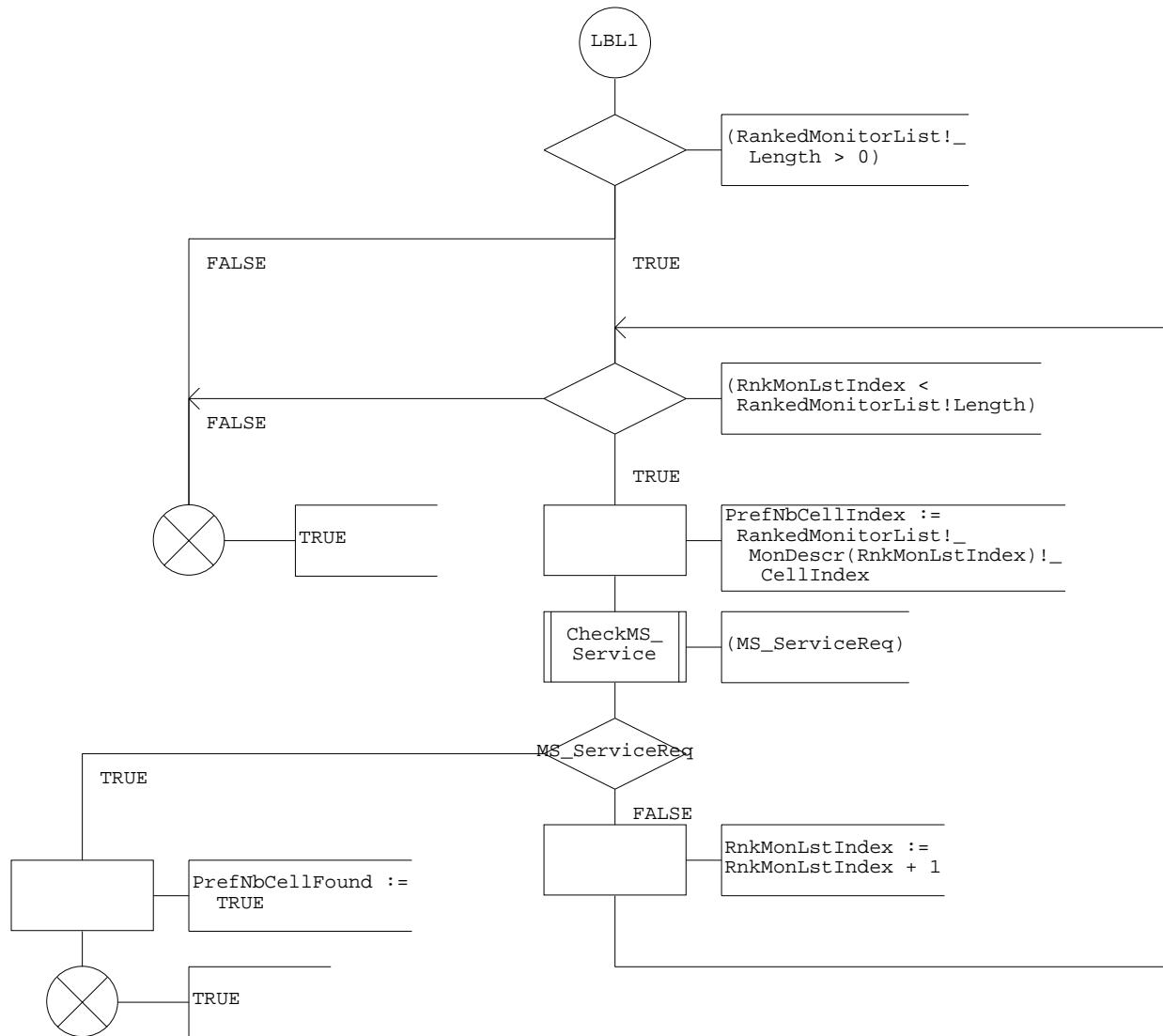
```
DCL
  MS_ServiceReq Boolean,
  RnkMonLstIndex Natural := 0,
  RnkScLstIndex Natural := 0;
```



Procedure ChkRadioLinkFailure

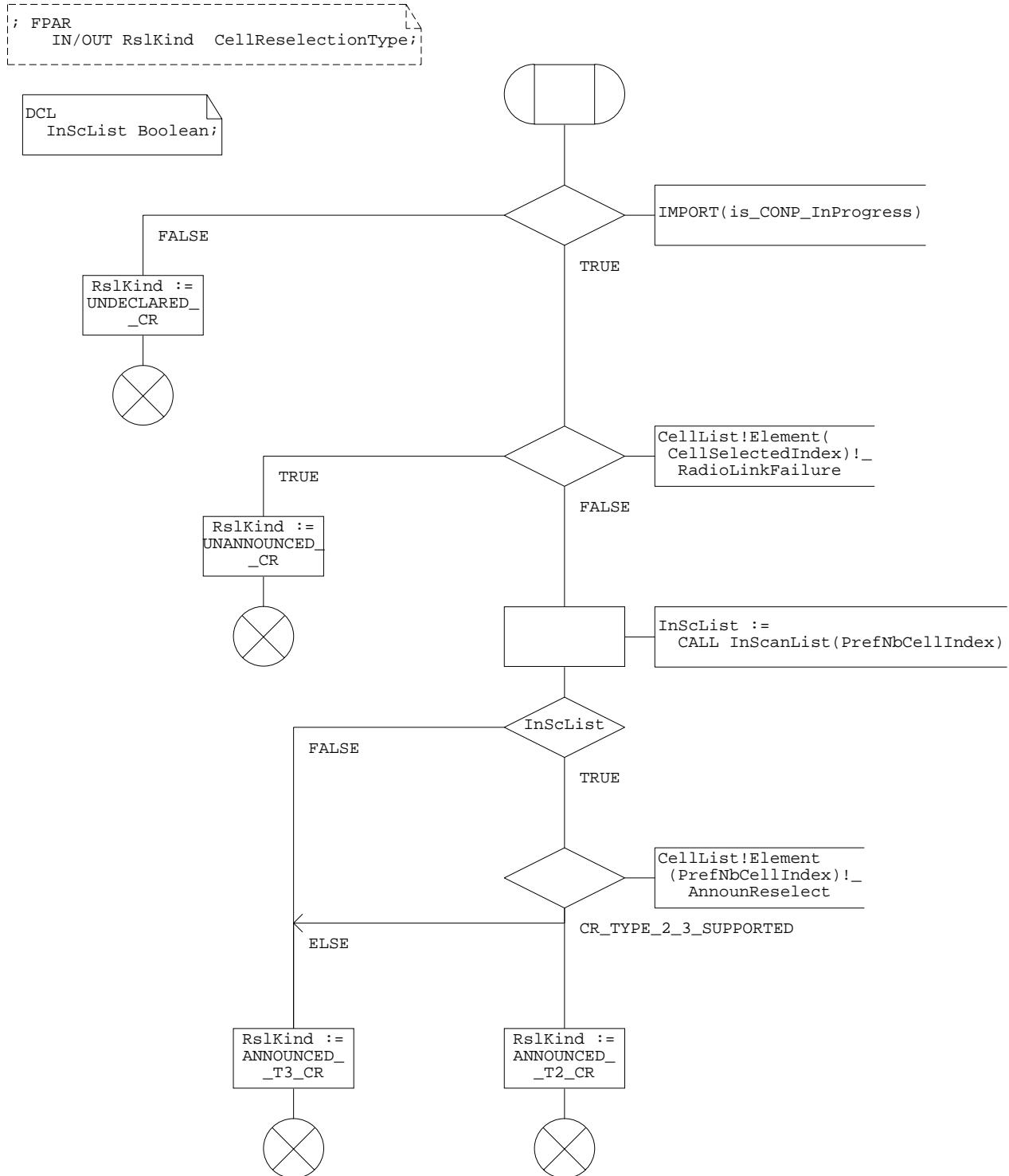
2 (2)

```
!---- FPAR
  IN/OUT PrefNbCellIndex Natural;
  IN/OUT PrefNbCellFound Boolean;
RETURNS Boolean;
```



Procedure SelectReselectionKind

1(1)



Procedure InitialCellSelect

1(7)

```
; FPAR
  IN/OUT ActivateReq MLE_ActivateReqType;
  IN/OUT CR_Status Boolean;
```

```
/* This procedure implements a simple initial cell selection.
   It selects the first cell that satisfies the cell selection
   criterion.
*/
```

```
/** LOCAL VARIABLES **/
```

```
DCL
  ActivateInd_Sent Boolean := FALSE,
  ChnIndex Natural := 0,
  MMC_MNC_checked Boolean := FALSE,
  RegInfoReady Boolean := FALSE,
  RegRequired Boolean,
  ScDescr ScanCellDescrType;
```

```
/** MLE PDU DATA VARIABLES **/
```

```
DCL
  SIN1PDU MLE_D_SIN1Type,
  SIN2PDU MLE_D_SIN2Type,
  ActConf MLE_ActivateConType;
```

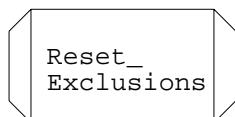
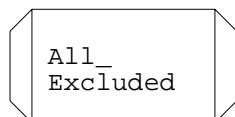
```
/** MLE SP DATA VARIABLES **/
```

```
DCL
  UpdateReq MLE_UpdateReqType;
```

```
/** LLC SP DATA VARIABLES **/
```

```
DCL
  SelReq TLC_SelectRequestType,
  SelConf TLC_SelectConfirmType;
```

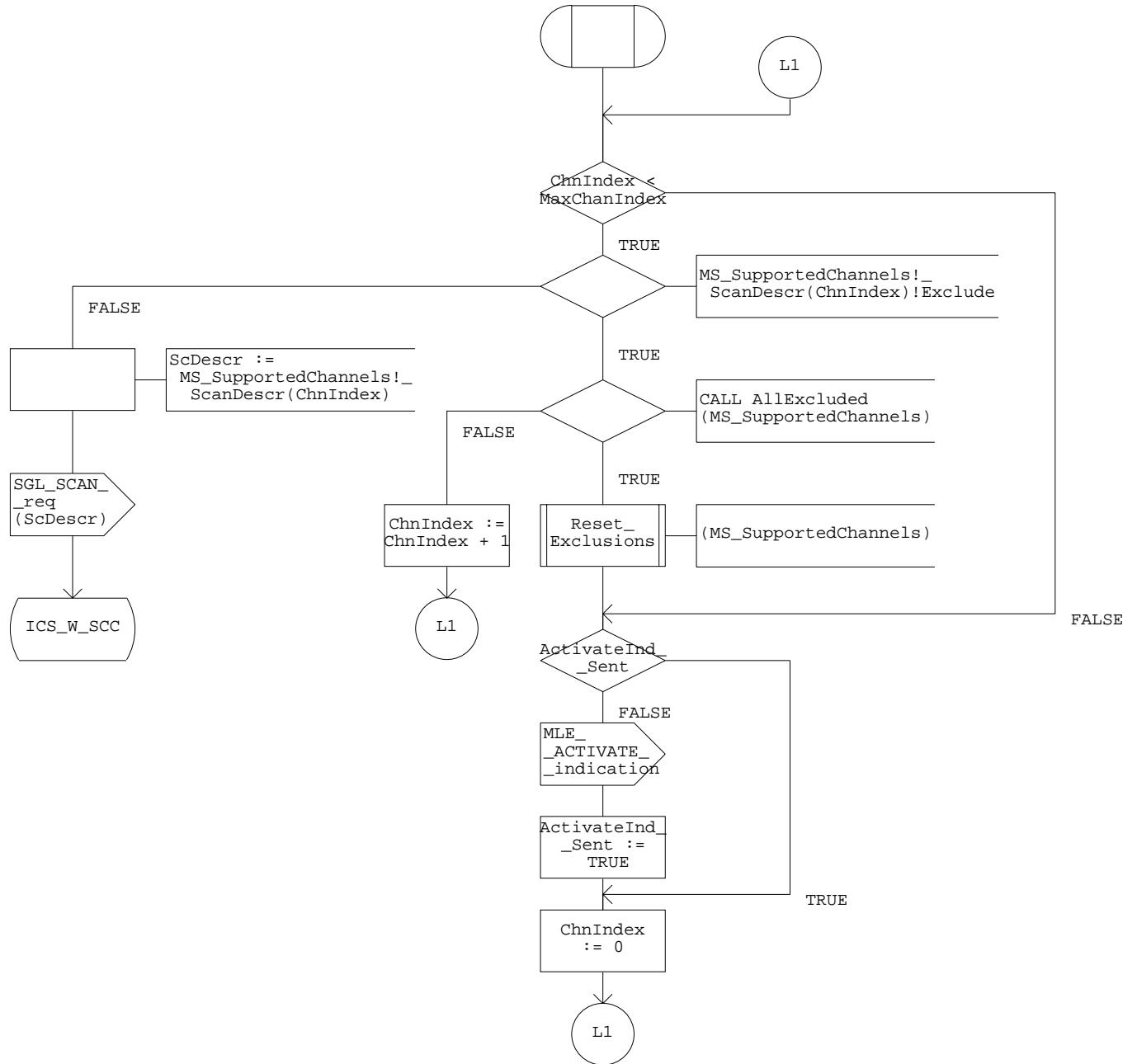
```
/** PROCEDURE REFERENCES **/
```



Procedure InitialCellSelect

2(7)

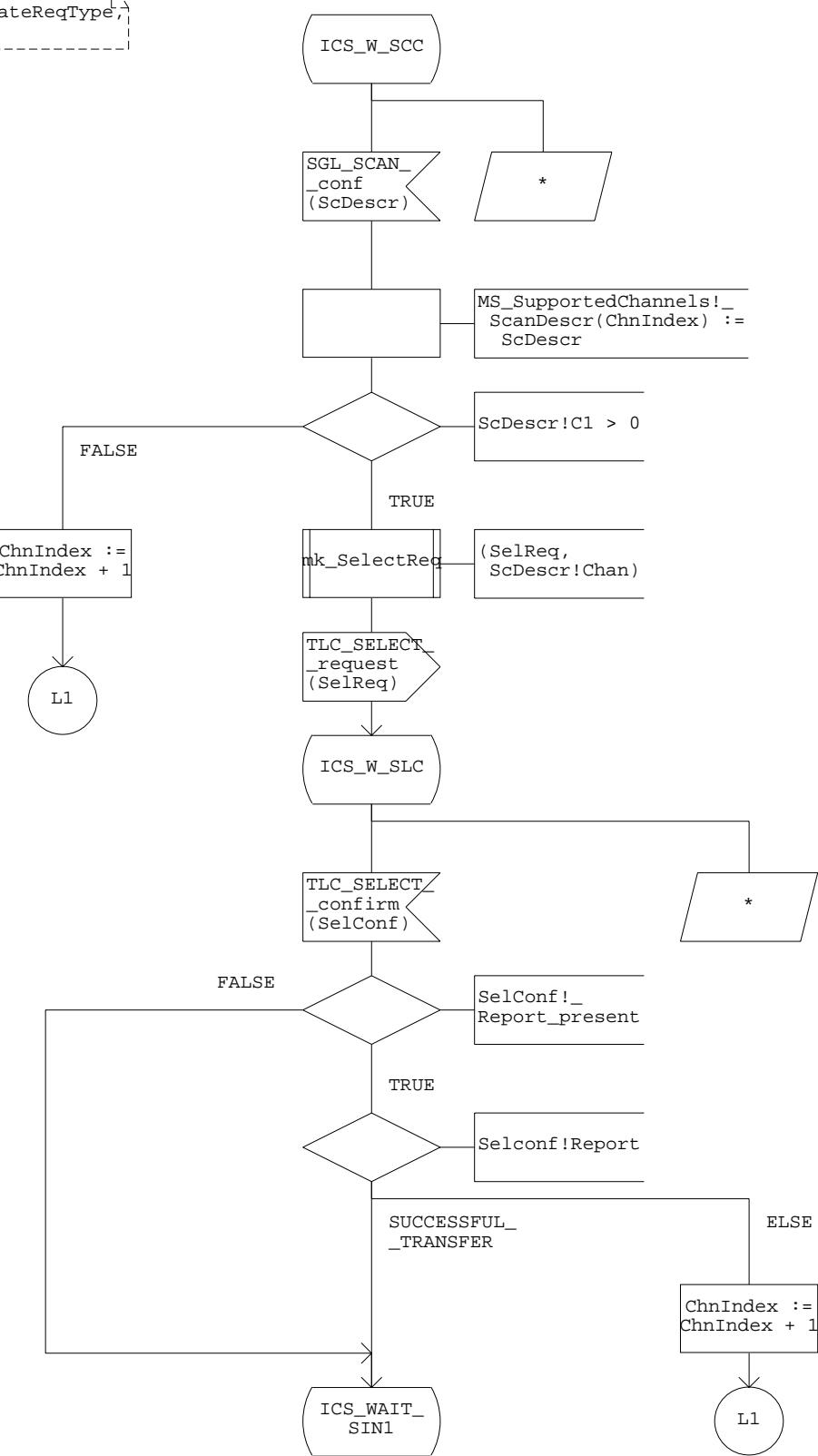
```
; FPAR
IN/OUT ActivateReq MLE_ActivateReqType;
IN/OUT CR_Status Boolean;
```



Procedure InitialCellSelect

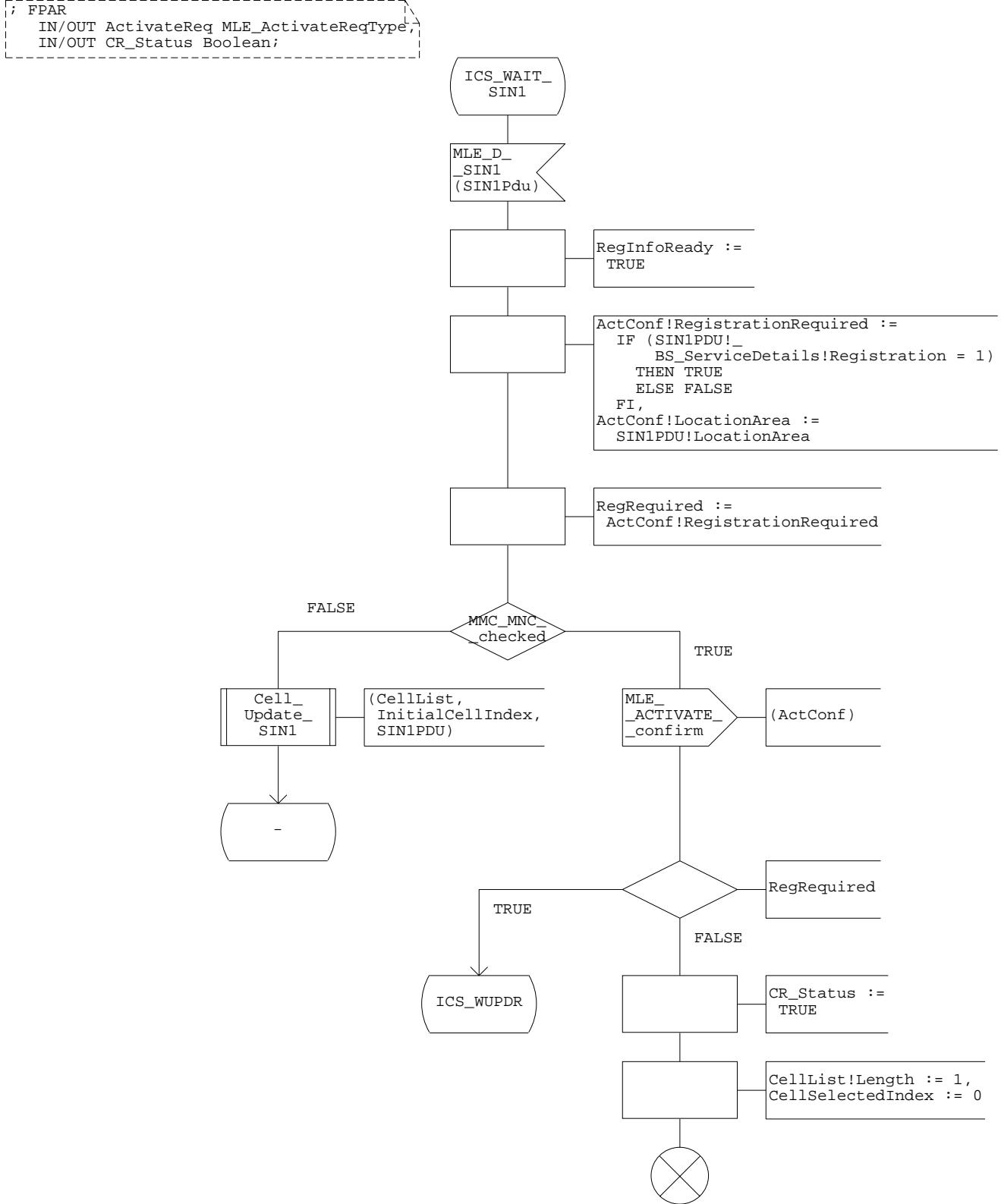
3(7)

```
; FPAR
IN/OUT ActivateReq MLE_ActivateReqType;
IN/OUT CR_Status Boolean;
```



Procedure InitialCellSelect

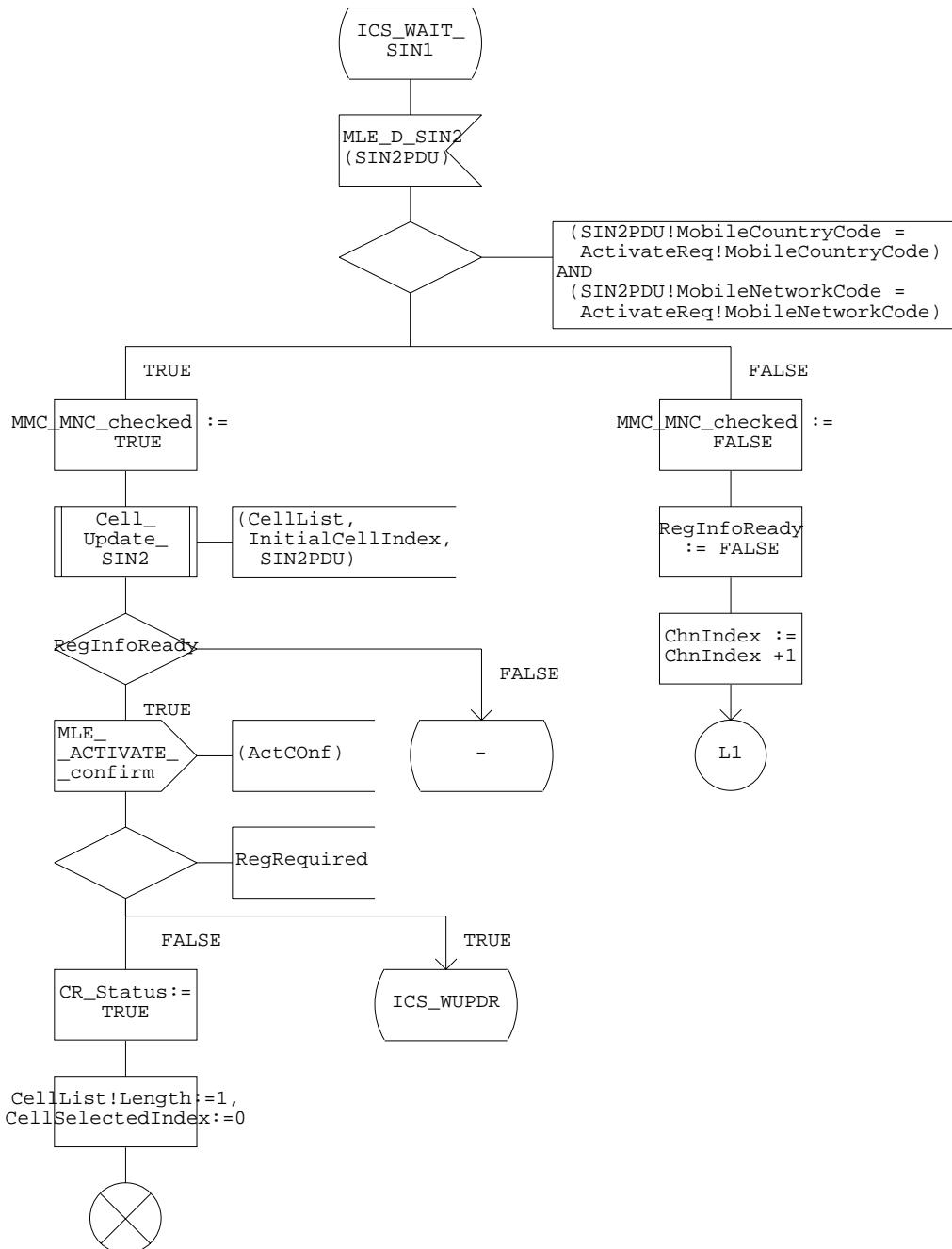
4(7)



Procedure InitialCellSelect

5(7)

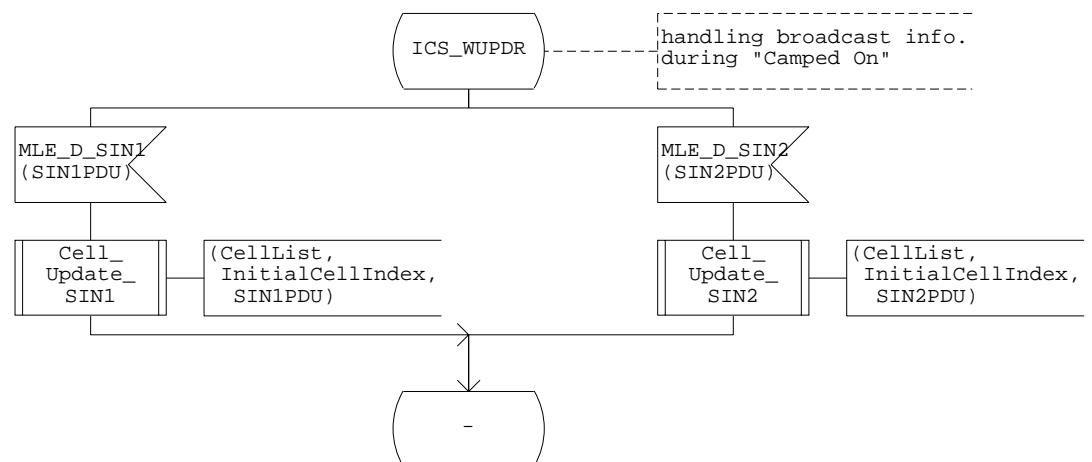
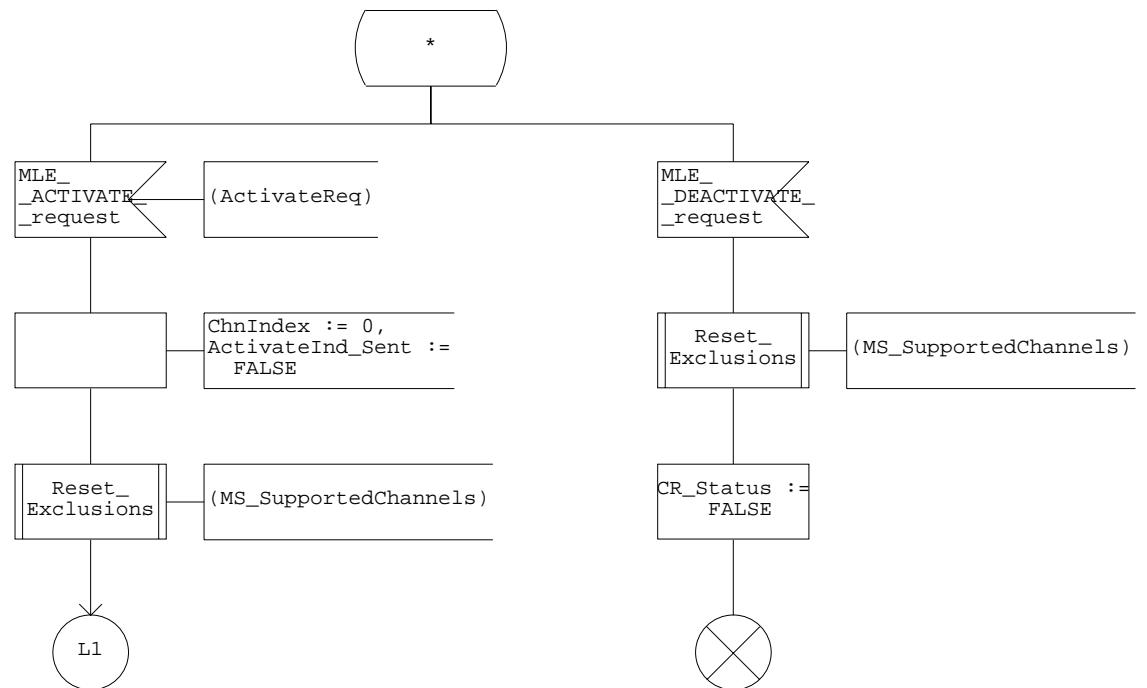
```
; FPAR
IN/OUT ActivateReq MLE_ActivateReqType;
IN/OUT CR_Status Boolean;
```



Procedure InitialCellSelect

6(7)

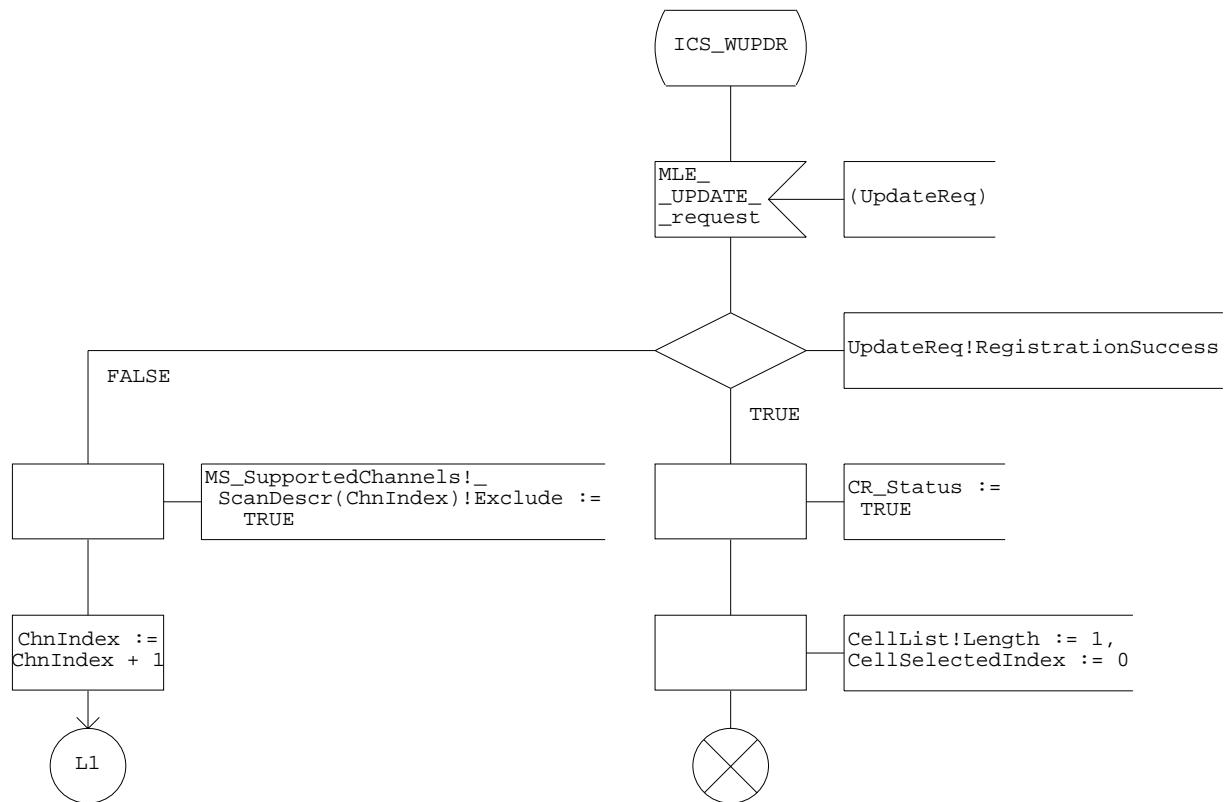
```
; FPAR
IN/OUT ActivateReq MLE_ActivateReqType;
IN/OUT CR_Status Boolean;
```



Procedure InitialCellSelect

7 (7)

```
; FPAR
IN/OUT ActivateReq MLE_ActivateReqType;
IN/OUT CR_Status Boolean;
```



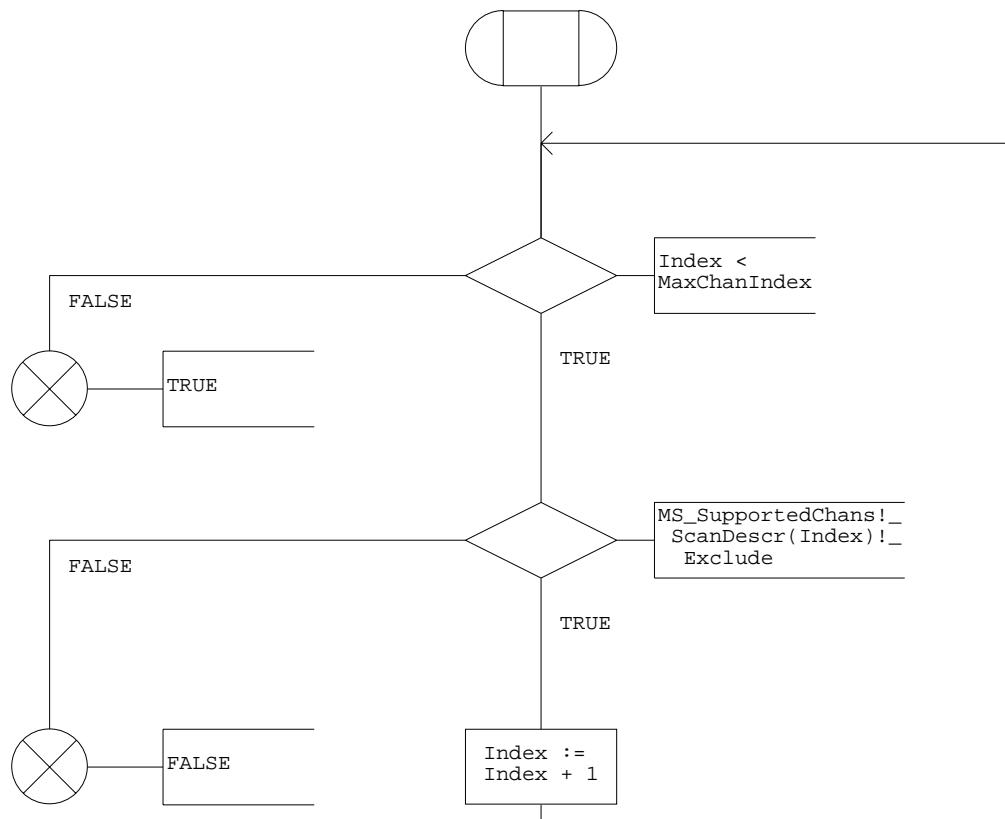
Procedure AllExcluded

1(1)

```
; FPAR
  IN MS_SupportedChans ScanListType;
  RETURNS Boolean;
```

```
/* Check if all channels in the initial scan
list has been excluded and if so return TRUE
otherwise FALSE
*/
```

```
DCL
Index Natural := 0;
```



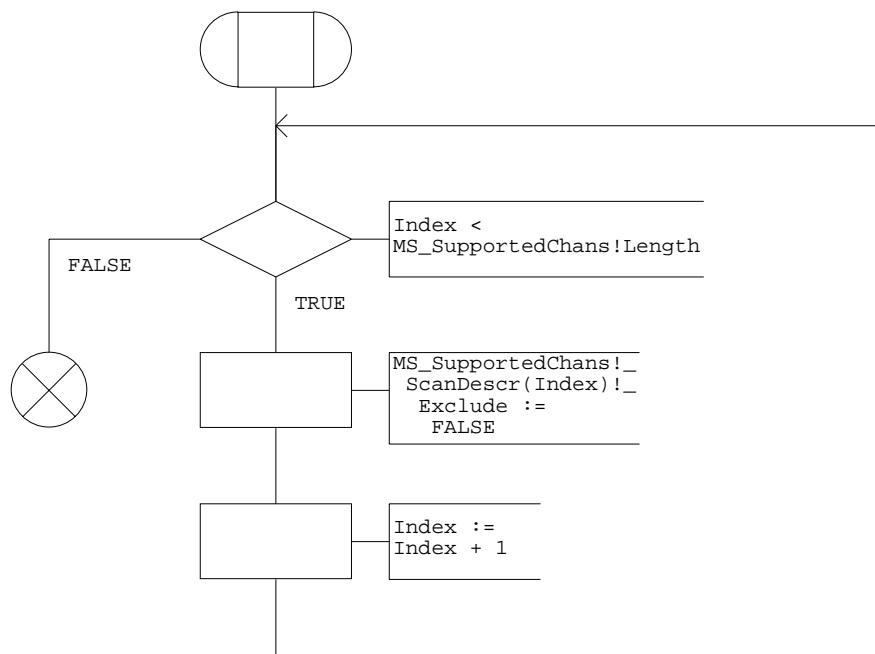
Procedure ResetExclusions

1(1)

```
; FPAR  
IN/OUT MS_SupportedChans ScanListType;
```

```
/* Make all predefined channels available for  
scanning again, i.e. reset the exclusion  
flag for all channels  
*/
```

```
DCL  
Index Natural := 0;
```



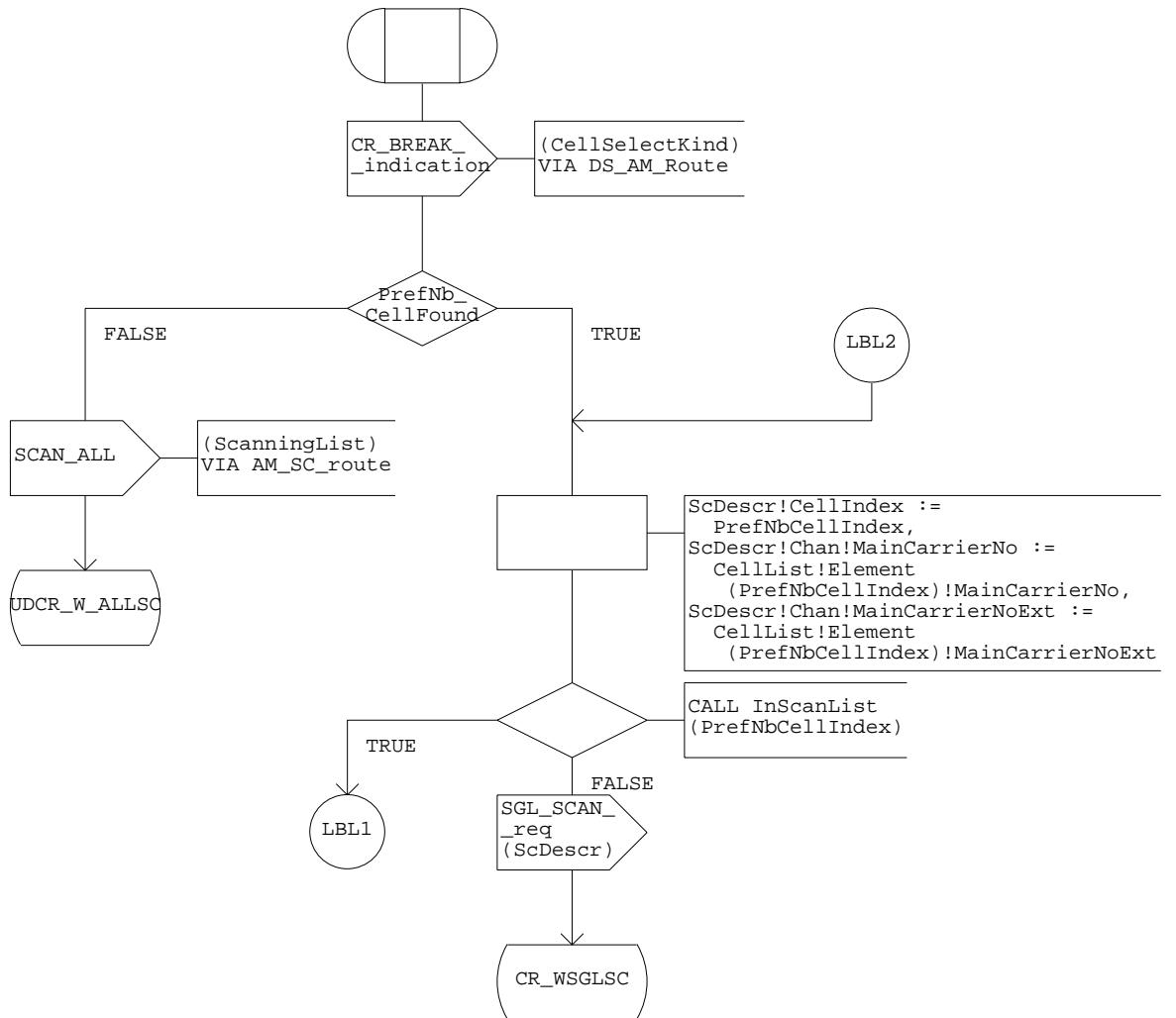
Procedure CR_Undecl

1(3)

```
; FPAR
  IN/OUT CR_Status Boolean;
```

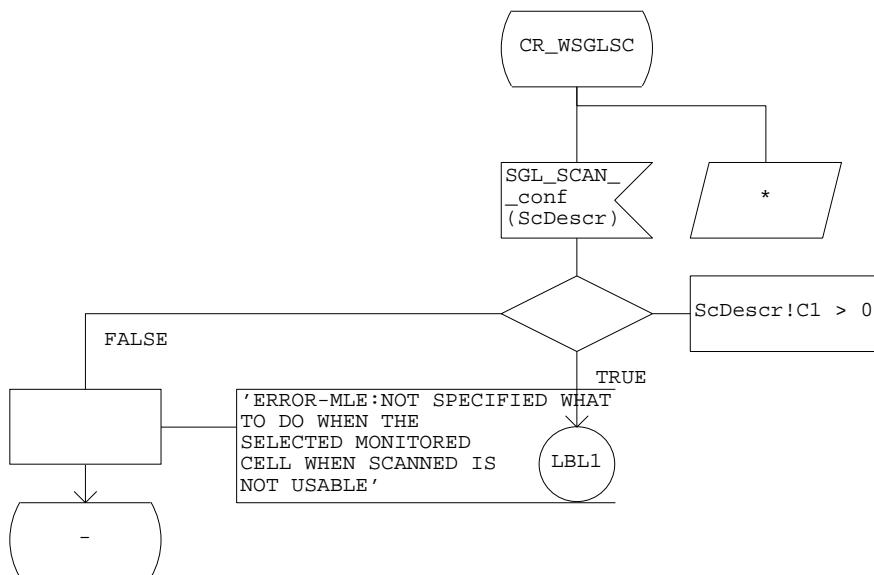
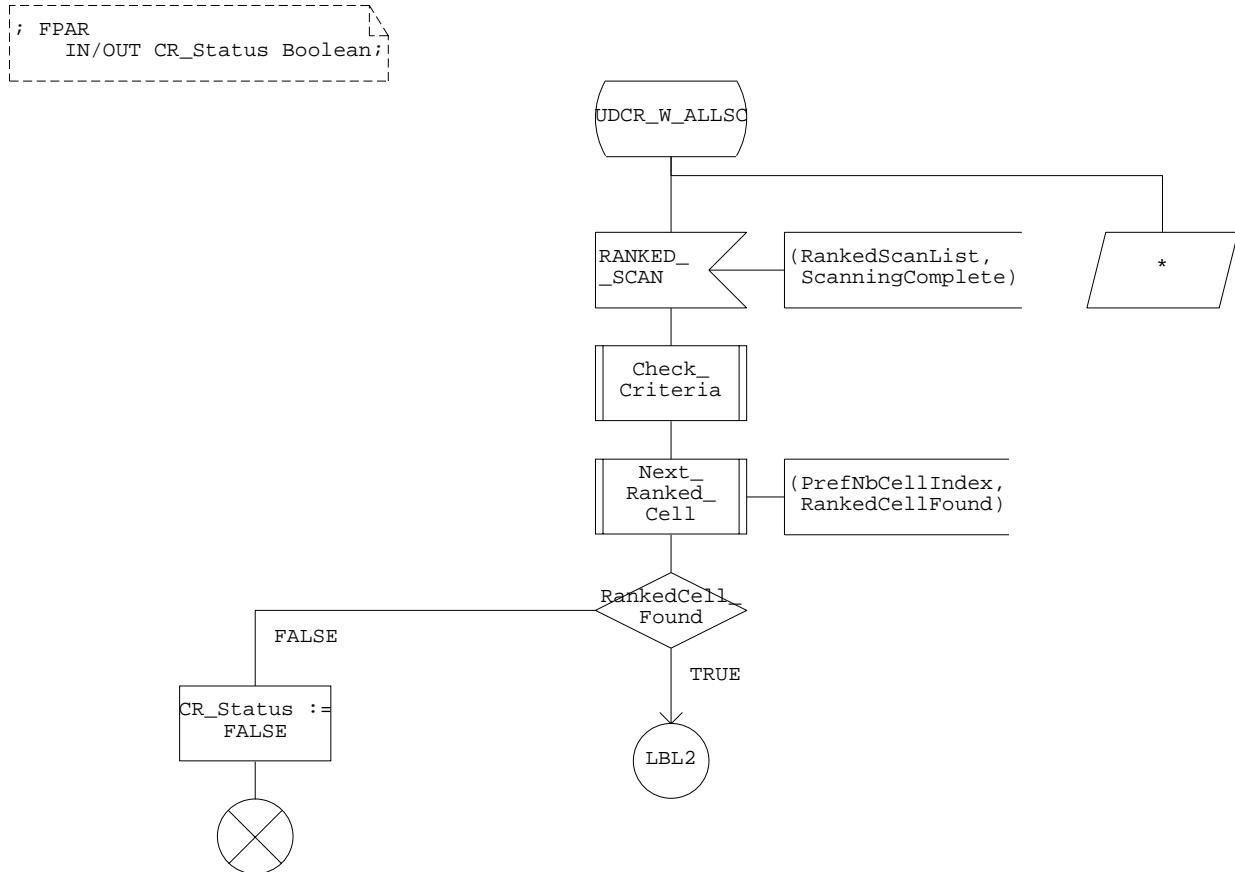
```
/* Perform the undeclared cell reselection procedure
and set parameter CR_Status TRUE if the
reselection was successful
*/
```

```
DCL
  RankedCellFound Boolean,
  ScDescr ScanCellDescrType,
  SelReq TLC_SelectRequestType;
```



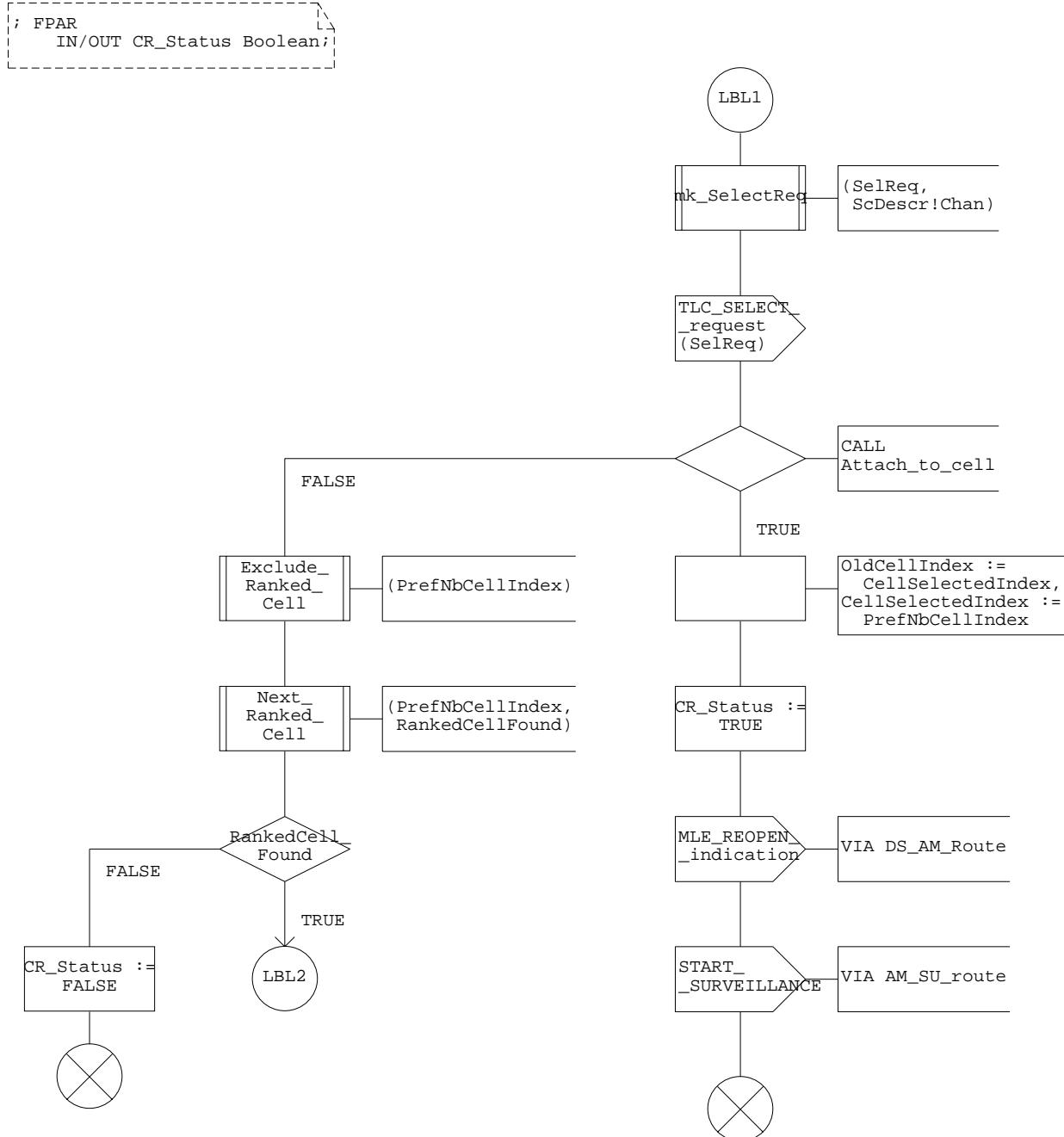
Procedure CR_Undecl

2(3)



Procedure CR_Undecl

3 (3)



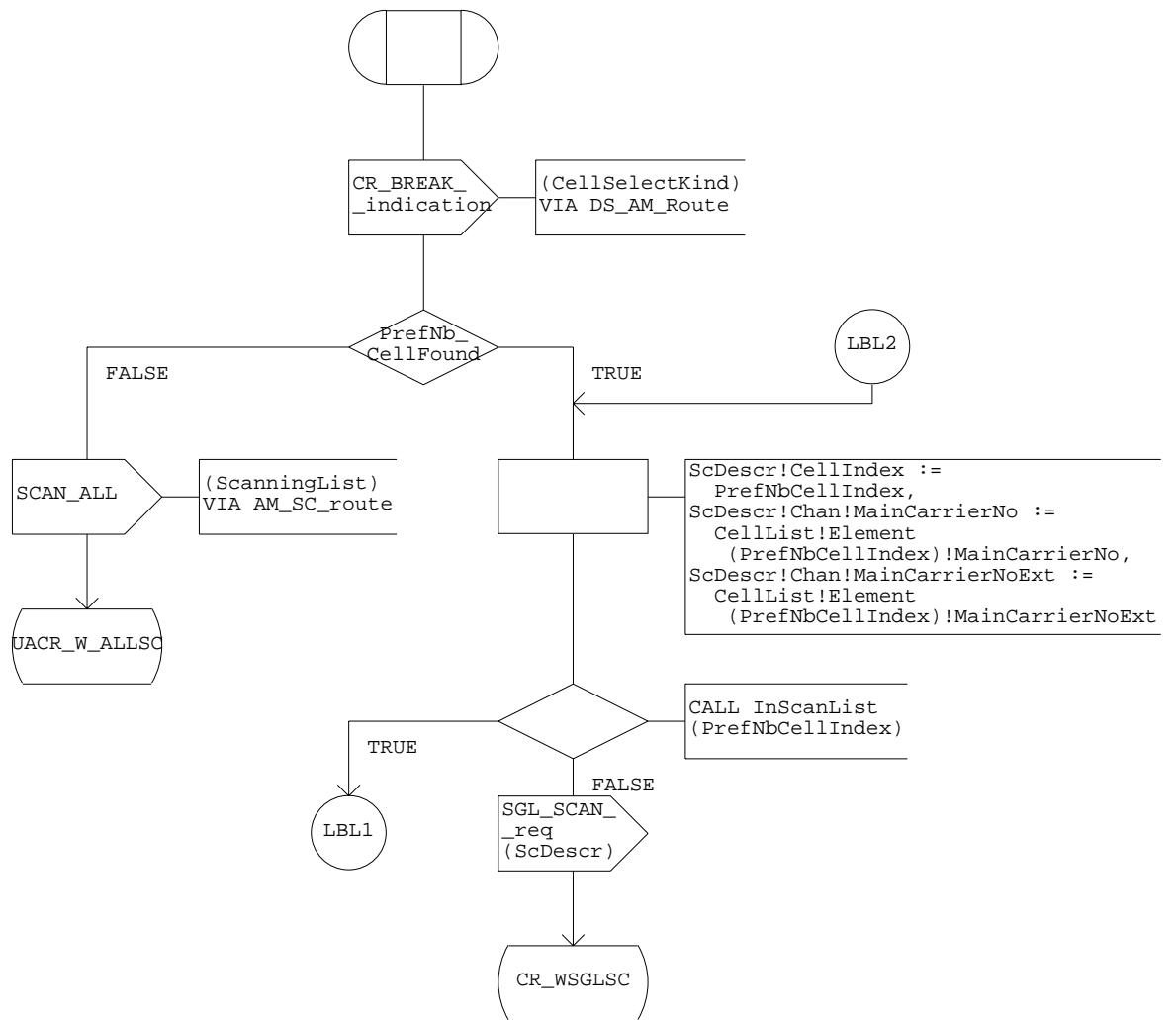
Procedure CR_Unanoun

1 (3)

```
; FPAR
  IN/OUT CR_Status Boolean;
```

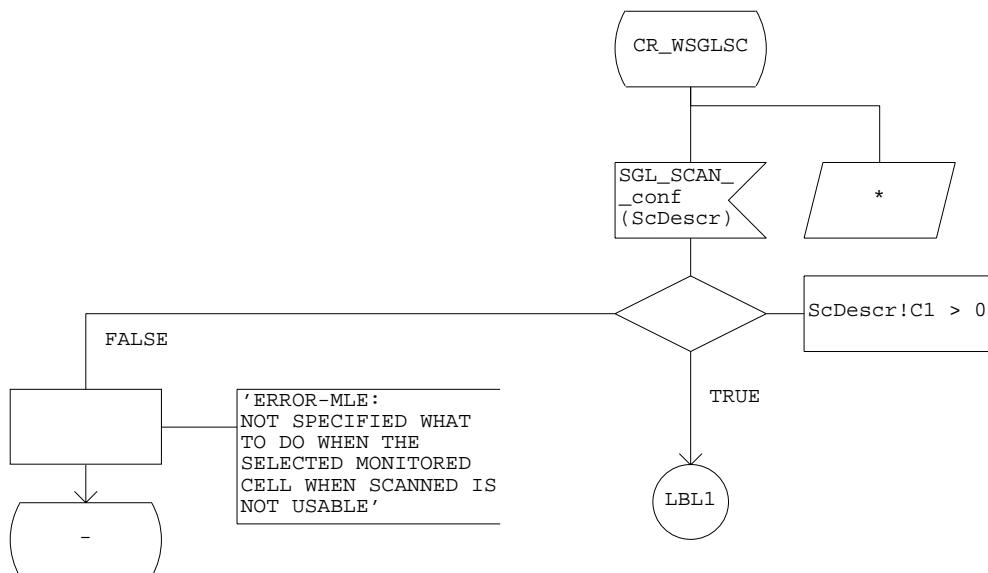
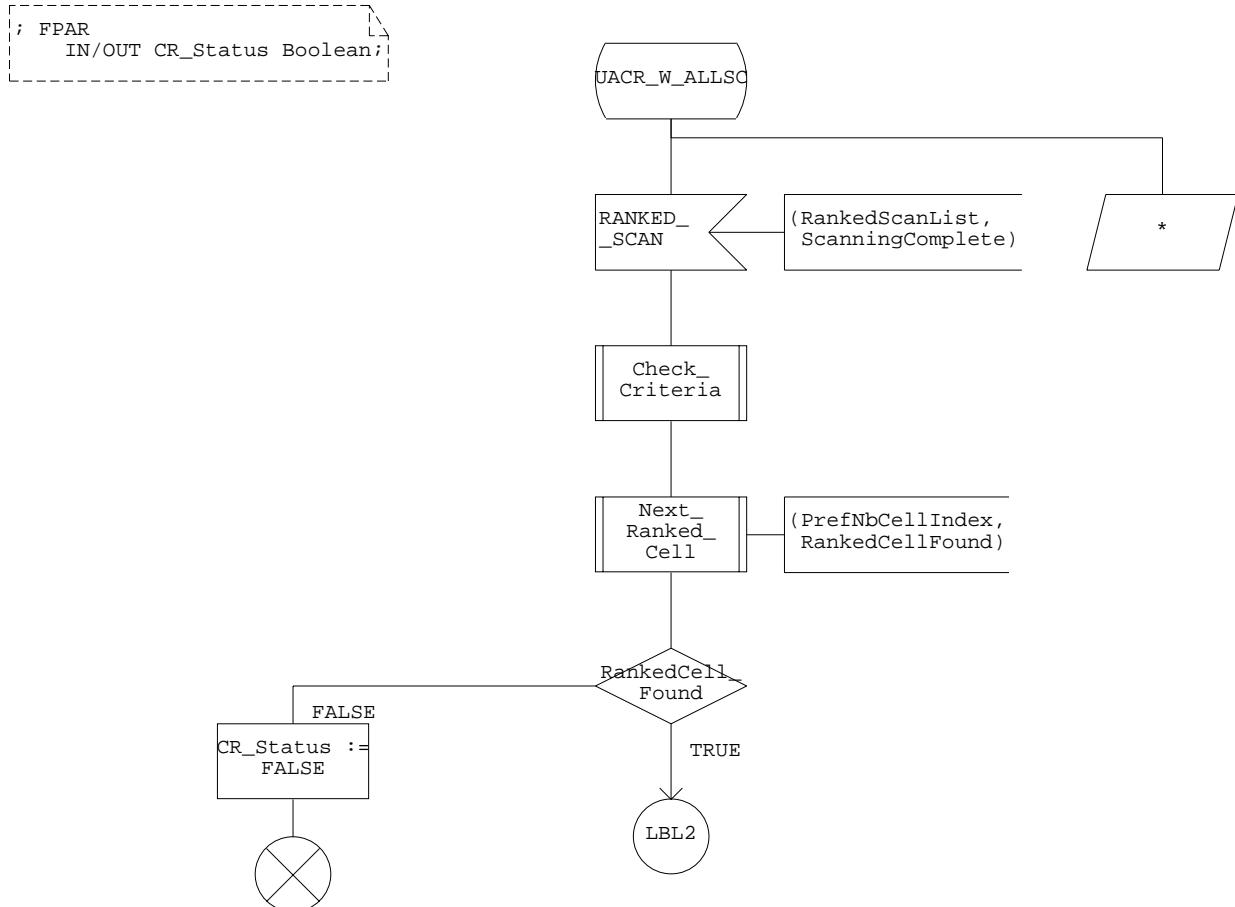
/* Perform unannounced cell reselection procedure
and set the parameter CR_Status TRUE if the
reselection was successful.
*/

```
DCL
  RankedCellFound Boolean,
  ScDescr ScanCellDescrType,
  NewCellInfo NewCellInfoType,
  SelReq TLC_SelectRequestType;
```



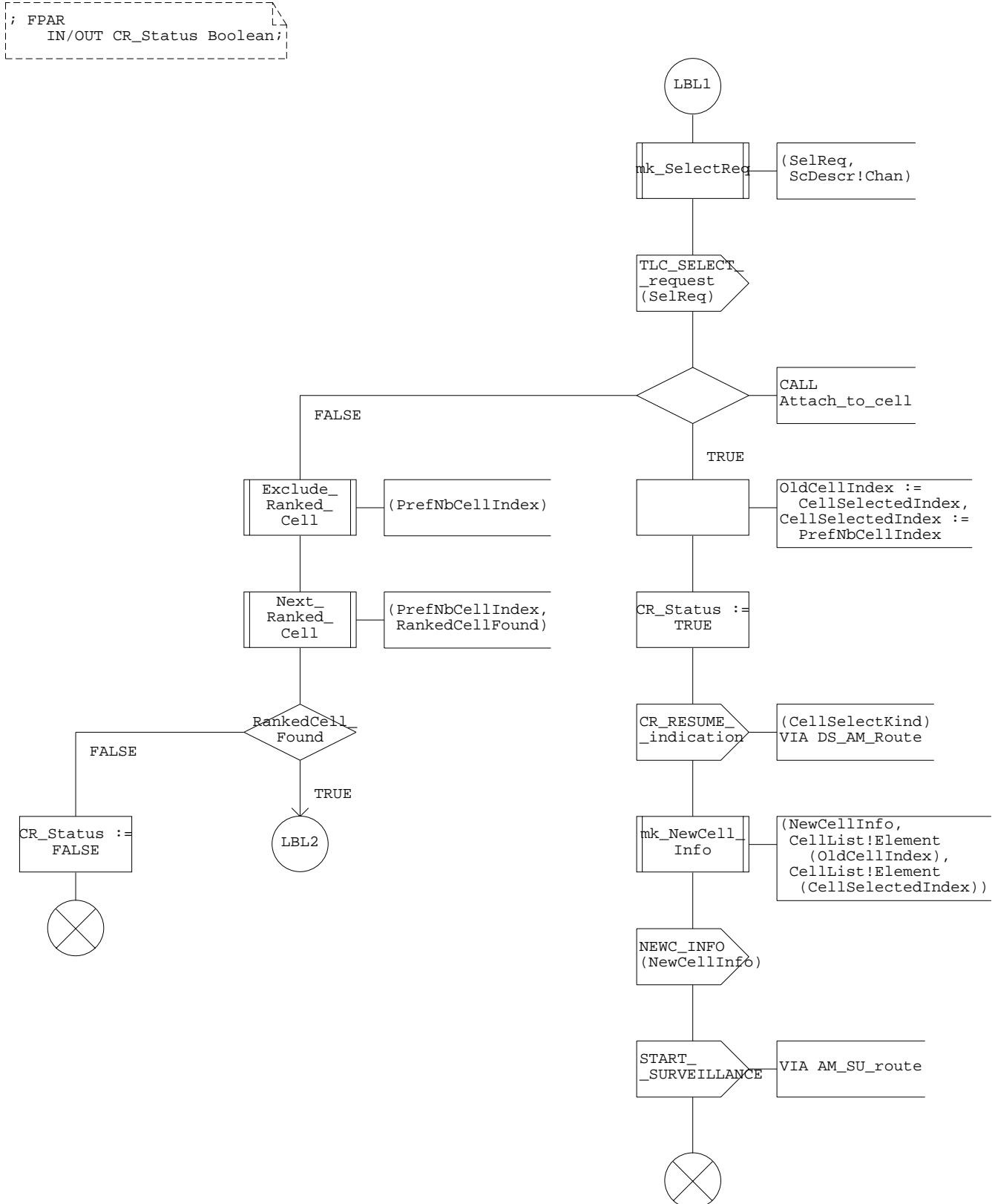
Procedure CR_Unanoun

2(3)



Procedure CR_Unanoun

3 (3)



Procedure CR_Announ_T3

1 (4)

```
; FPAR
  IN/OUT CR_Status Boolean;
```

```
/* Perform announced cell reselection type 3
   and set the parameter CR_Status TRUE if the
   reselection was successful.
*/
```

```
/** LOCAL VARIABLES **/
```

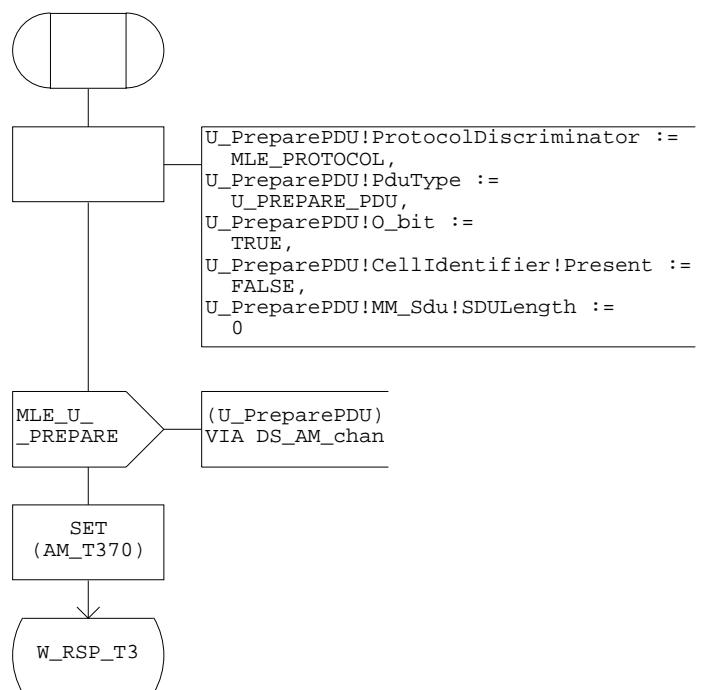
```
DCL
  NewCellInfo      NewCellInfoType,
  RankedCellFound Boolean,
  ScDescr          ScanCellDescrType;
```

```
/** MLE SP DATA VARIABLES **/
```

```
DCL
  U_PrepPDU MLE_U_PrepType,
  NewCellPDU MLE_D_NewCellType;
```

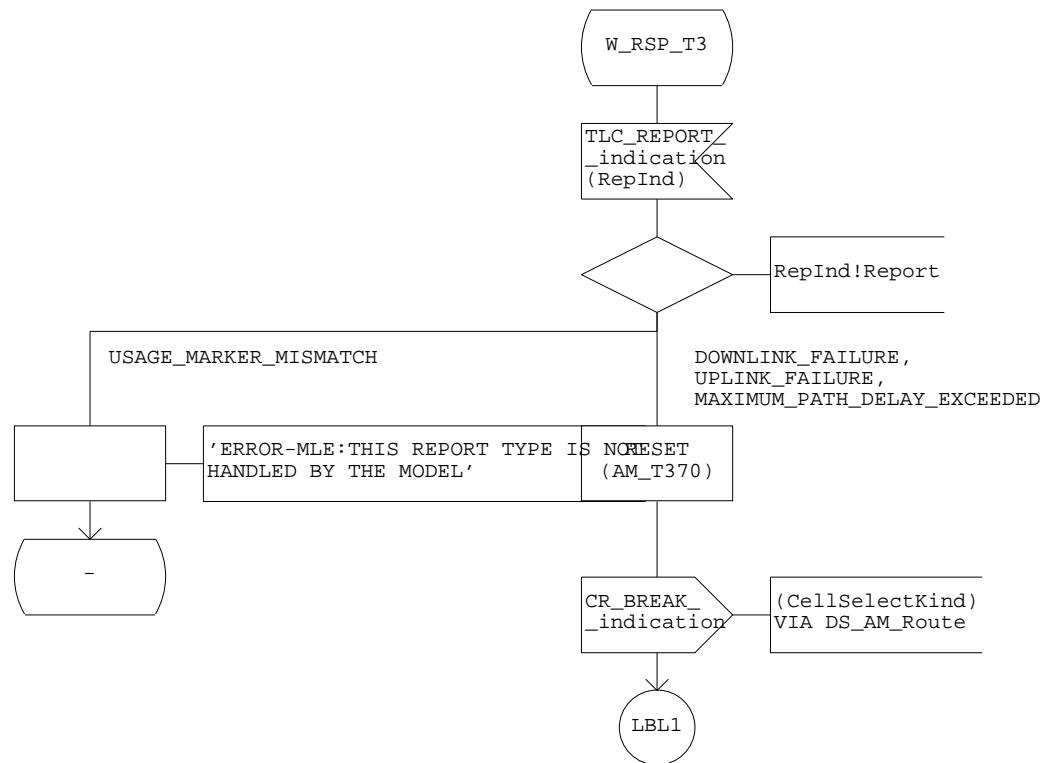
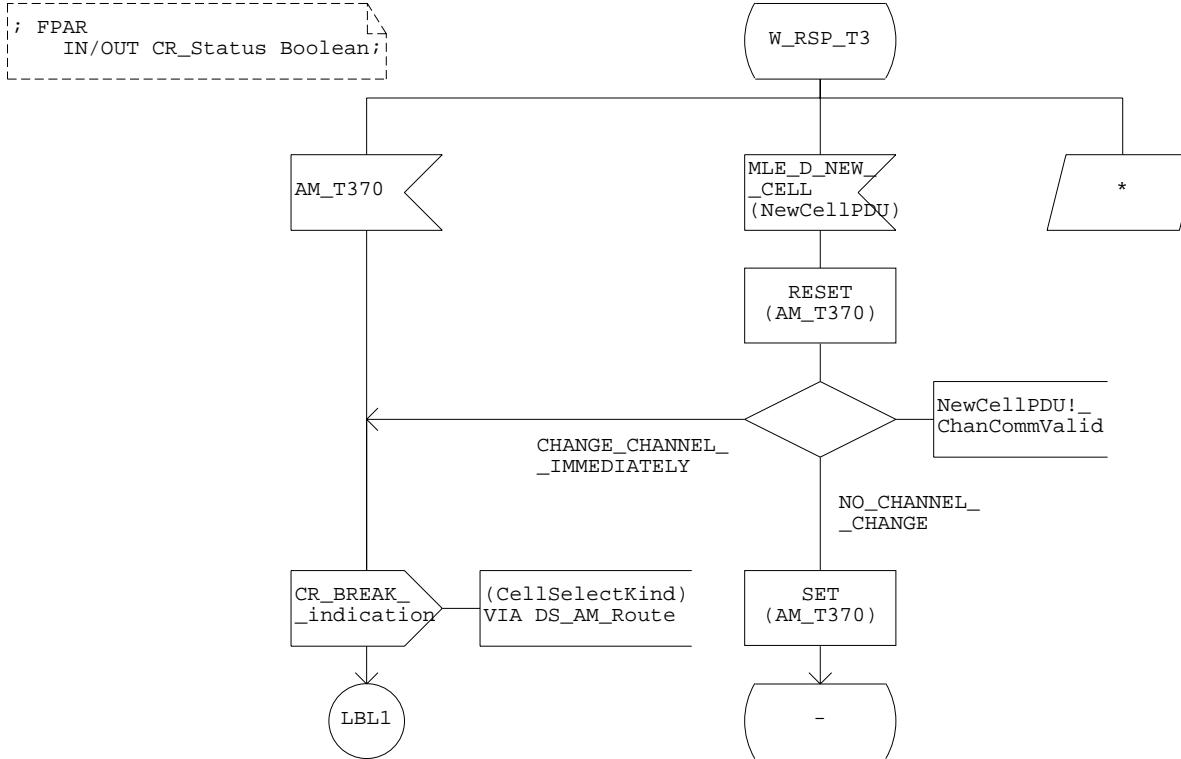
```
/** LLC SP DATA VARIABLES **/
```

```
DCL
  RepInd TLC_ReportIndicationType,
  SelReq TLC_SelectRequestType;
```



Procedure CR_Announ_T3

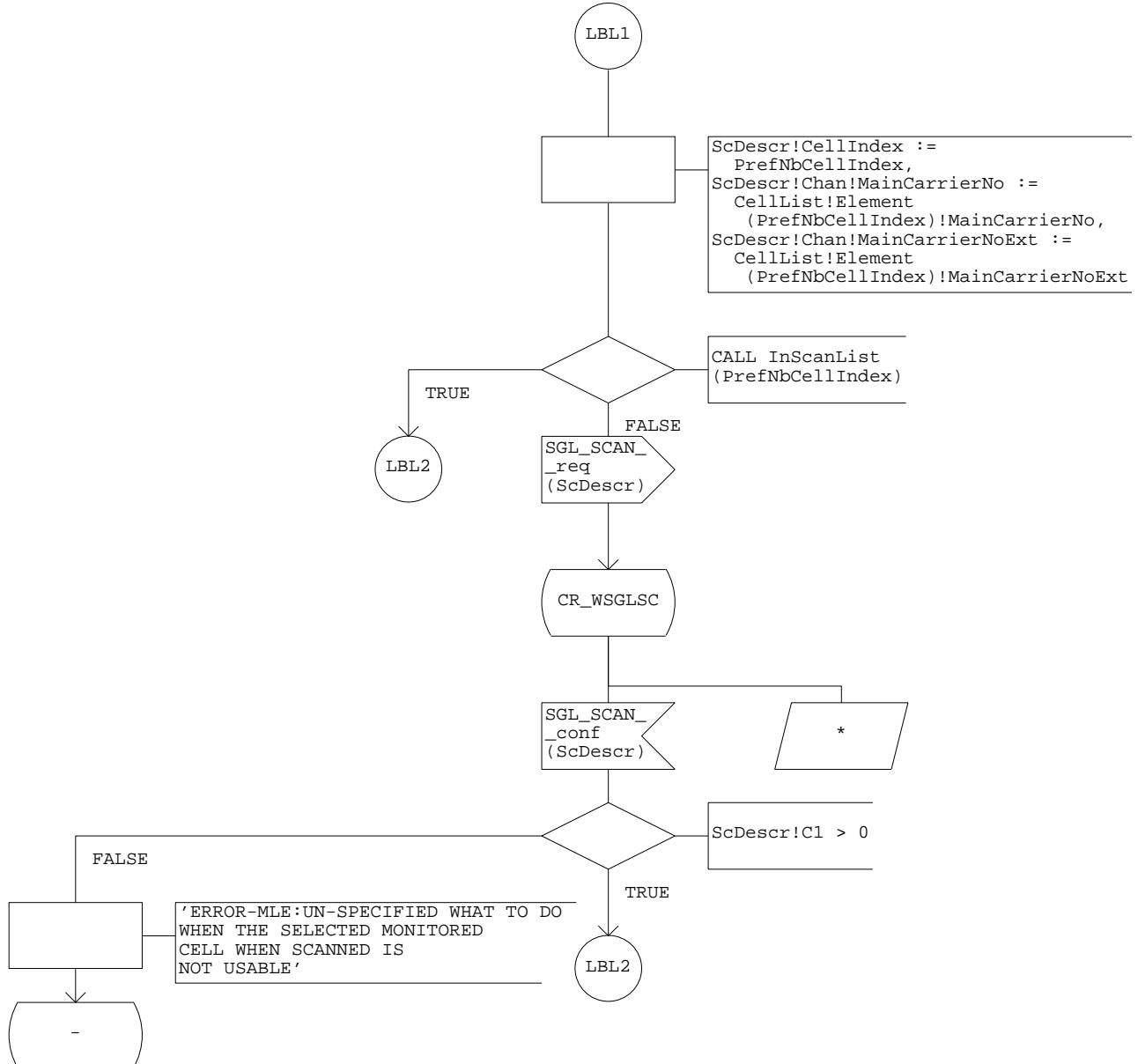
2(4)



Procedure CR_Announ_T3

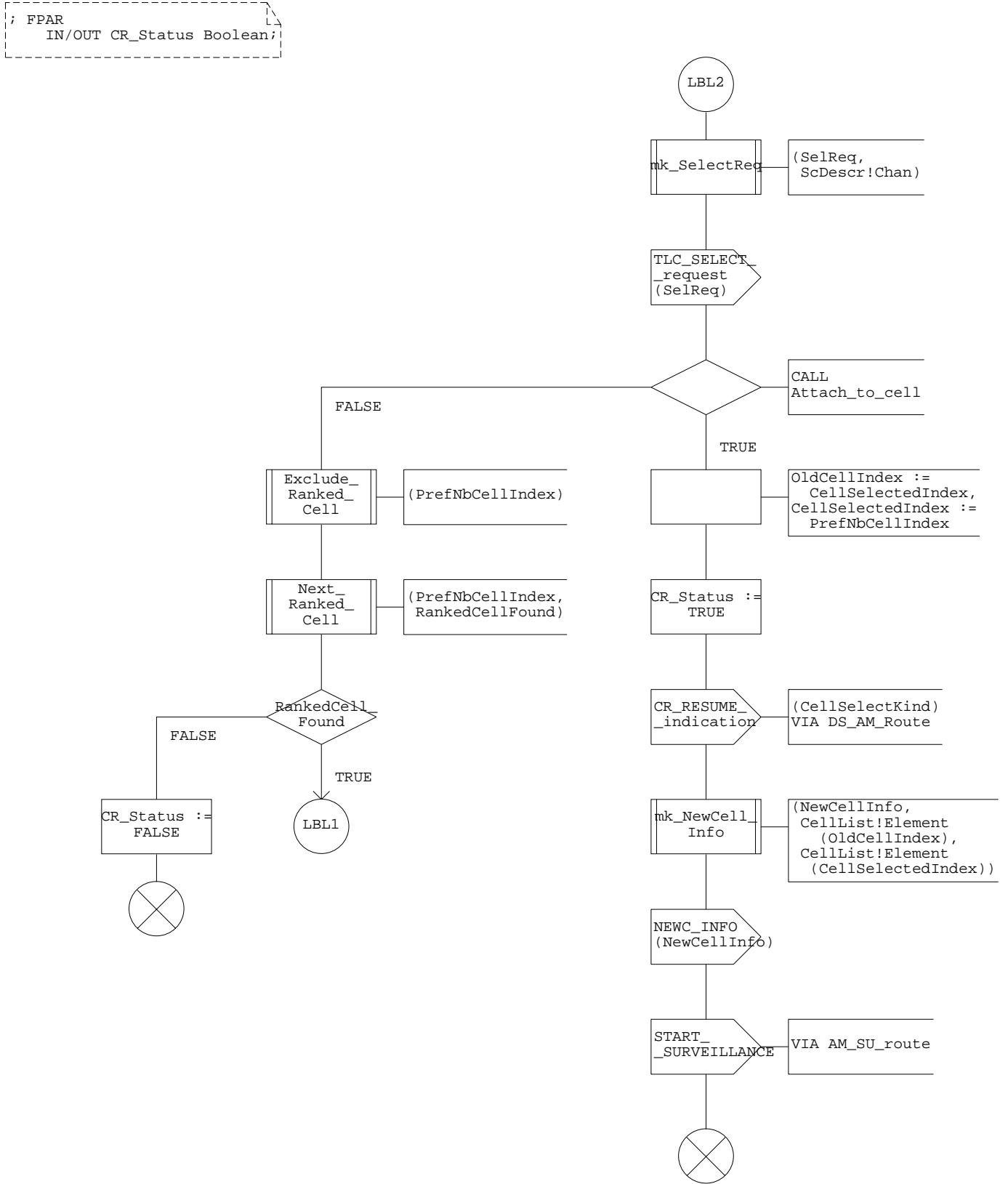
3(4)

```
; FPAR
IN/OUT CR_Status Boolean;
```



Procedure CR_Announ_T3

4 (4)



Procedure CR_Announ_T2

1(5)

```
; FPAR  
  IN/OUT CR_Status Boolean,  
  IN/OUT StayOnCell Boolean;
```

```
/* Perform announced cell reselection type 2  
   and set the parameter CR_Status  
   TRUE if the reselection was successful.  
*/
```

```
/** LOCAL VARIABLES **/
```

```
DCL  
  Chan          ChanType,  
  NewCellInfo   NewCellInfoType,  
  NwCellDsr    CellElemType,  
  RankedCellFound Boolean;
```

```
/** MLE PDU DATA VARIABLES **/
```

```
DCL  
  U_PrepPDU  MLE_U_PrepType,  
  PrepFailPDU MLE_D_PrepFailType,  
  NewCellPDU  MLE_D_NewCellType;
```

```
/** MLE SP DATA VARIABLES **/
```

```
DCL  
  LinkInd     MLE_LinkIndType,  
  UpdateReq   MLE_UpdateReqType;
```

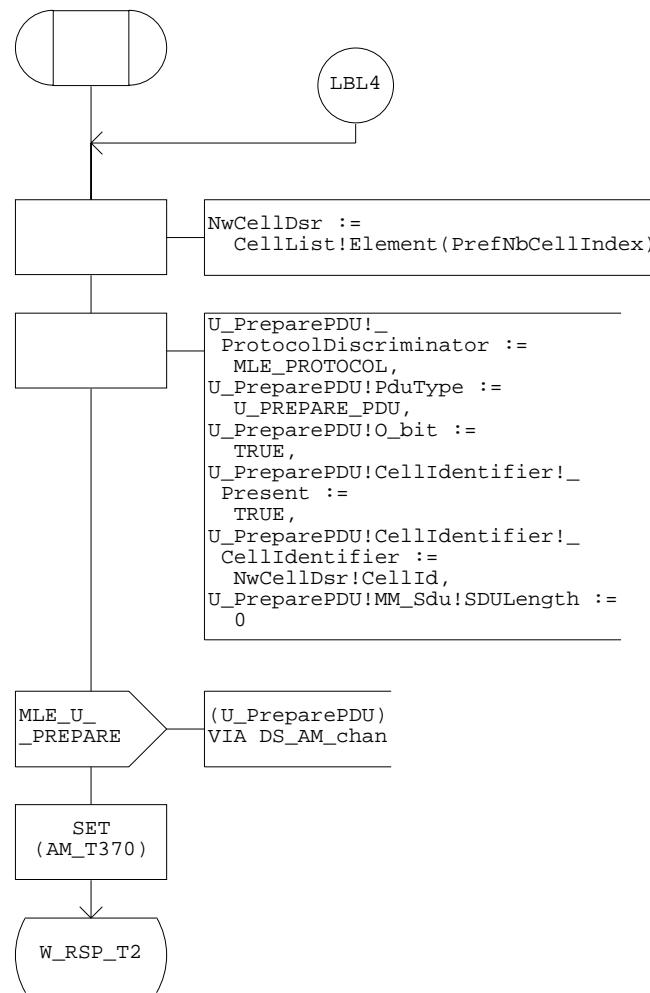
```
/** LLC SP DATA VARIABLES **/
```

```
DCL  
  RepInd      TLC_ReportIndicationType,  
  SelReq      TLC_SelectRequestType;
```

Procedure CR_Announ_T2

2(5)

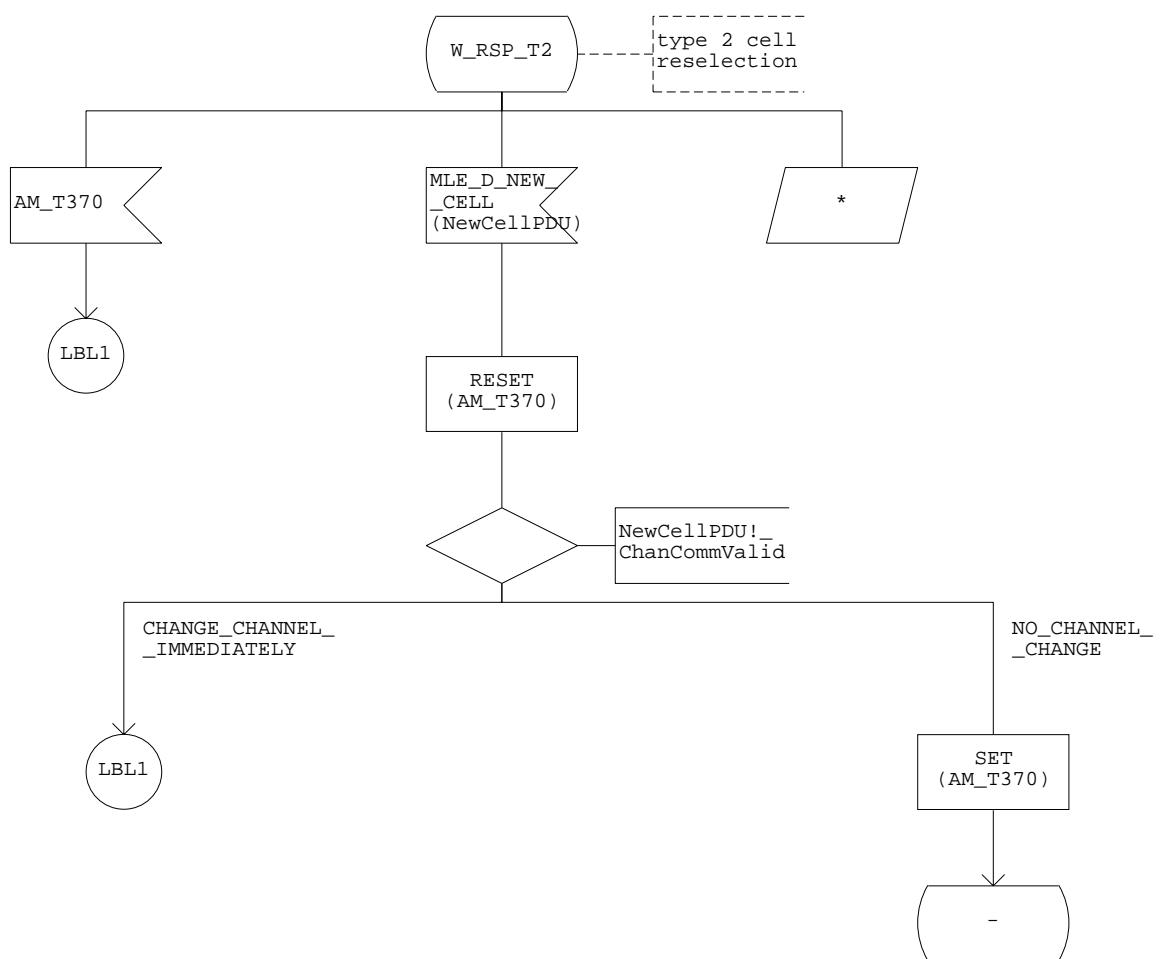
```
; FPAR
  IN/OUT CR_Status Boolean,
  IN/OUT StayOnCell Boolean;
```



Procedure CR_Announ_T2

3(5)

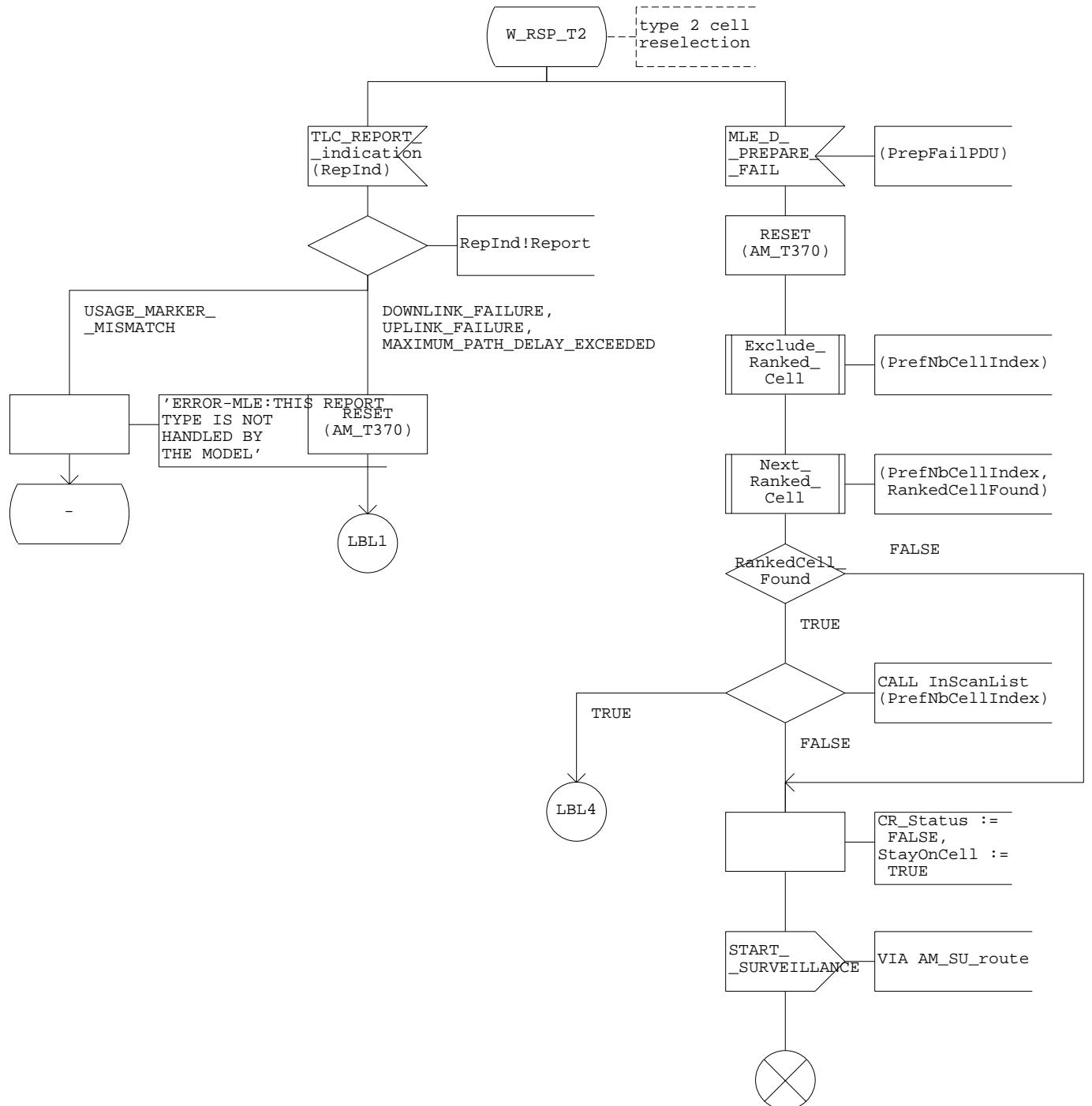
```
; FPAR
  IN/OUT CR_Status Boolean,
  IN/OUT StayOnCell Boolean;
```



Procedure CR_Announ_T2

4(5)

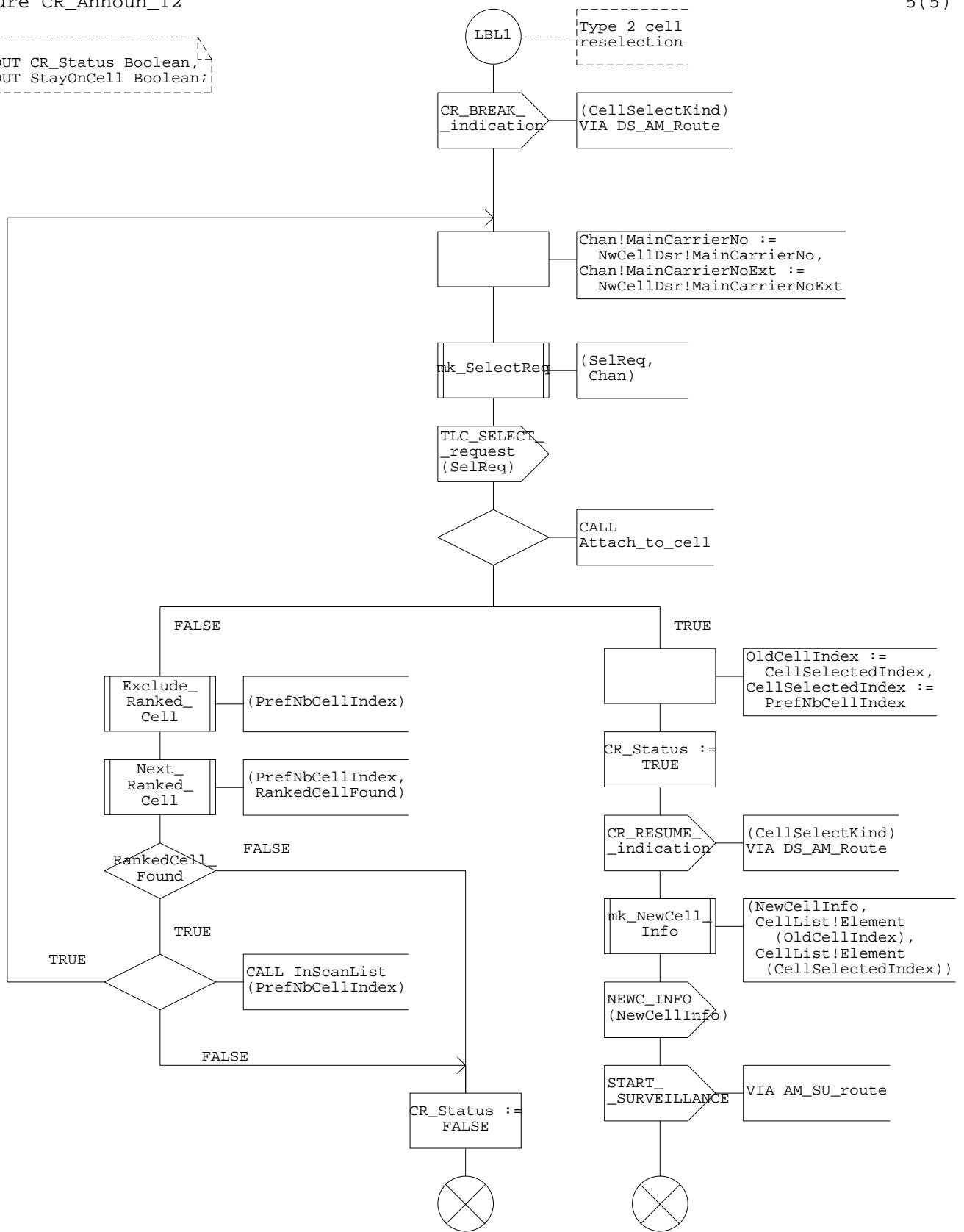
```
; FPAR
  IN/OUT CR_Status Boolean,
  IN/OUT StayOnCell Boolean;
```



Procedure CR_Announ_T2

```
; FPAR
  IN/OUT CR_Status Boolean,
  IN/OUT StayOnCell Boolean;
```

5 (5)

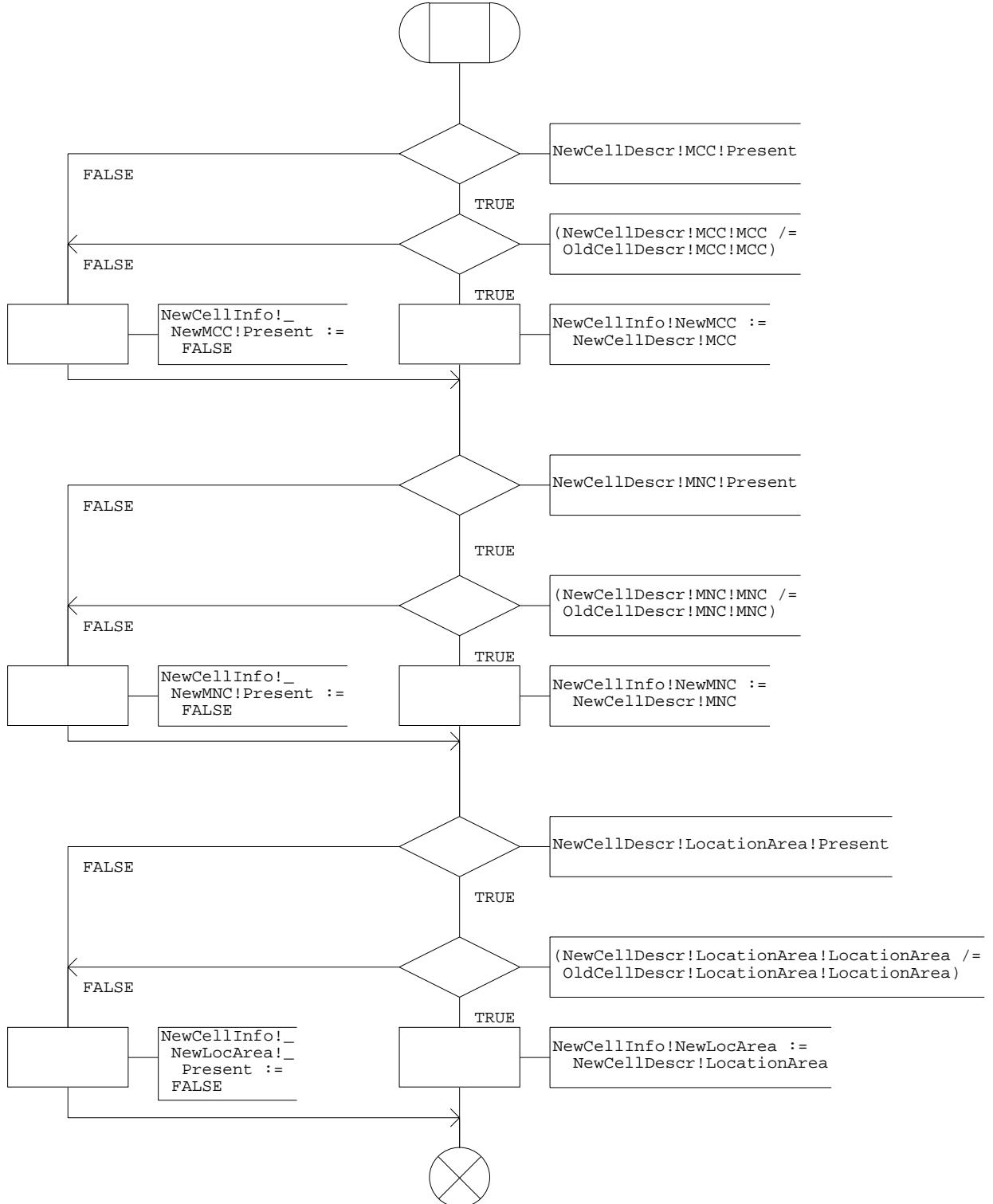


Procedure mk_NewCellInfo

1(1)

```
; FPAR
  IN/OUT NewCellInfo NewCellInfoType,
  IN NewCellDescr CellElemType,
  IN OldCellDescr CellElemType;
```

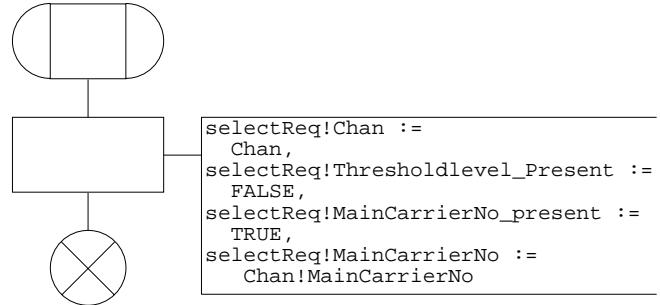
```
/* Insert information on the new MCC, MNC and
location area in the NewCellInfo descriptor
if it is different from the old cell.
*/
```



Procedure mk_SelectReq

1(1)

```
; FPAR
IN/OUT selectReq TLC_SelectRequestType;
IN Chan ChanType;
```



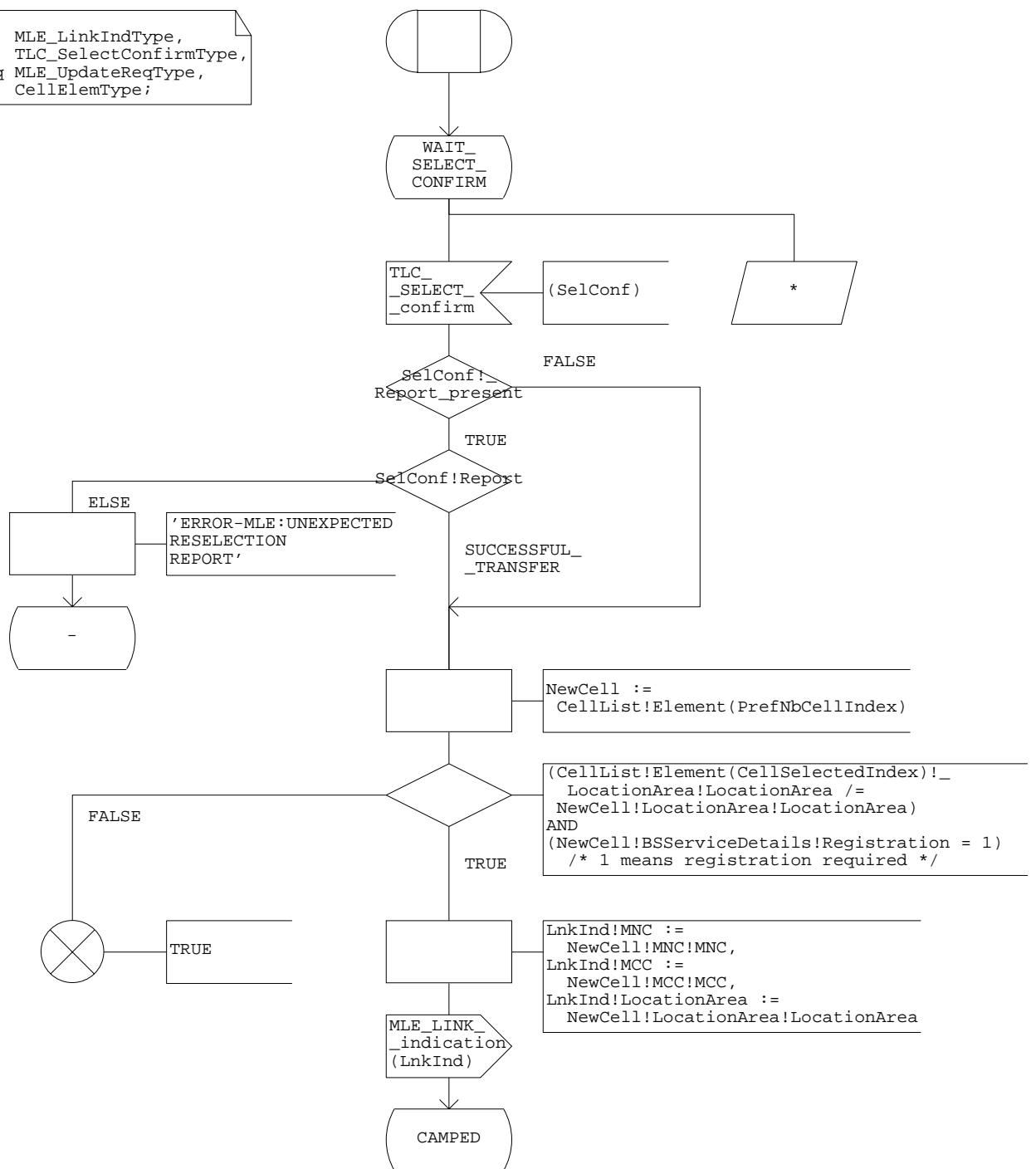
Procedure Attach_to_cell

1(2)

```
; RETURNS Boolean;
```

```
/* Common part of cell reselection procedure for
Announced type 2, type 3, Unannounced, and
Undeclared cell reselection.
The procedure returns TRUE if the attachment
is successful.
```

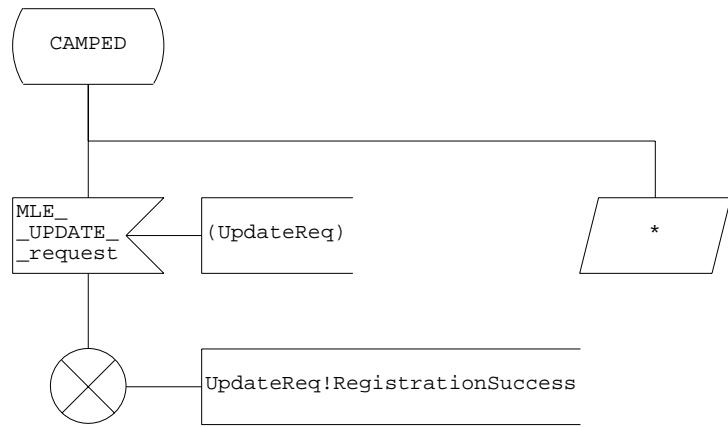
```
DCL
LnkInd      MLE_LinkIndType,
SelConf     TLC_SelectConfirmType,
UpdateReq   MLE_UpdateReqType,
NewCell     CellElemType;
```



Procedure Attach_to_cell

2(2)

; RETURNS Boolean;



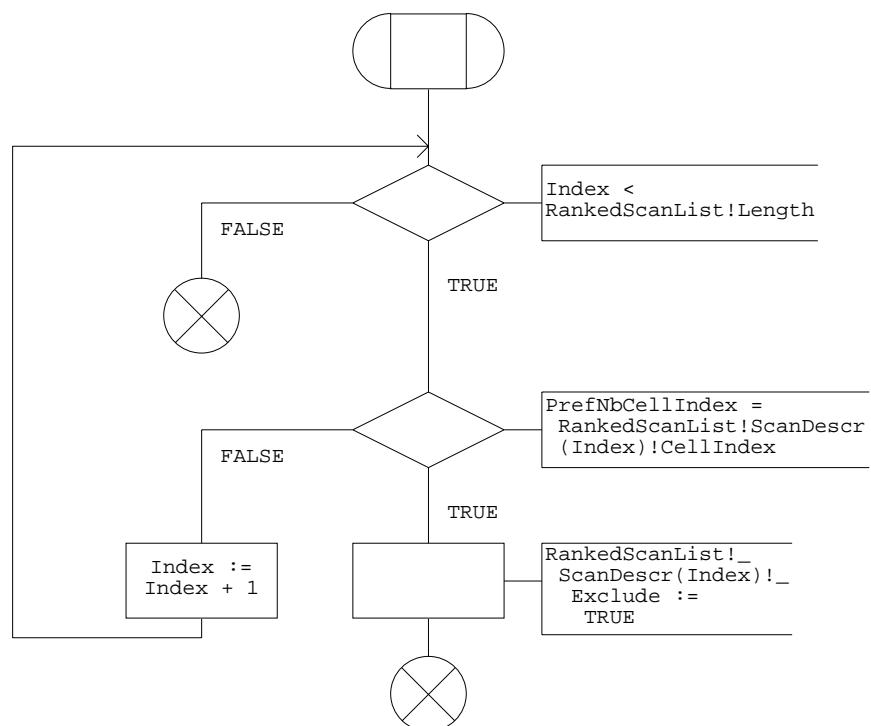
Procedure ExcludeRankedCell

1(1)

```
; FPAR
  IN PrefNbCellIndex Natural;
```

```
/* Update the List of scanned ranked cells
so that the examined neighbour cell
PrefNeighbourCellIndex is excluded from
next search for a new cell to select.
If the cell is not in the scanned
list no update takes place.
```

```
DCL
  Index Natural := 0;
```



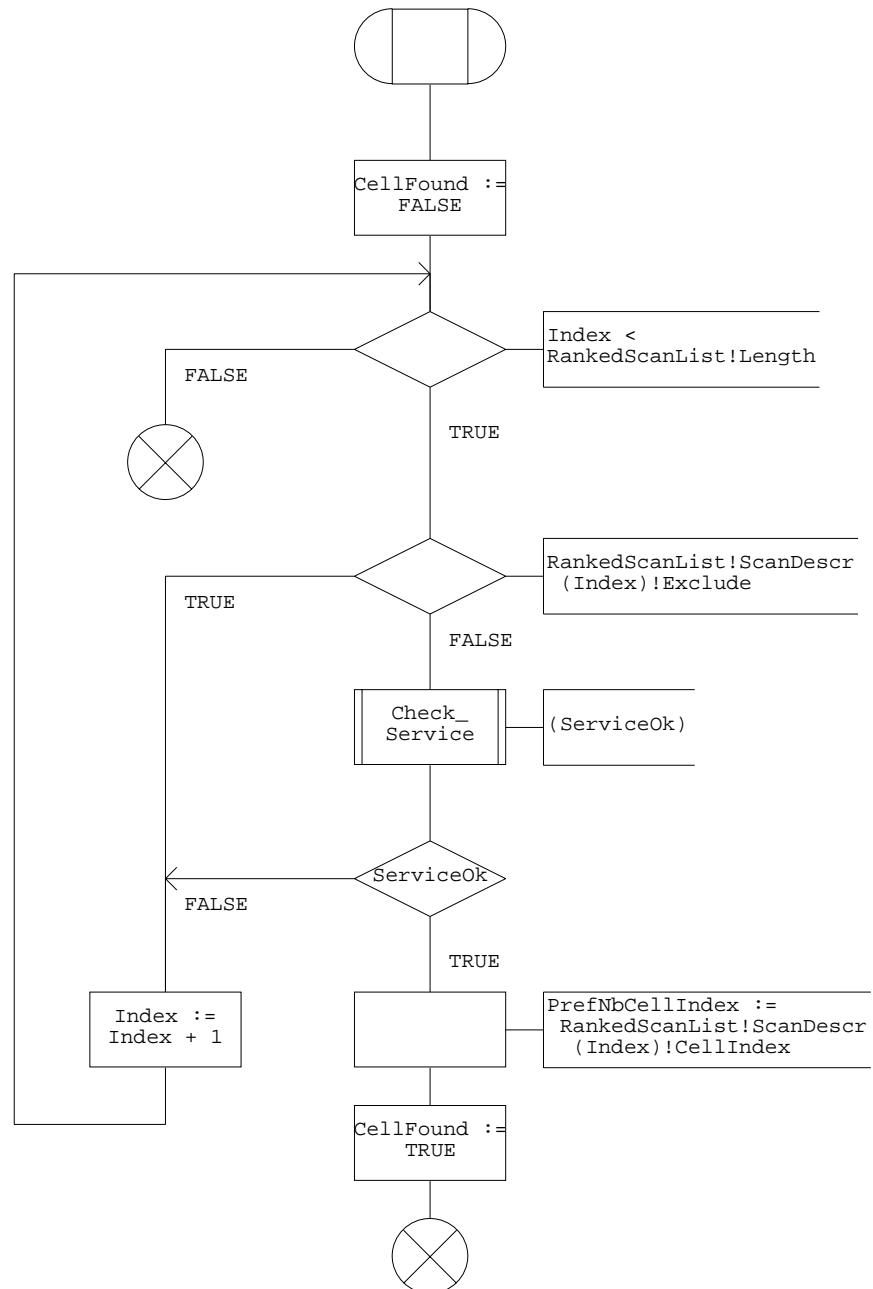
Procedure NextRankedCell

1(1)

```
; FPAR
  IN/OUT PrefNbCellIndex Natural,
  IN/OUT CellFound Boolean;
```

```
/* Search and return the best possible scanned
cell from the list of ranked cells, if any.
If found the index to the in the cell list
is returned in the PrefNbCellIndex
parameter and the CellFound parameter is
TRUE. Otherwise FALSE is returned.
```

```
DCL
  Index  Natural,
  ServiceOk Boolean;
```

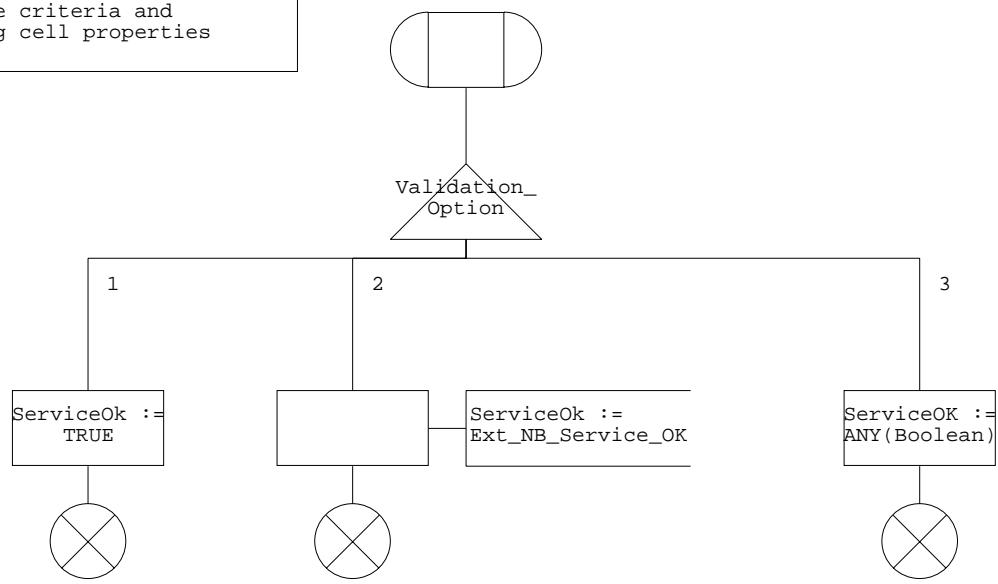


Procedure CheckService

1(1)

```
; FPAR  
IN/OUT ServiceOk Boolean;
```

```
/* Check if the suggested preferred  
cell satisfies the service  
requirements with respect to  
the service criteria and  
the serving cell properties  
*/
```



Process Monitor_Cells

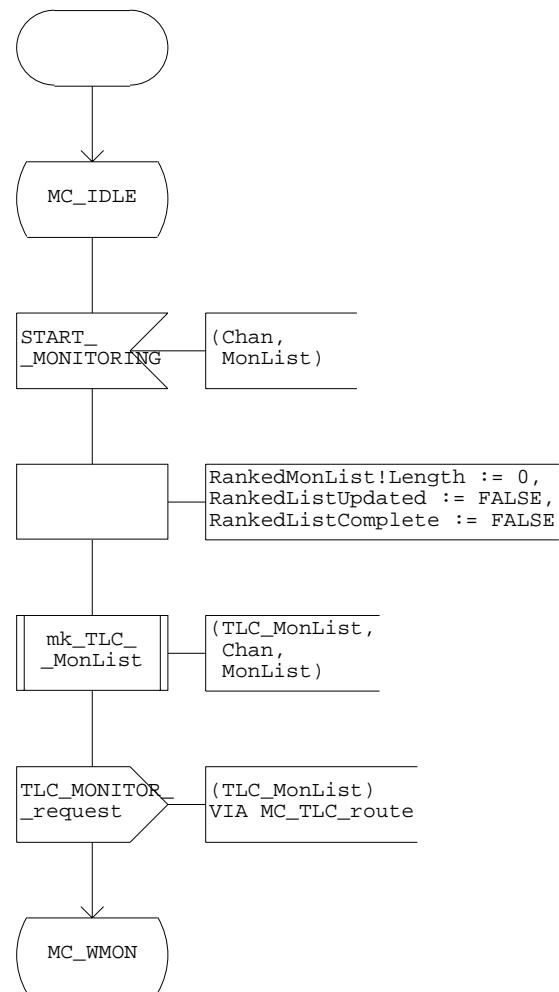
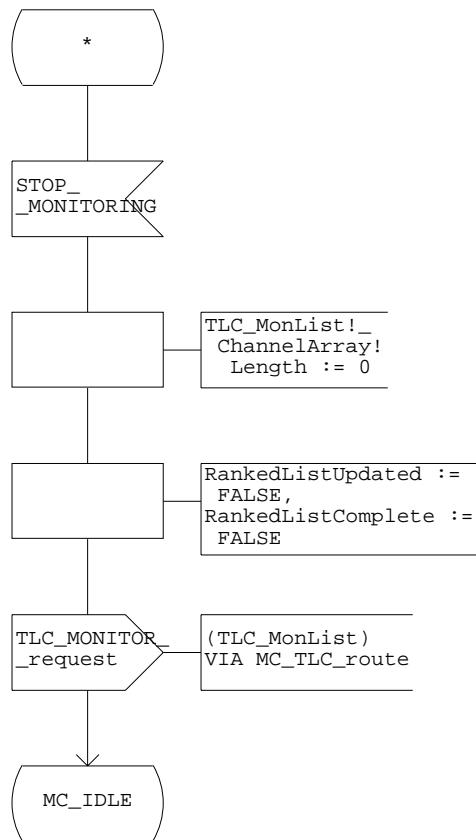
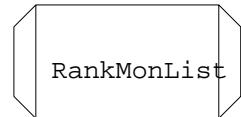
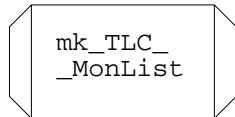
1 (2)



```
DCL
  Chan          ChanType,
  MonList      MonListType,
  RankedMonList MonListType,
  RankedListUpdated Boolean := FALSE,
  RankedListComplete Boolean := FALSE;
```

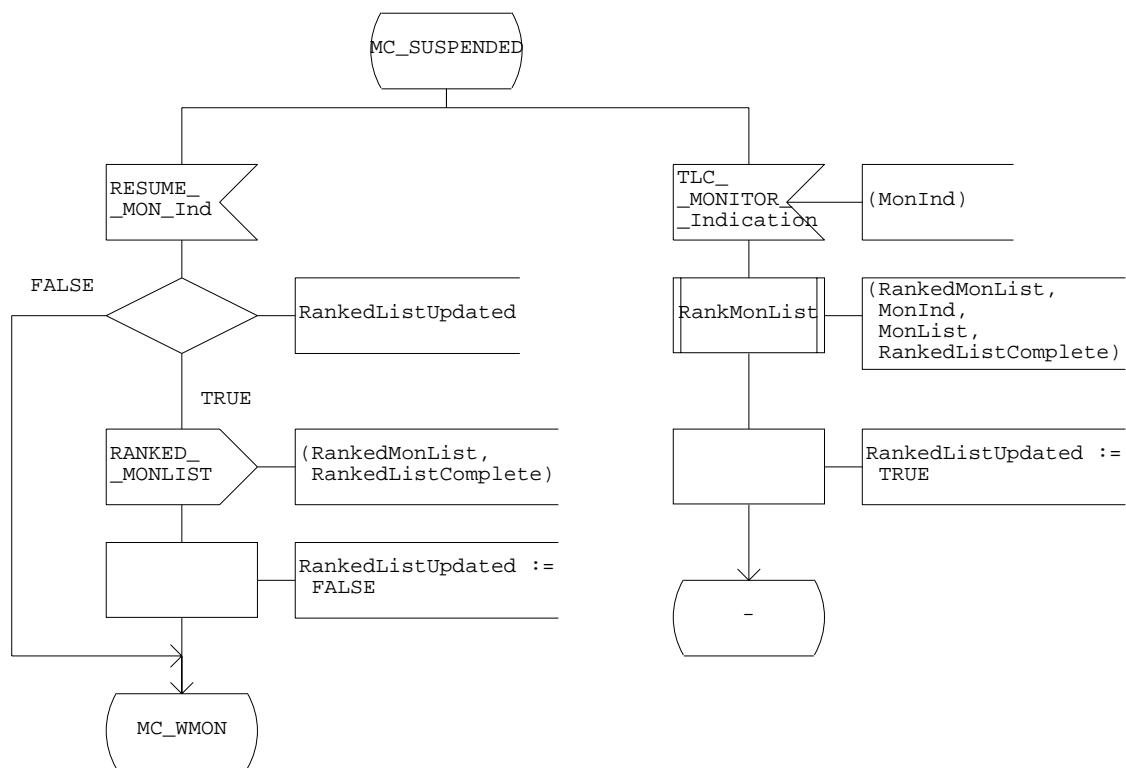
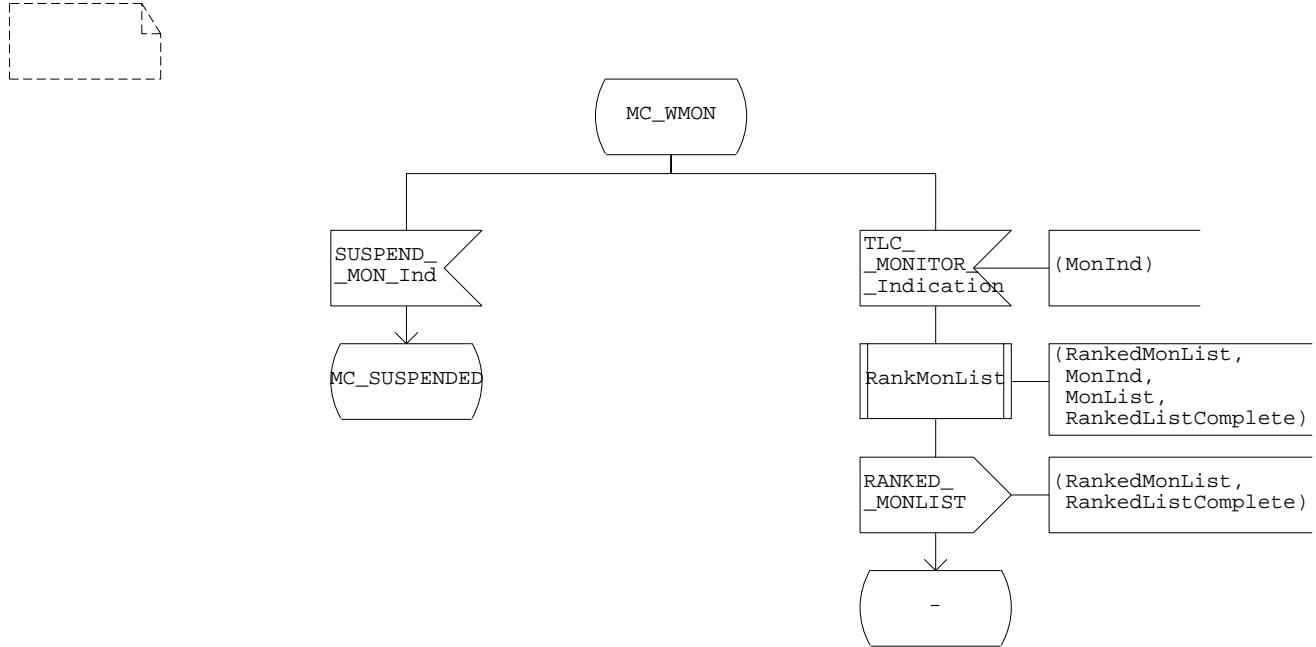
```
/** PROCEDURE REFERENCES **/
```

```
DCL
  MonInd      TLC_MonitorIndicationType,
  TLC_MonList TLC_MonitorRequestType;
```



Process Monitor_Cells

2 (2)



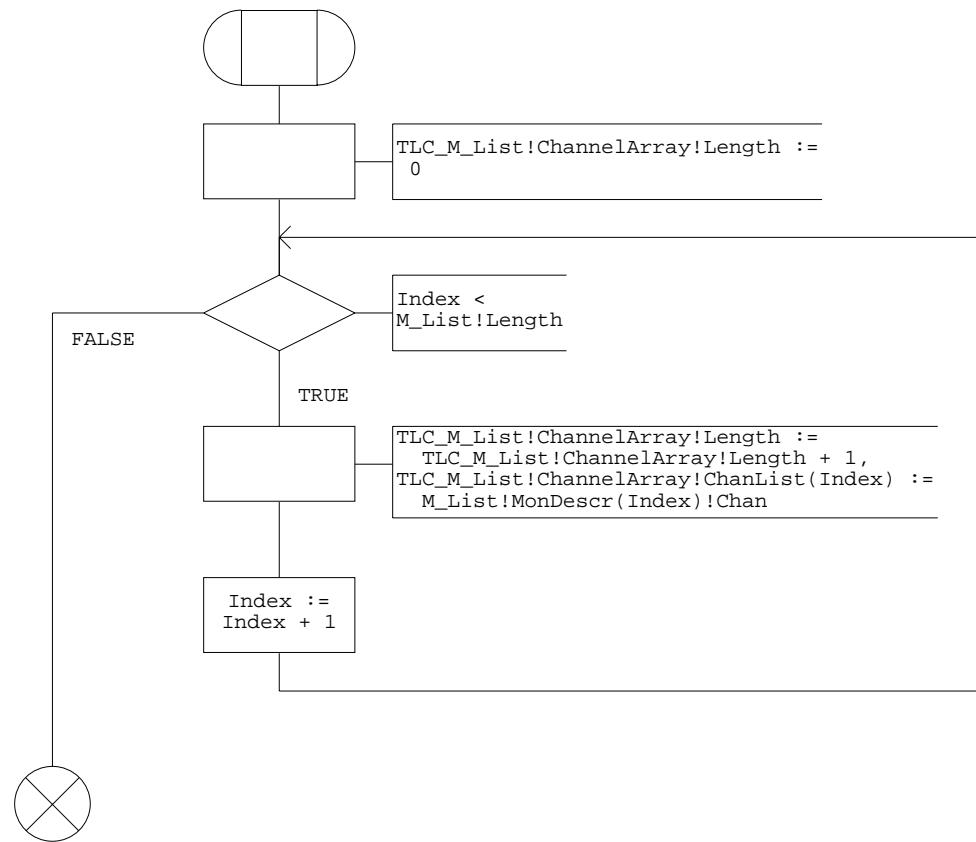
Procedure mk_TLC_MonList

1(1)

```
; FPAR
  IN/OUT TLC_M_List TLC_MonitorRequestType;
  IN Chan ChanType,
  IN M_List MonListType;
```

```
/* Build the TLC monitor list from the list of
neighbour cells to be monitored described in
M_List
*/
```

```
DCL
Index Natural := 0;
```



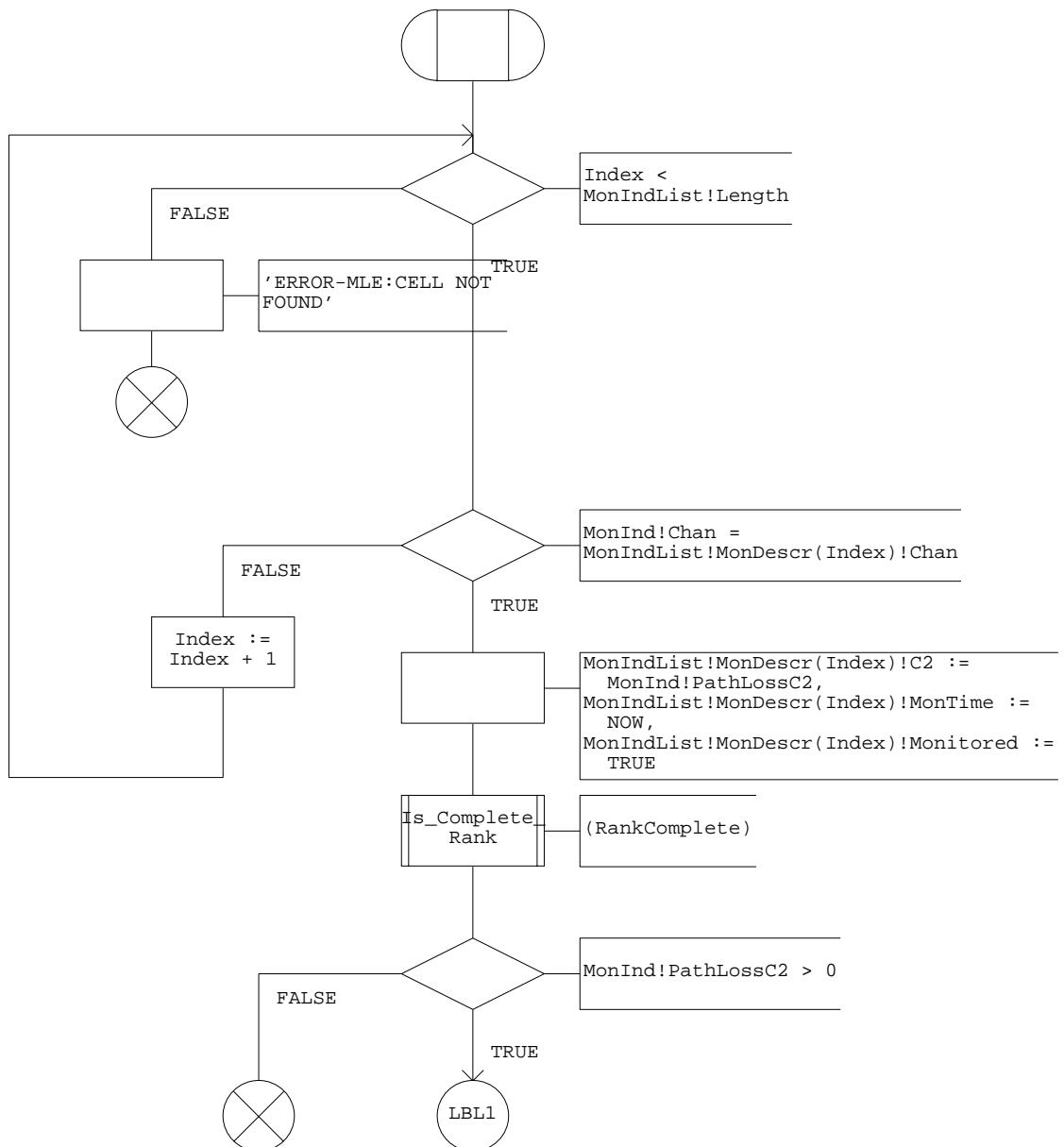
Procedure RankMonList

1 (2)

```
; FPAR
  IN/OUT RankedList MonListType,
  IN MonInd TLC_MonitorIndicationType,
  IN/OUT MonIndList MonListType,
  IN/OUT RankComplete Boolean;
```

```
DCL
  Index Natural := 0,
  R_Indx Natural := 0,
  Li Natural;
```

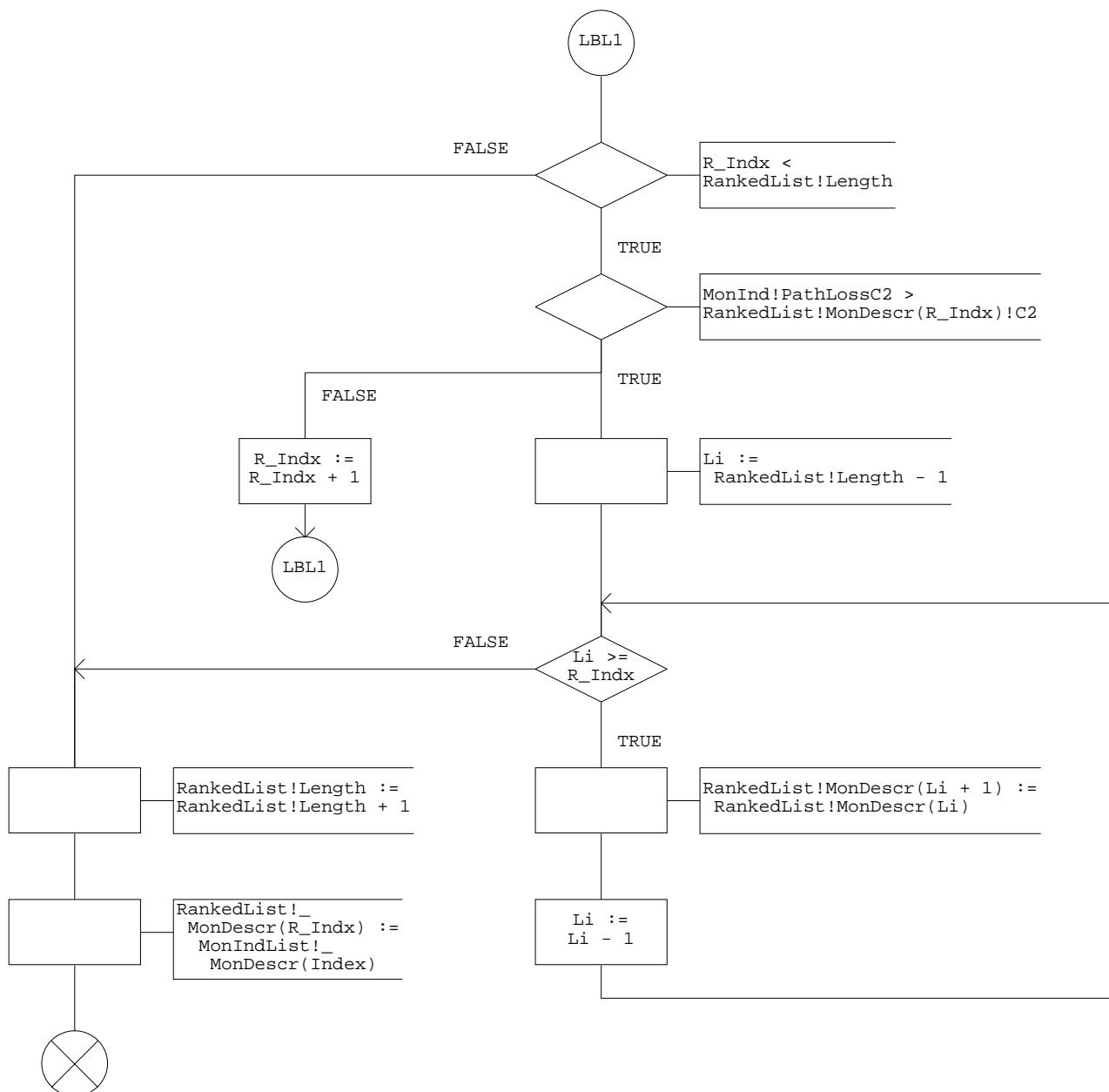
Is_Complete
Rank



Procedure RankMonList

2(2)

```
; FPAR
  IN/OUT RankedList MonListType,
  IN MonInd TLC_MonitorIndicationType,
  IN/OUT MonIndList MonListType,
  IN/OUT RankComplete Boolean;
```



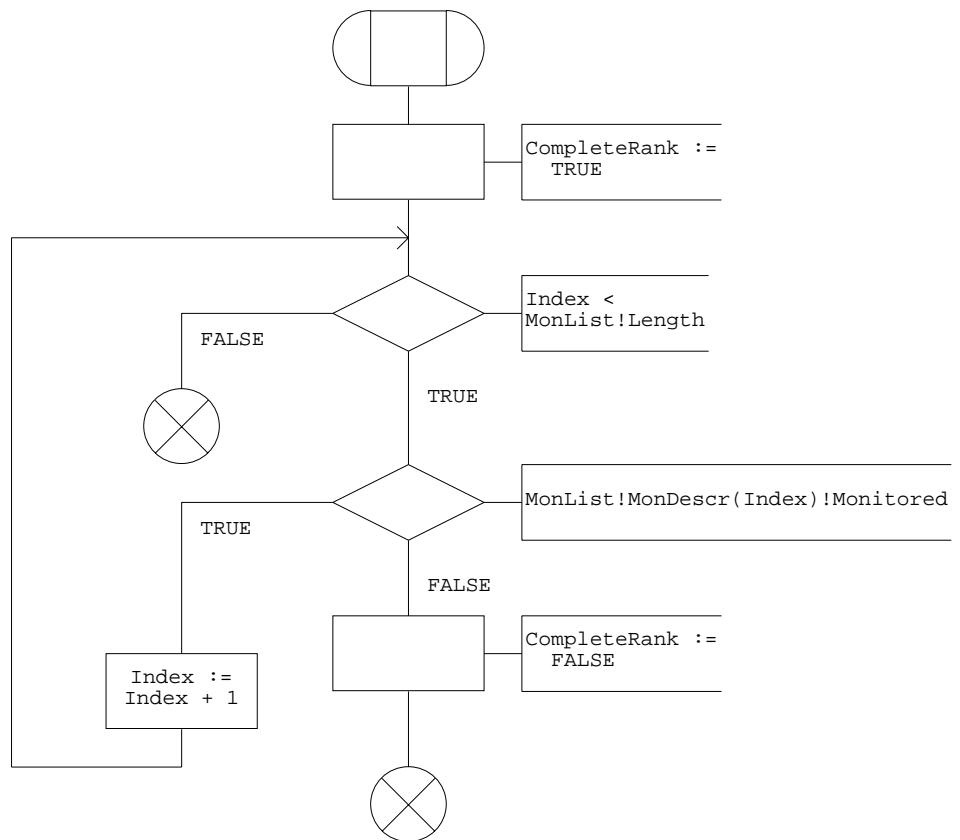
Procedure Is_CompleteRank

1(1)

```
; FPAR  
  IN/OUT CompleteRank Boolean;  
  IN MonList MonListType;
```

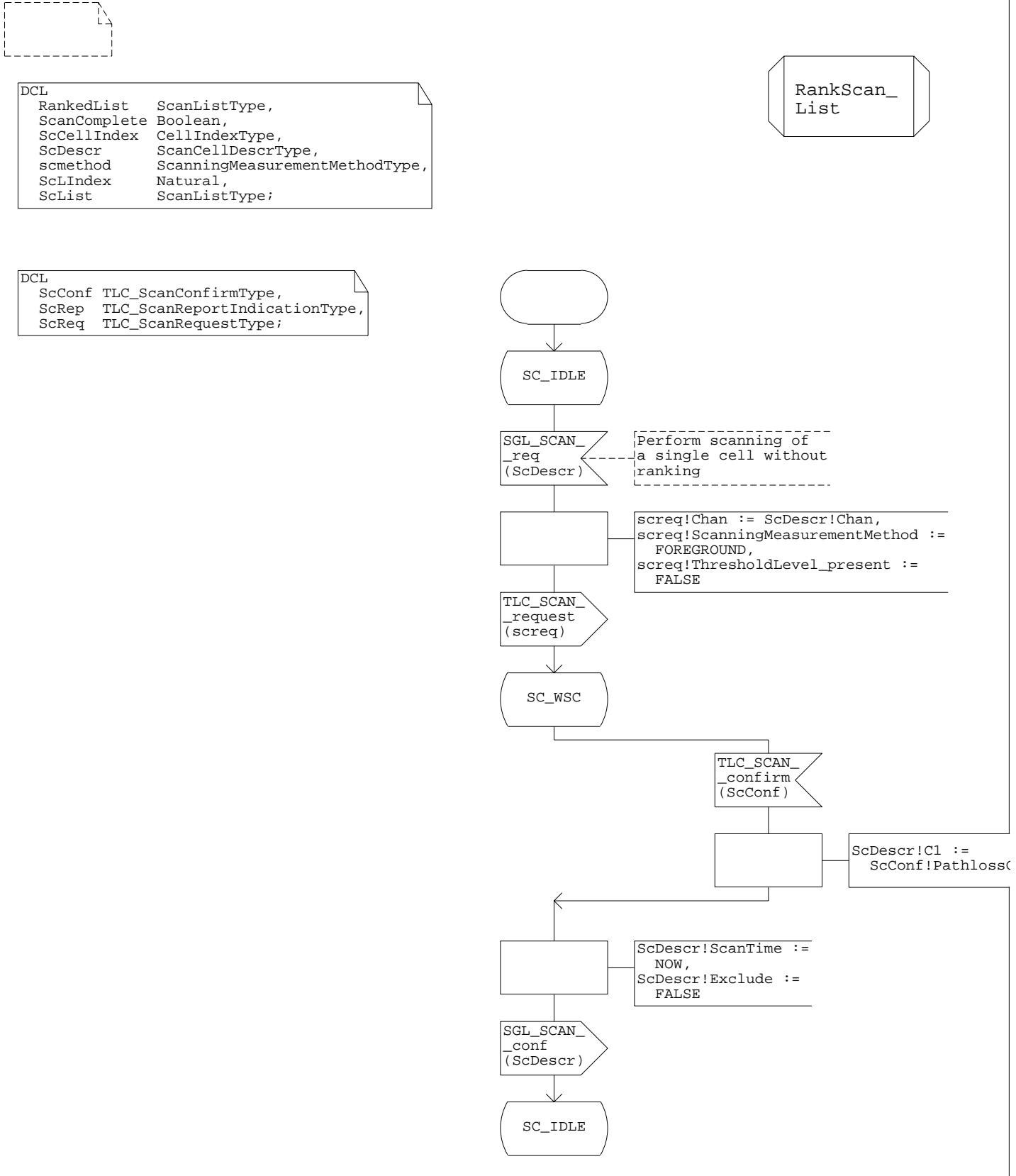
```
/* Check if all neighbour cells has been monitored  
   and if so set parameter CompleteRank to TRUE  
*/
```

```
DCL  
  Index Natural := 0;
```



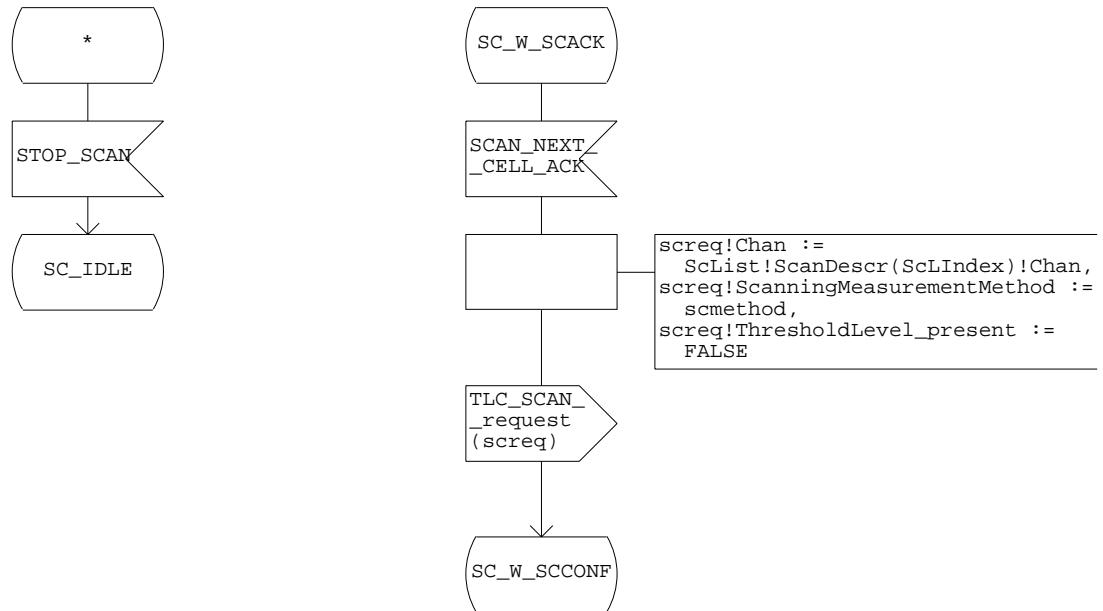
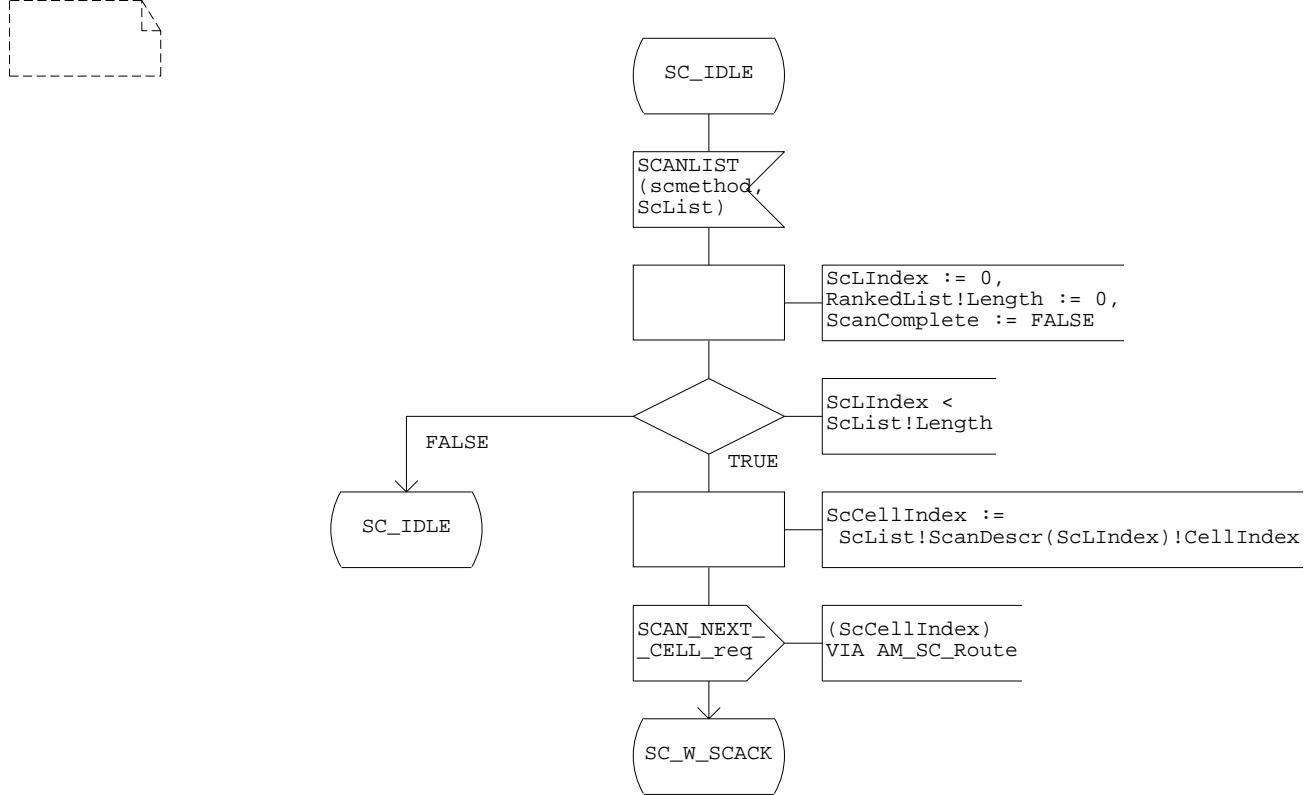
Process Scan_Cells

1 (5)



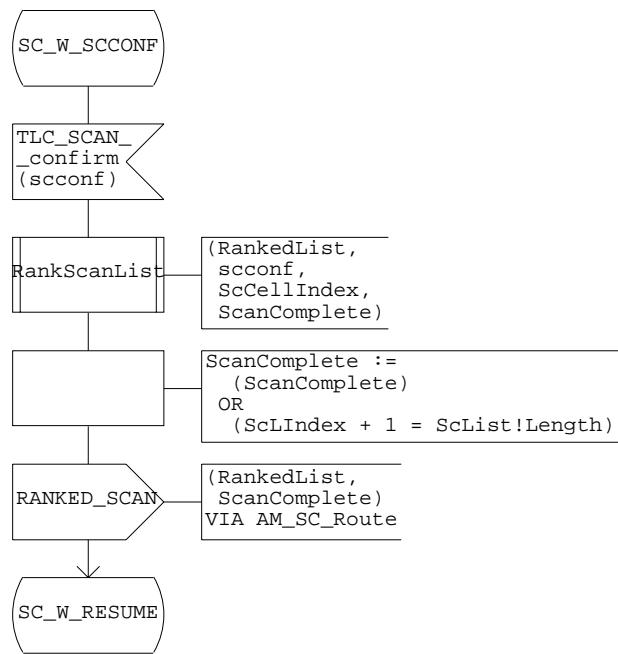
Process Scan_Cells

2 (5)



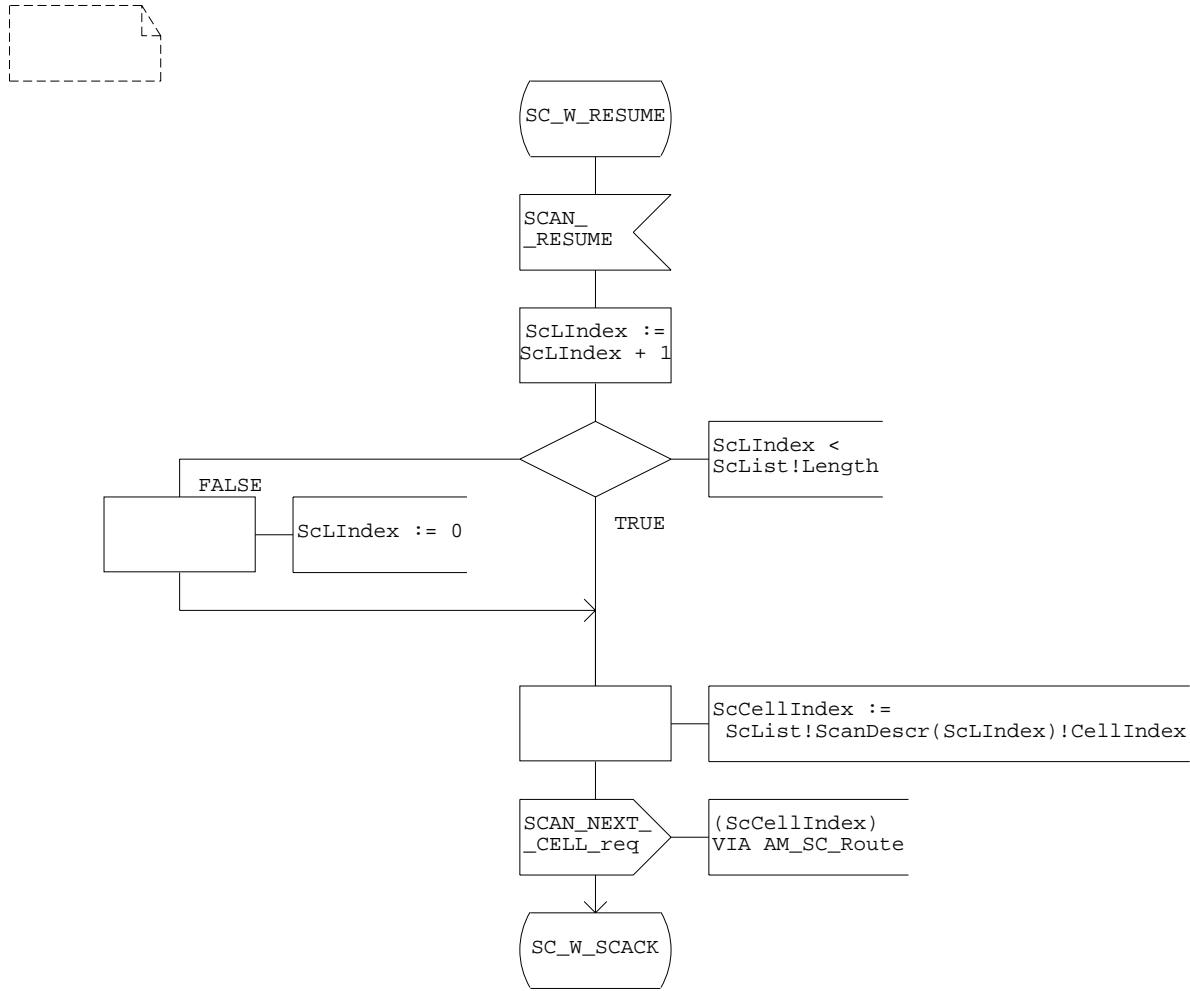
Process Scan_Cells

3 (5)



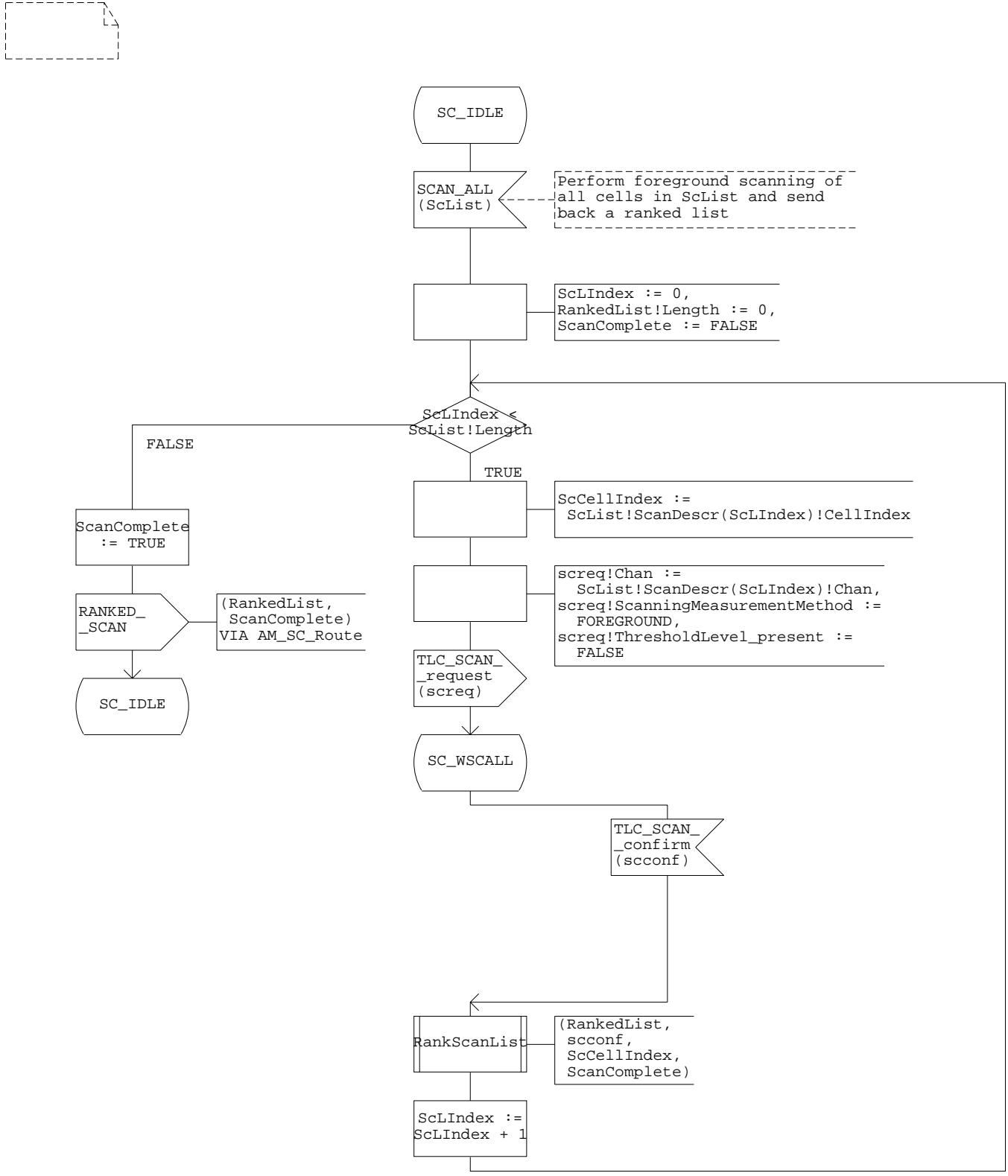
Process Scan_Cells

4 (5)



Process Scan_Cells

5 (5)

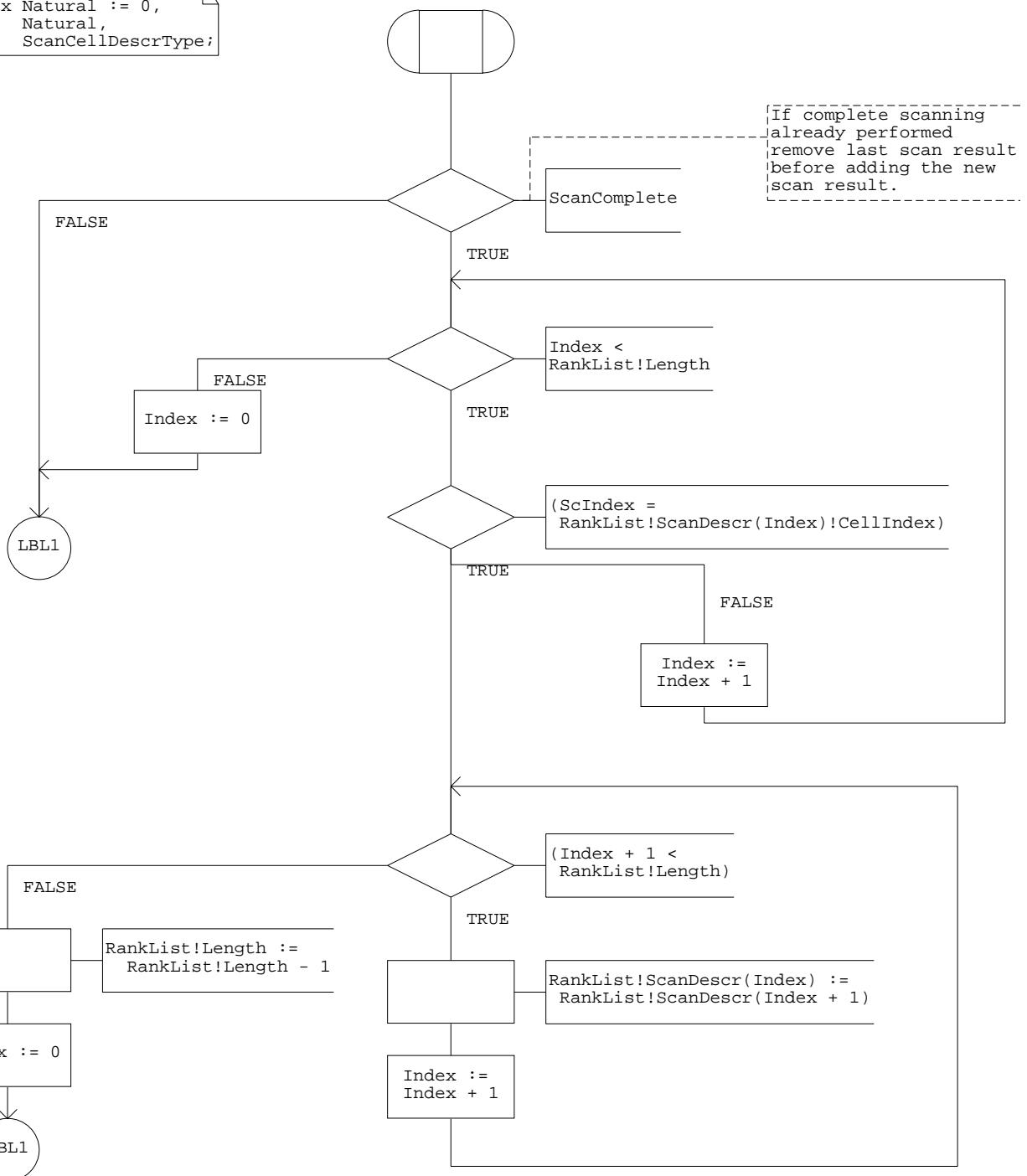


Procedure RankScanList

1 (2)

```
; FPAR
  IN/OUT RankList ScanListType,
  IN ScConf TLC_ScanConfirmType,
  IN scIndex CellIndexType,
  IN ScanComplete Boolean;
```

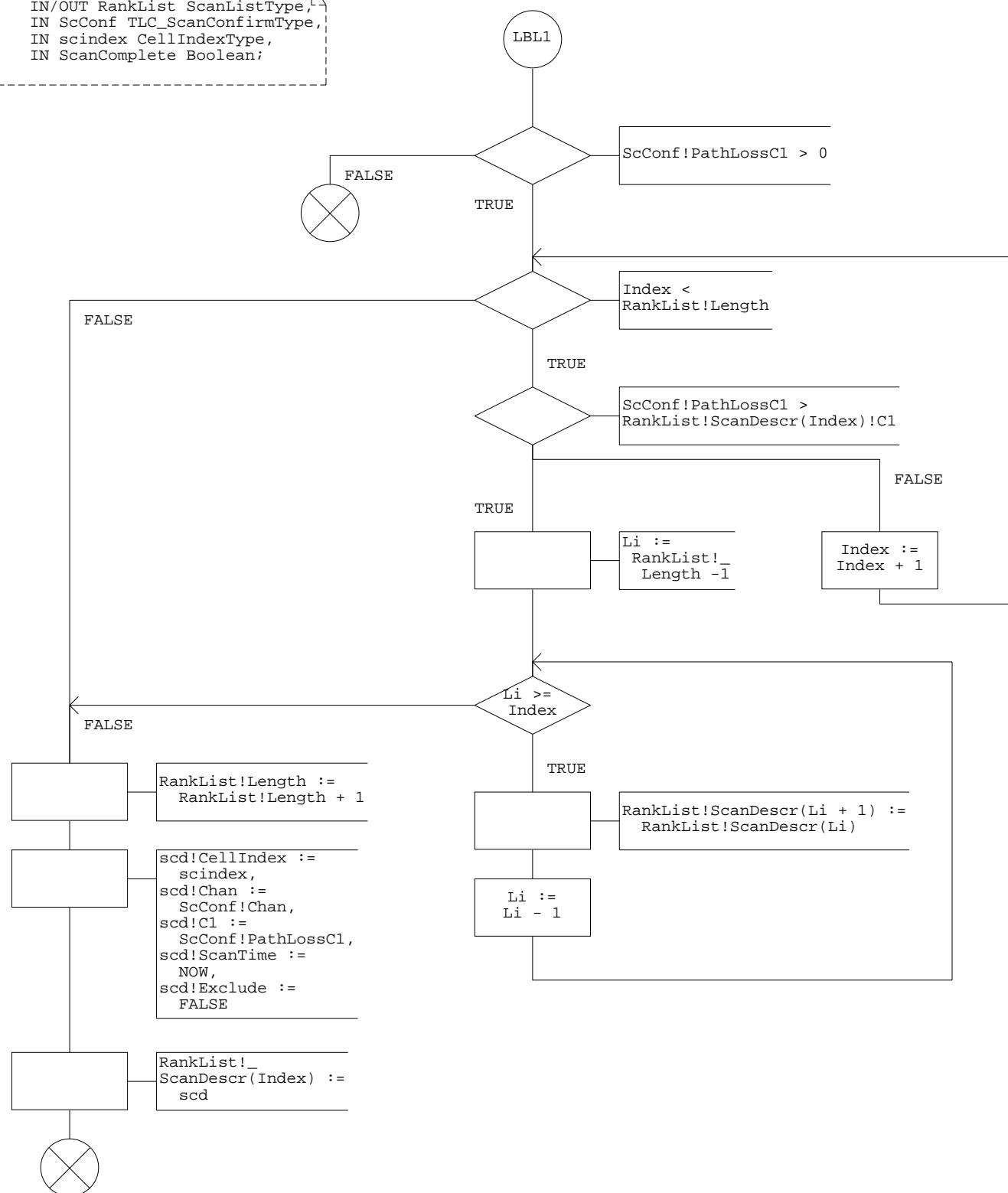
```
DCL
  Index Natural := 0,
  Li  Natural,
  scd ScanCellDescrType;
```

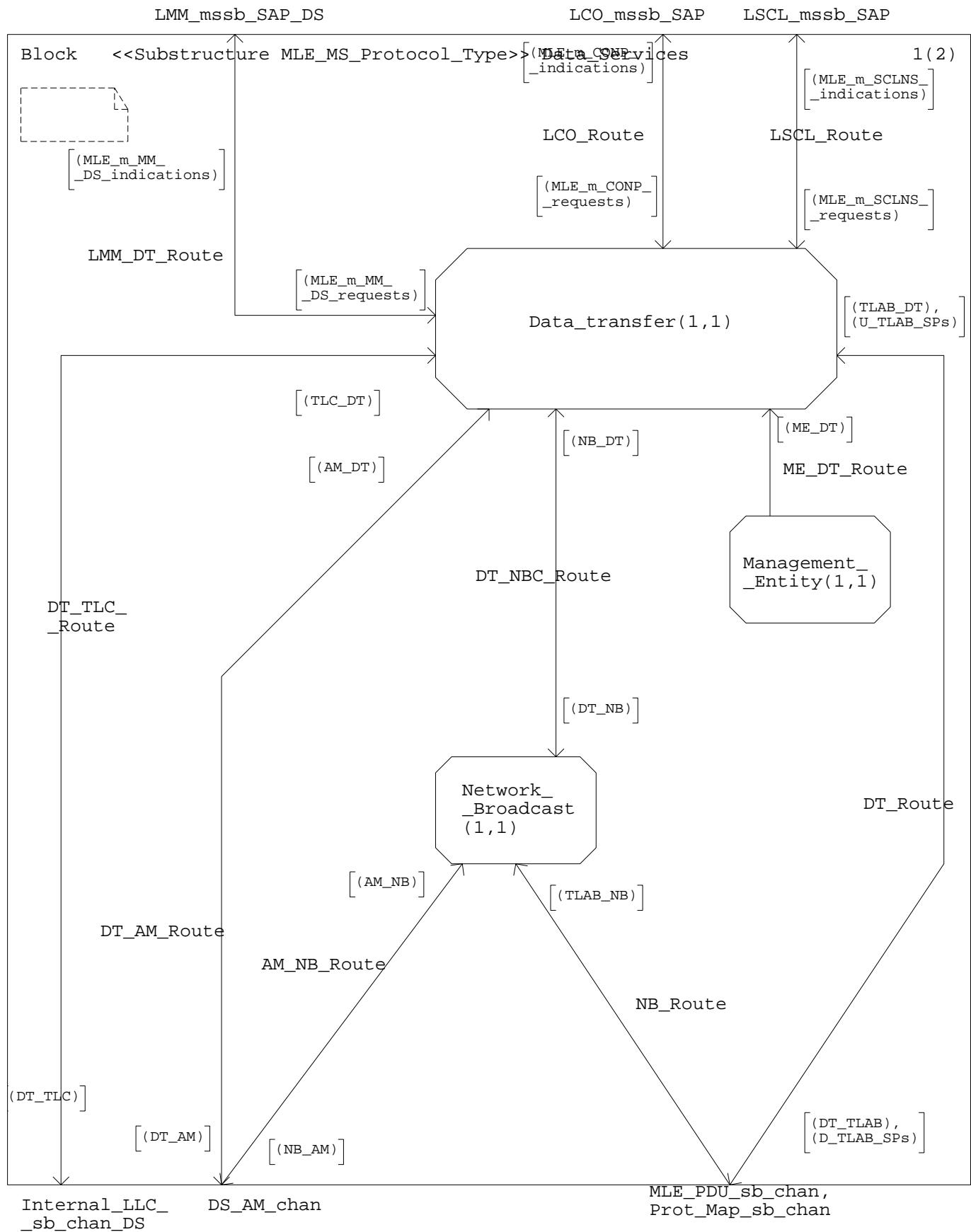


Procedure RankScanList

2(2)

```
; FPAR
  IN/OUT RankList ScanListType,
  IN ScConf TLC_ScanConfirmType,
  IN scindex CellIndexType,
  IN ScanComplete Boolean;
```





Block <<Substructure MLE_MS_Protocol_Type>> Data_Services

2(2)



```
/* Transfer the TMI from the  
Management Entity to the  
Data Transfer Entity  
*/  
  
SIGNAL SetTMI( TMI_Type );
```

```
SIGNALLIST TLC_DT =  
    TLC_CONFIGURE_confirm;  
  
SIGNALLIST DT_TLC =  
    TLC_CONFIGURE_request;  
  
SIGNALLIST TLAB_NB =  
    (TLB_Downlink_PDUs);  
  
SIGNALLIST DT_NB =  
    MLE_D_PREPARE_FAIL,  
    MLE_D_NWRK_BROADCAST;  
  
SIGNALLIST NB_DT =  
    MLE_D_NWRK_BROADCAST,  
    MLE_U_PREPARE,  
    MLE_D_SIN1;  
  
SIGNALLIST AM_DT =  
    CR_BREAK_indication,  
    CR_RESUME_indication,  
    MLE_BREAK_indication,  
    MLE_REOPEN_indication,  
    MLE_U_PREPARE,  
    NEWC_INFO;  
  
SIGNALLIST DT_AM =  
    MLE_D_NEW_CELL,  
    MLE_D_NWRK_BROADCAST,  
    MLE_D_PREPARE_FAIL,  
    MLE_OPEN_request;  
  
SIGNALLIST AM_NB =  
    SCANSTATUS;  
  
SIGNALLIST NB_AM =  
    SCANSTATUS_ACK,  
    MLE_D_NWRK_BROADCAST,  
    MLE_D_SIN2,  
    MLE_D_SIN1;  
  
SIGNALLIST TLAB_DT =  
    (TLA_DownLink_PDUs);  
  
SIGNALLIST DT_TLAB =  
    (TLA_Uplink_PDUs);  
  
SIGNALLIST ME_DT =  
    SetTMI;
```

Process Data_transfer

1(25)

```
/* *** LOCAL DATA TYPE DEFINITIONS ***/
```

```
/* Data types for current transmissions in the data transfer process */

NEWTYPE TransmissionDescrType STRUCT
  Protocol      MLE_ProtocolDiscriminatorType;
  EndpointID    Endpoint_ID_Type;
  Layer2Service Layer2ServiceType;
  CancelPossible Boolean;
ENDNEWTYPE TransmissionDescrType;

NEWTYPE TransmissionArrayType
  ARRAY(Natural,TransmissionDescrType)
ENDNEWTYPE;

NEWTYPE TransmissionListType STRUCT
  Length Natural;
  Transmission TransmissionArrayType;
ENDNEWTYPE TransmissionListType;
```

```
/* *** LOCAL SYNONYM DEFINITIONS ***/
```

```
/* Maximum number of concurrent transmission in the MLE.
   NOTE: This NOT a normative value, but used only in this validation model. */
SYNONYM MaxTransmissionIndexValue Natural = 15;
```

Process Data_transfer

2(25)

```
/* ** LOCAL VARIABLE DEFINITIONS **/
```

```
/** LOCAL VARIABLES **/
```

```
/* Variable containing the list of ongoing data  
transmissions from the MLE data transfer process. */
```

```
DCL  
    CurrTransmissions    TransmissionListType;
```

```
DCL  
    CellId           CellIdentifierType,  
    CR_Kind          CellReselectionType,  
    CommBroken       Boolean := FALSE,  
    CommClosed        Boolean := TRUE,  
/* ChgReq           Boolean,           True if Channel change request and group addressed PDU */  
    ProtocolKind     MLE_ProtocolDiscriminatorType,  
    L2Service         Layer2ServiceType,  
    NewCellInfo      NewCellInfoType,  
    NoOf_CONP_Trms   Natural,          /* Number of current CONP transmissions */  
    TrmElem          TransmissionDescrType,  
    TrmElemFound     Boolean,  
    TrfResult         Boolean,  
    RecReqs          Natural := 0,  
    ScrambleCode     ScramblingCodeType, /* For LLC service primitives. Defined via remote  
                                         procedure GetScramblingCode */  
    Current_EndpointID Endpoint_ID_Type,      /* current EndpointID value */  
    MLE_ProtoParms   DataReqInfoType,  
    VaddrL           ValidAddressesType;
```

```
DCL  
    TMI              TMI_Type,  
    ISSI             SSI_Type2,  
    ASSI             SSI_Type2,  
    GSSI_List        TSI_ListType,  
    Temp_GSSI_List  TSI_ListType,  
    dTadr            GSSI_Type2 := (. FALSE, 0 .); /* default temporary GSSI value */
```

```
DCL  
    SubscriClass     SubscriberClassType := ALL_SUBSCRIBERCLASS_ALLOWED,  
    CellSubscriClass SubscriberClassType := ALL_SUBSCRIBERCLASS_ALLOWED;
```

Process Data_transfer

3 (25)

```
/* *** LOCAL VARIABLE DEFINITIONS ***/
```

```
/** MLE PDU DATA VARIABLES **/
```

```
DCL  
    U_Prep_PDU MLE_U_PrepareType;
```

```
DCL  
    D_PrepFail_PDU MLE_D_PrepFailType,  
    D_NewCell_PDU MLE_D_NewCellType,  
    D_NwrkBroadc_PDU MLE_D_NWRK_BroadcastType,  
    D_SIN2_PDU MLE_D_SIN2Type,  
    D_SIN1_PDU MLE_D_SIN1Type; /* copy of system information originally received  
                                by the the Network Broadcast process */
```

```
DCL  
    DataTrf_PDU MLE_DataTransferPDUType;
```

```
/** MLE SP DATA VARIABLES **/
```

```
DCL  
    MLE_Cancel MLE_CancelType,  
    MLE_InfoReq MLE_InfoReqType,  
    MLE_MM_Identities MLE_MM_IdentitiesType,  
    MLE_MM_UnitdatReq MLE_MM_UnitDataReqType,  
    MLE_MSC_UnitdatInd MLE_MSC_UnitDataIndType,  
    MLE_ReportInd MLE_ReportIndType,  
    MLE_SC_UnitdatReq MLE_SC_UnitdataReqType;
```

Process Data_transfer

4 (25)

```
/* *** LOCAL VARIABLE DEFINITIONS ***/
```

```
/** LLC SP DATA VARIABLES **/
```

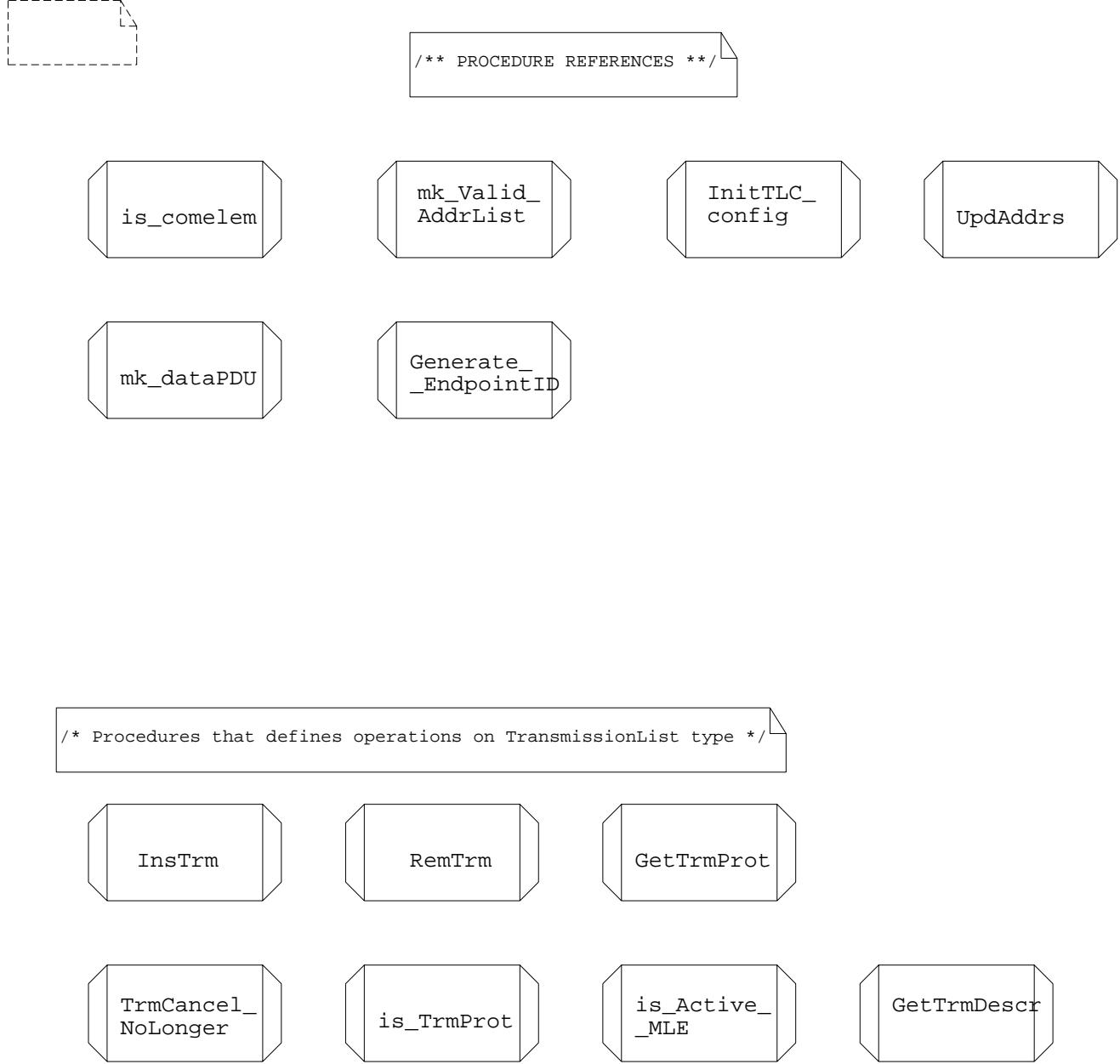
```
DCL  
      TLA_DatConf TLA_DataConfirmType,  
      TLA_UDatConf TLA_UnitdataConfirmType;
```

```
DCL  
      TLC_CfgReq  TLC_ConfigureRequestType,  
      TLC_CfgConf  TLC_ConfigureConfirmType;
```

```
/* Remote procedure to get scrambling code (MNI)  
   from process Attachment_management */  
  
IMPORTED PROCEDURE GetScramblingCode;  
RETURNS ScramblingCodeType;
```

Process Data_transfer

5 (25)



Process Data_transfer

6 (25)



/* *** EXPORT OF REMOTE PROCEDURES *** /

/* The exported procedure returns
the endpoint id value to the
TLLAB_Formatter */

/* Exported procedure to transfer additional
information to the formatter when sending
MLE PDUS via the LLC Service Interface
*/

EXPORTED
GetEndpointID

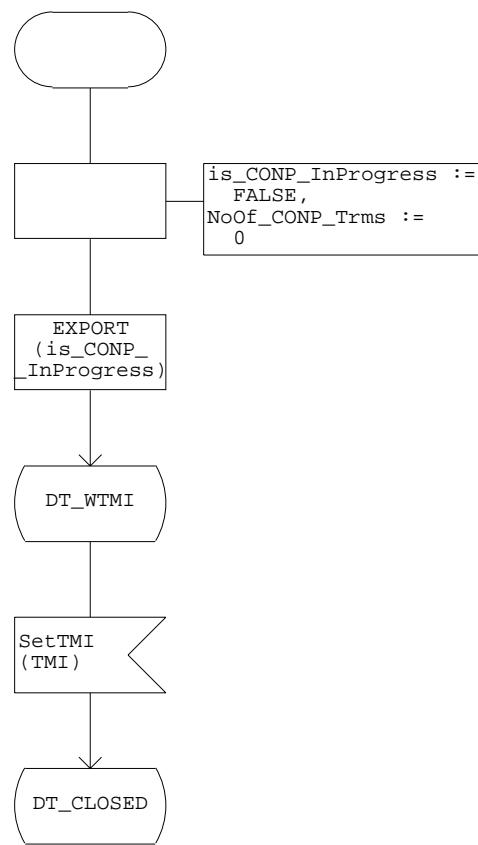
EXPORTED
GetSend_
DataParms

/* Exported variable to transfer
information if there are any CONP
transfer in progress. This
information is used by the
Attachment management process.
*/

DCL
EXPORTED is_CONP_InProgress Boolean;

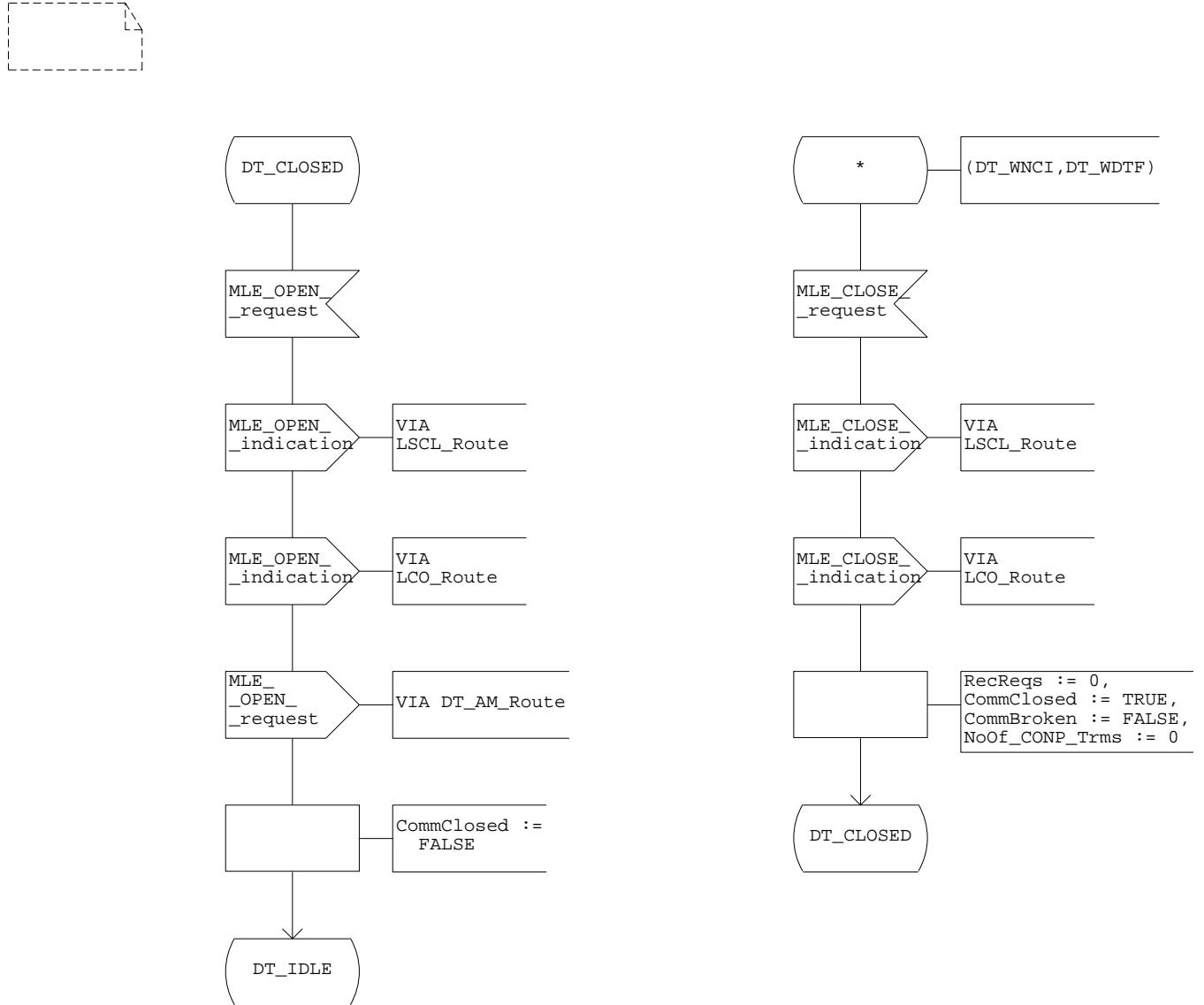
Process Data_transfer

7 (25)



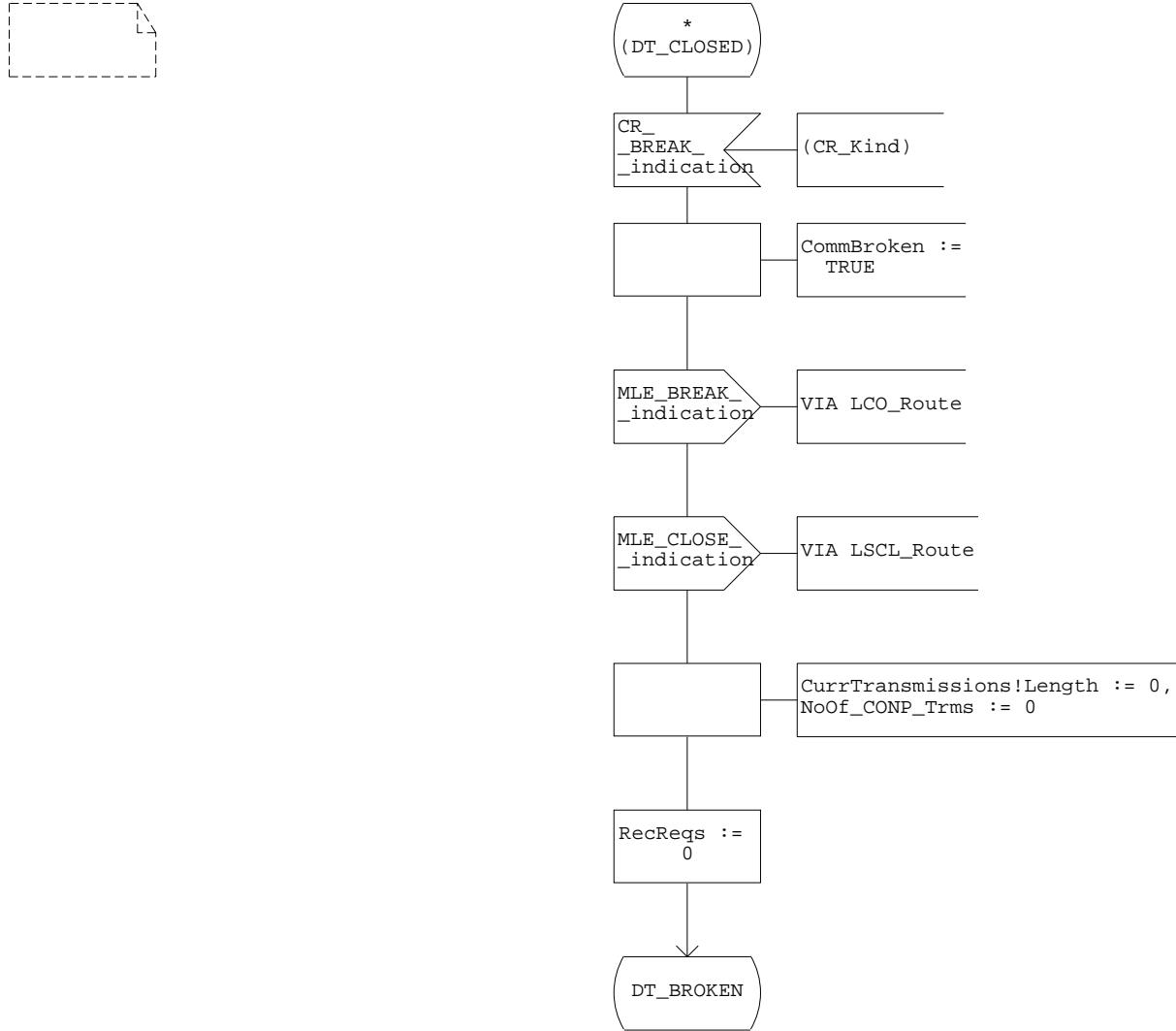
Process Data_transfer

8 (25)



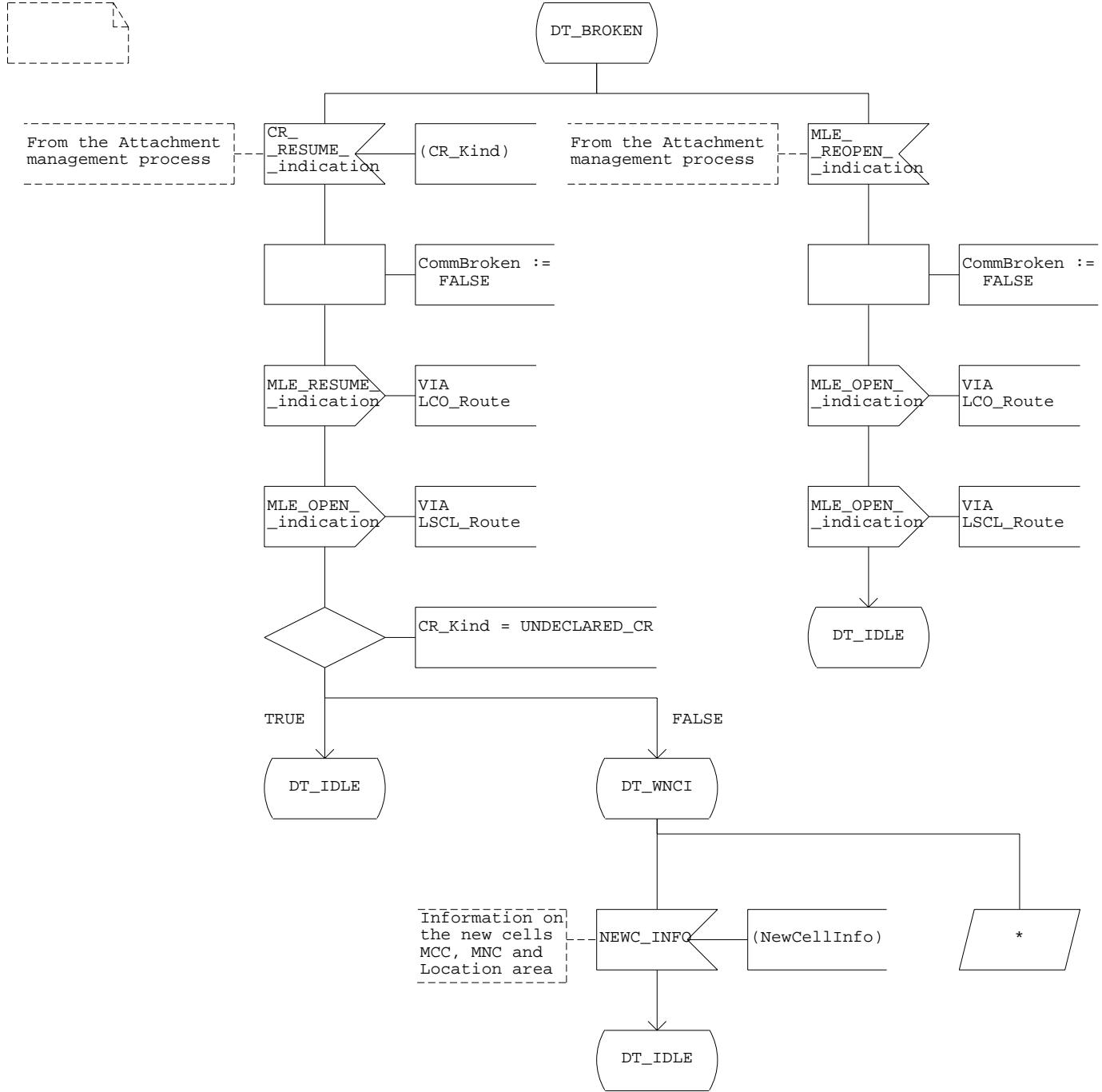
Process Data_transfer

9 (25)



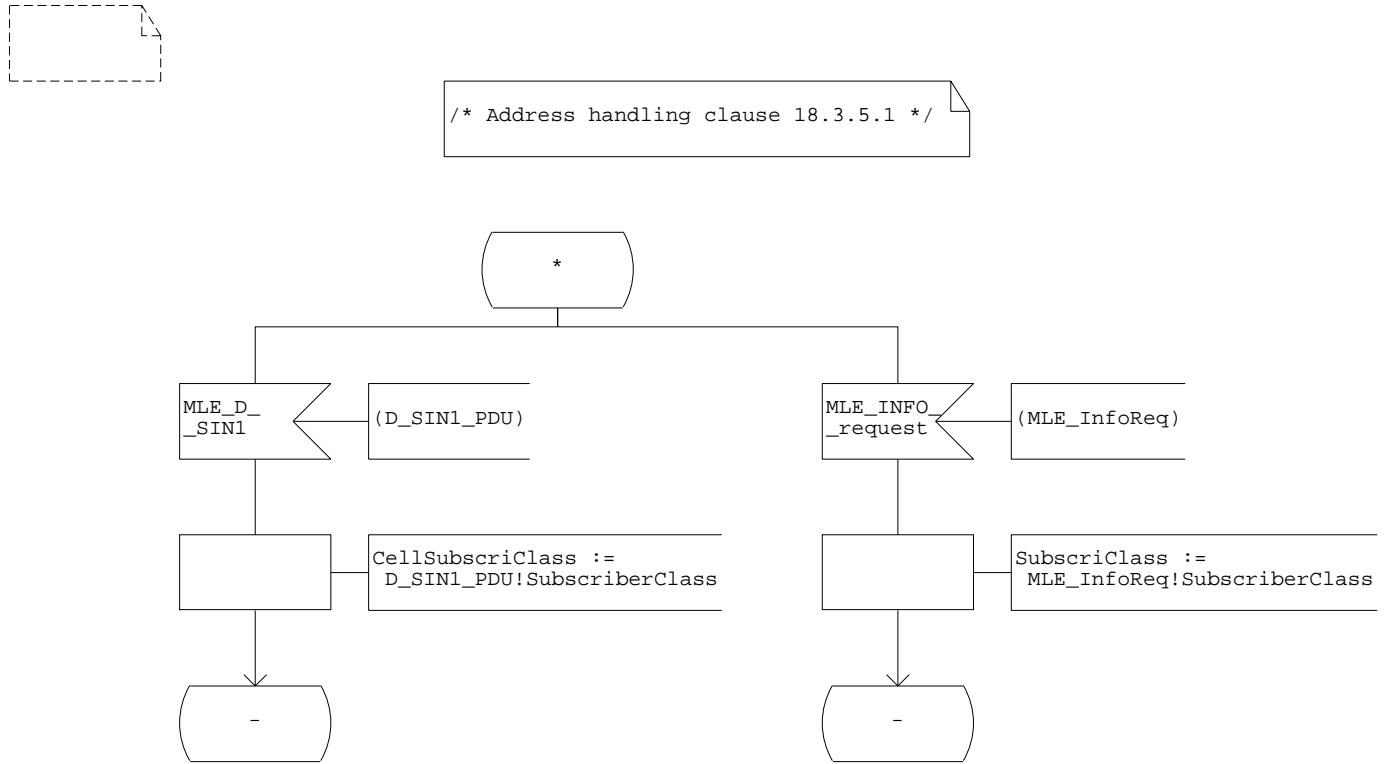
Process Data_transfer

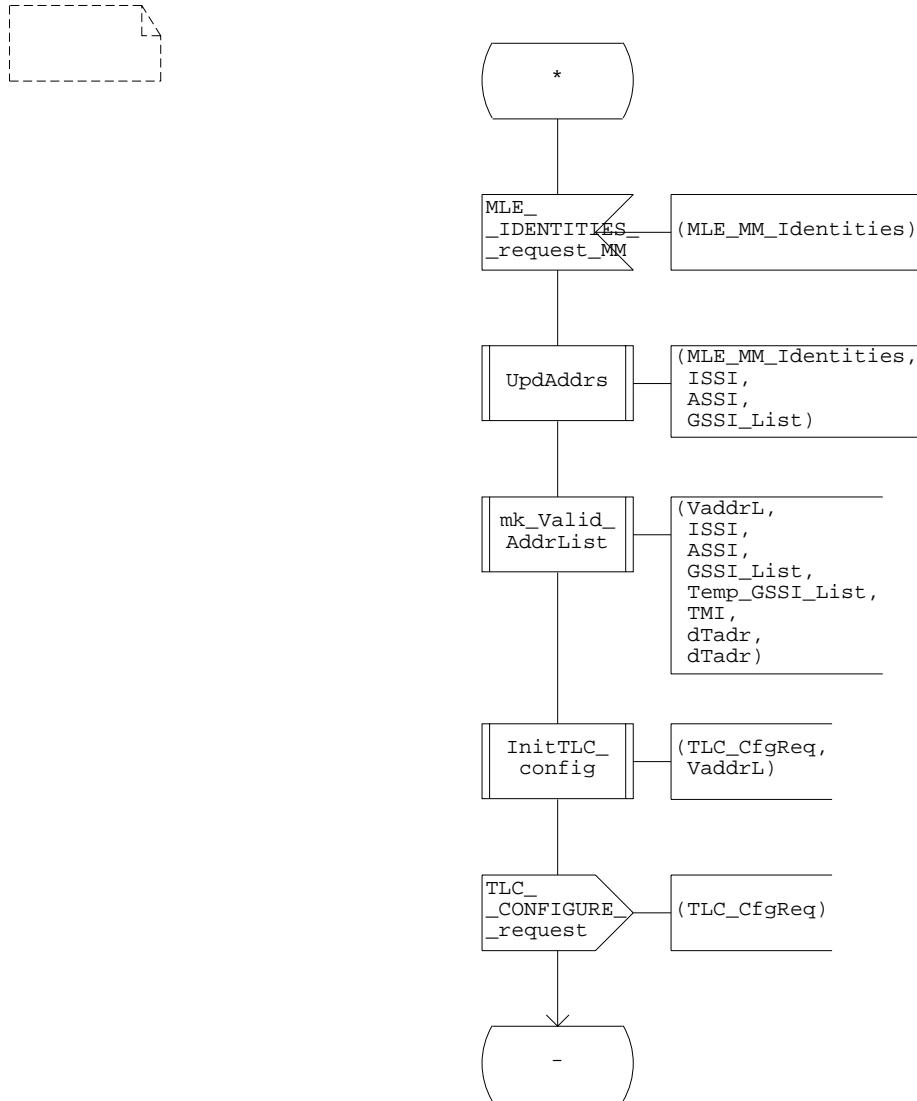
10 (25)



Process Data_transfer

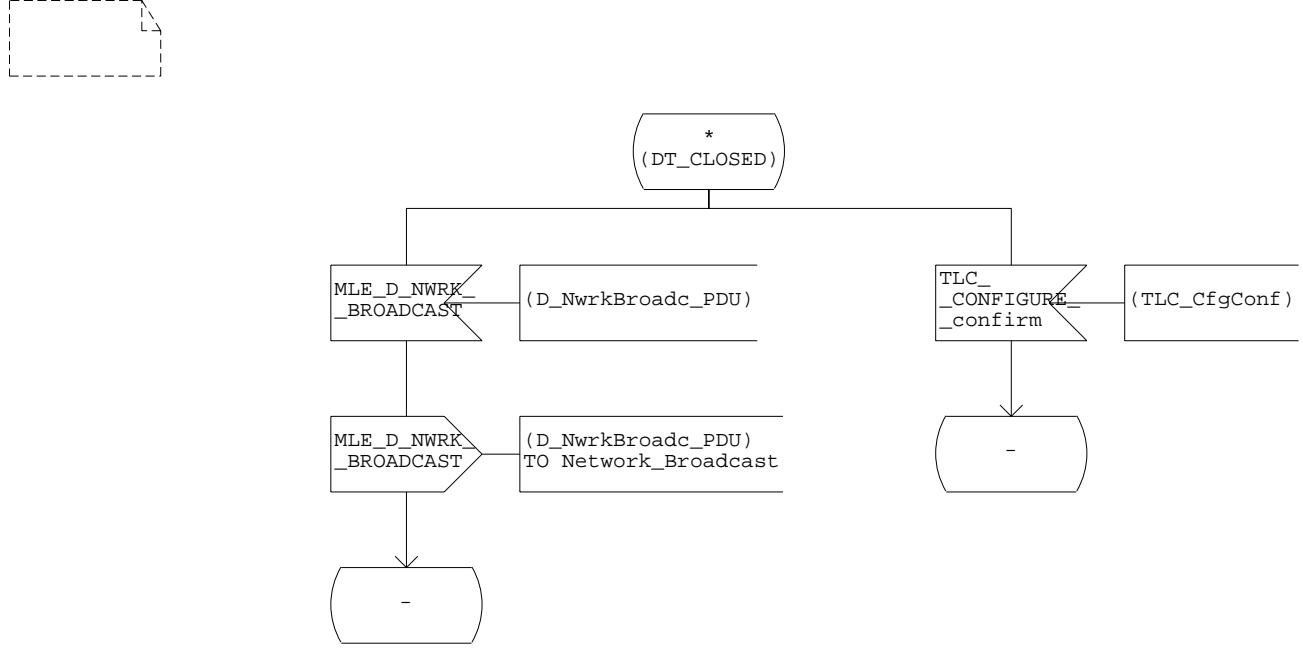
11(25)

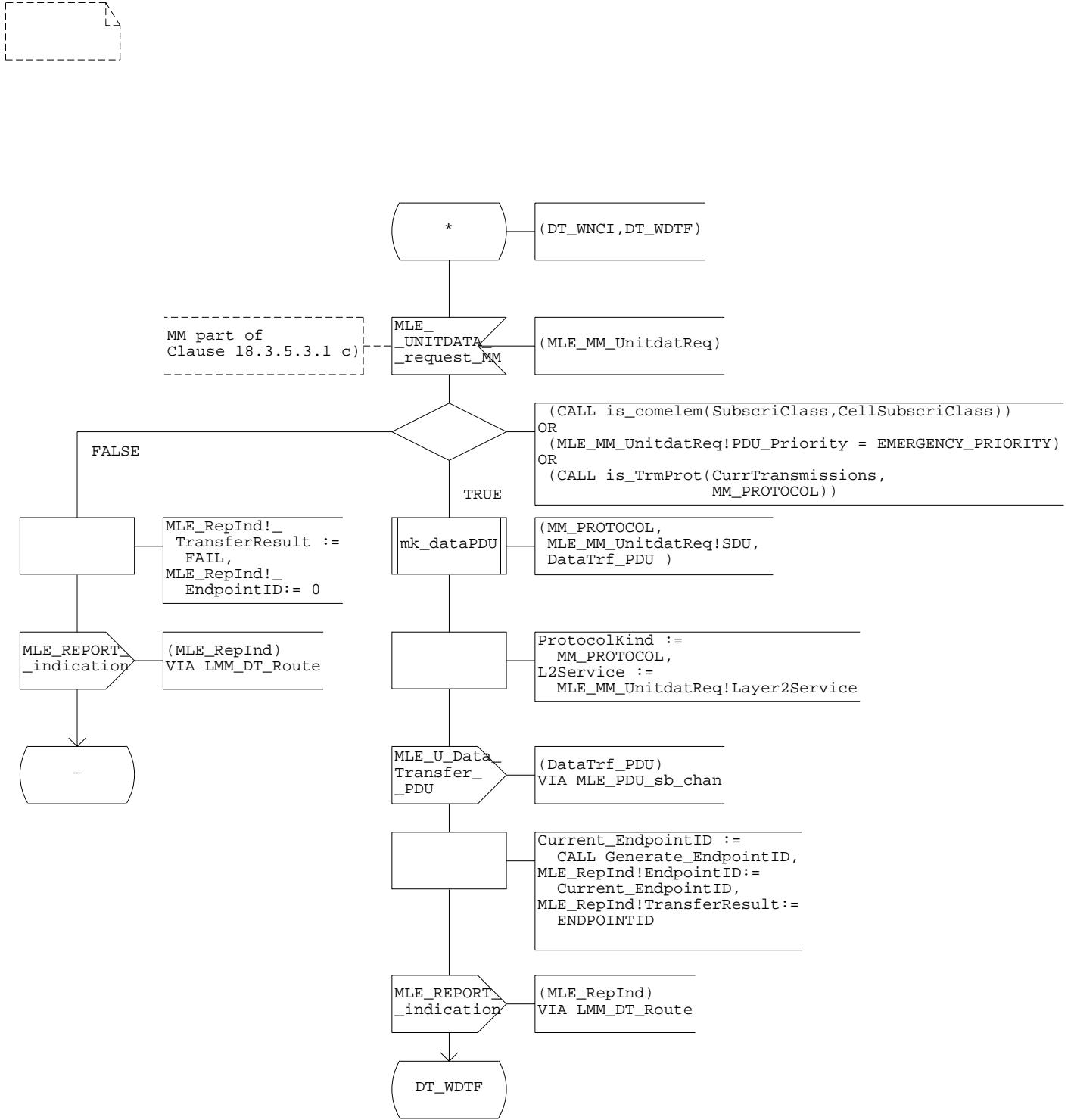


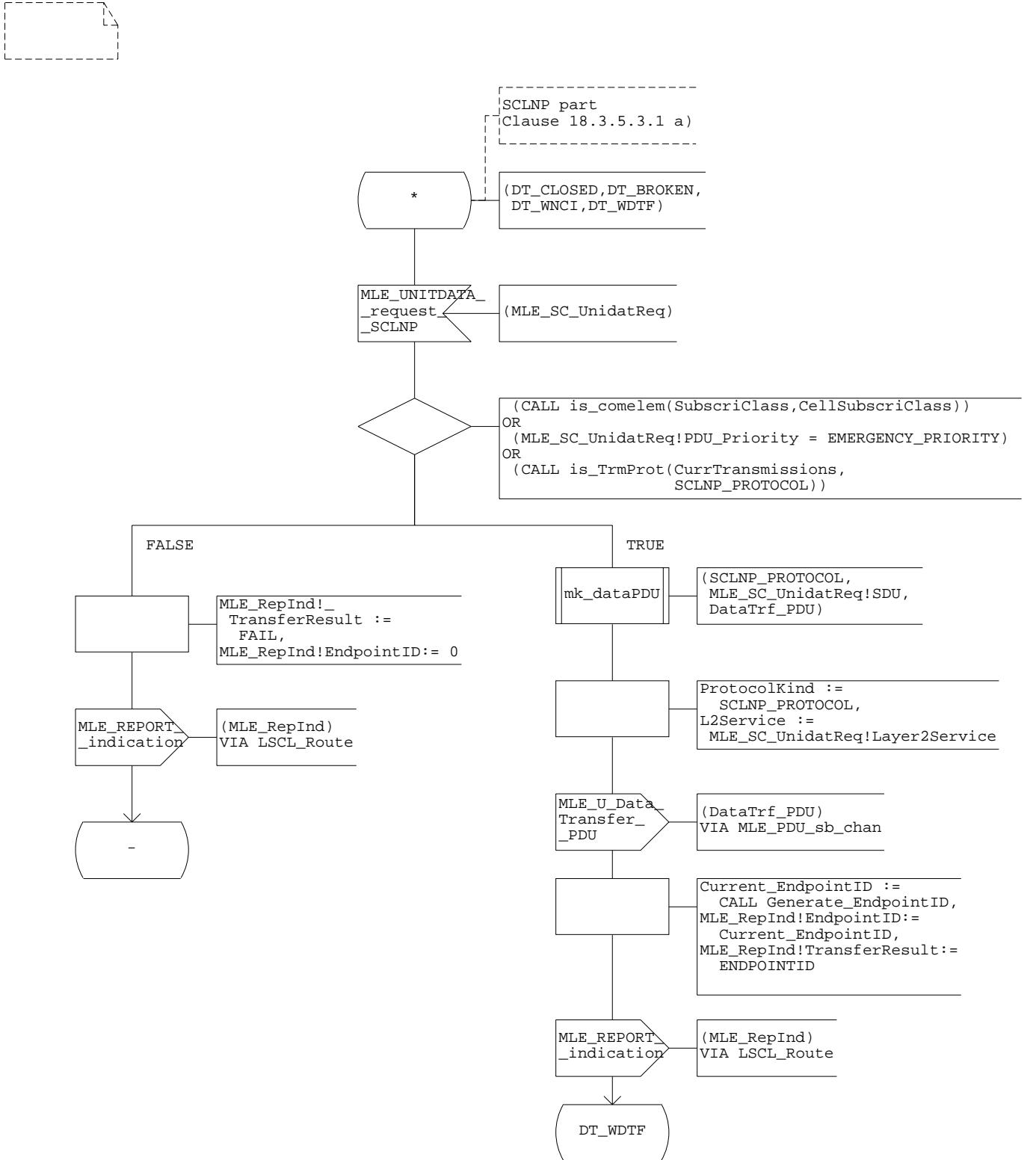


Process Data_transfer

13 (25)

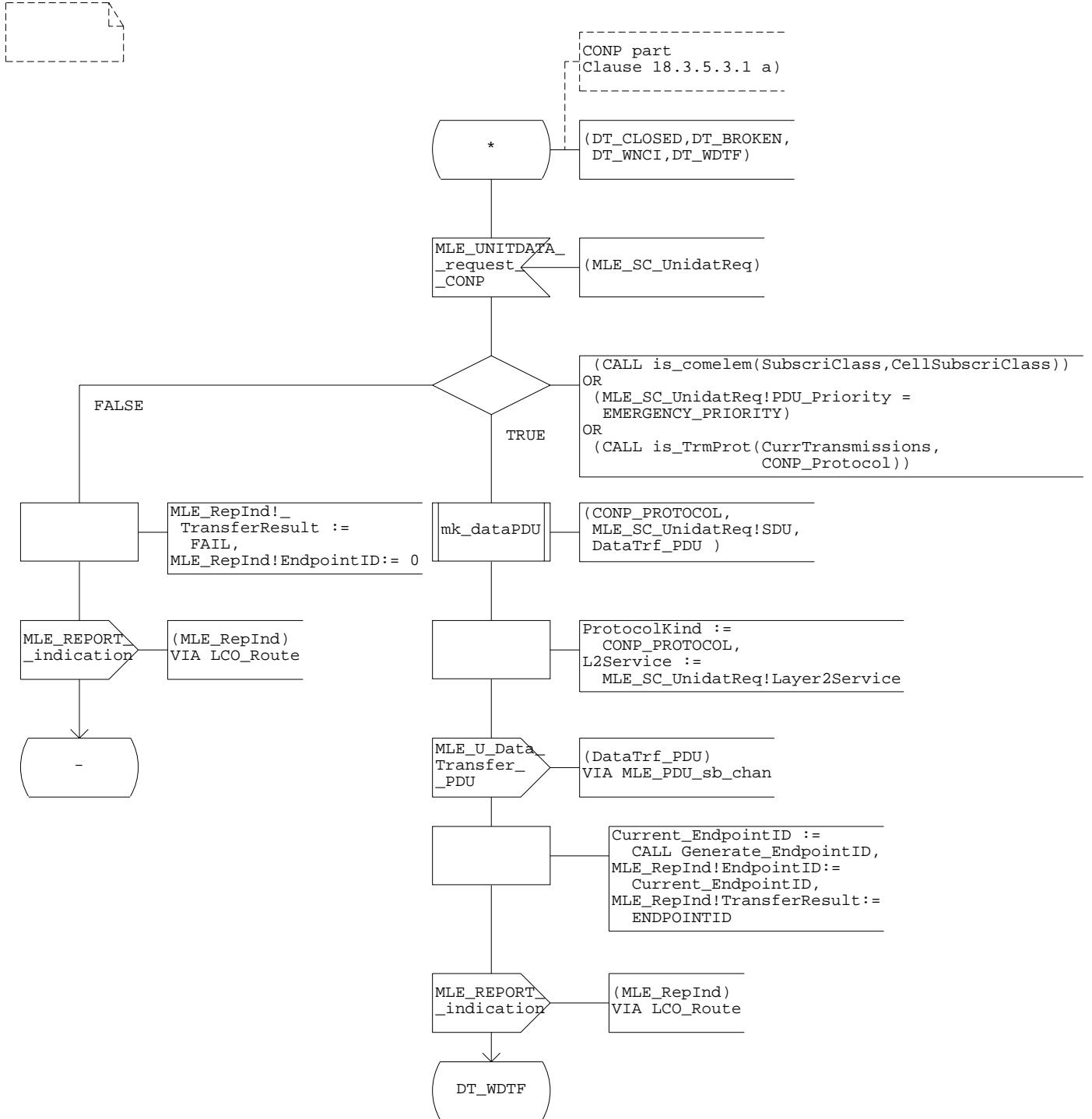


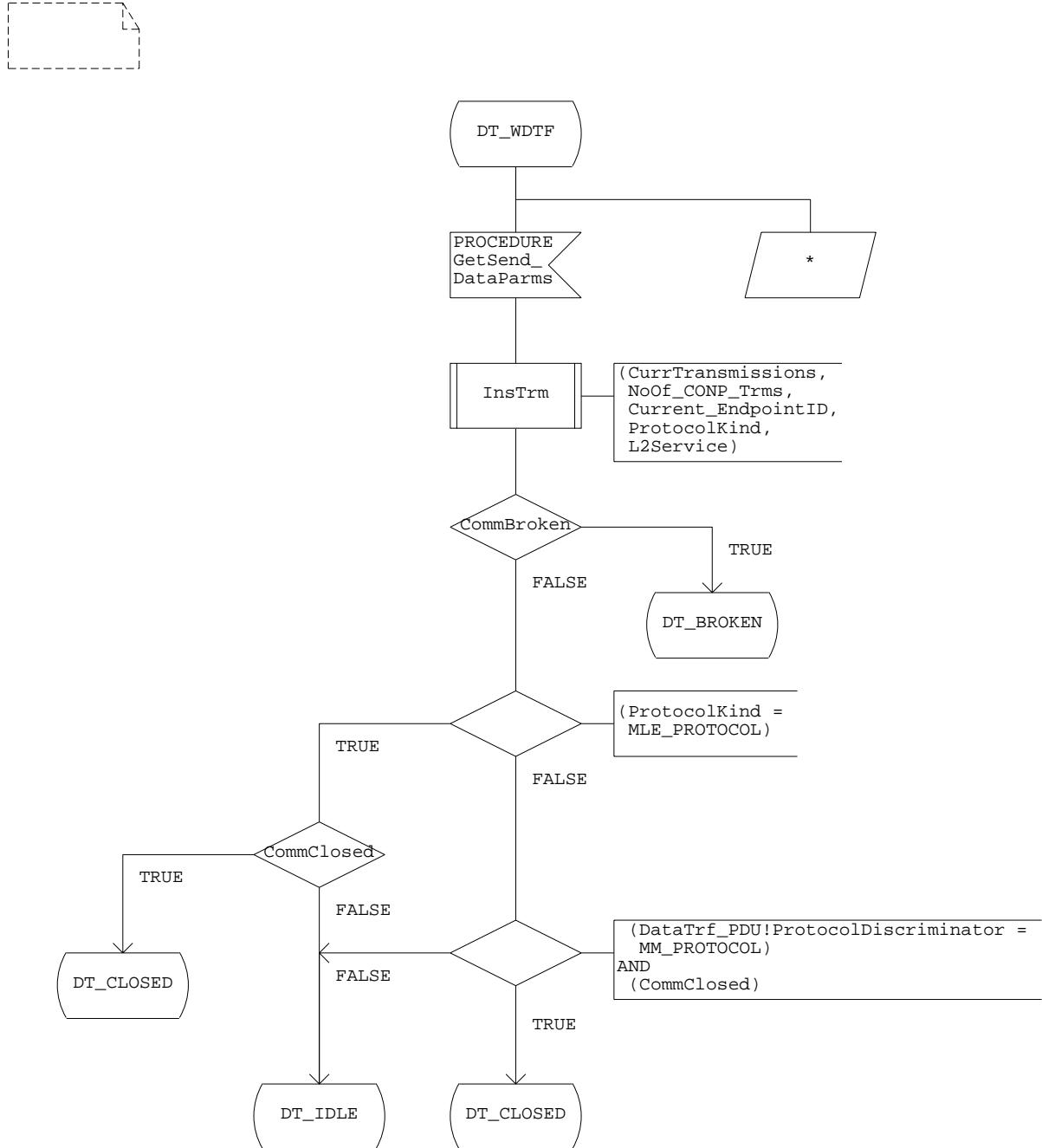


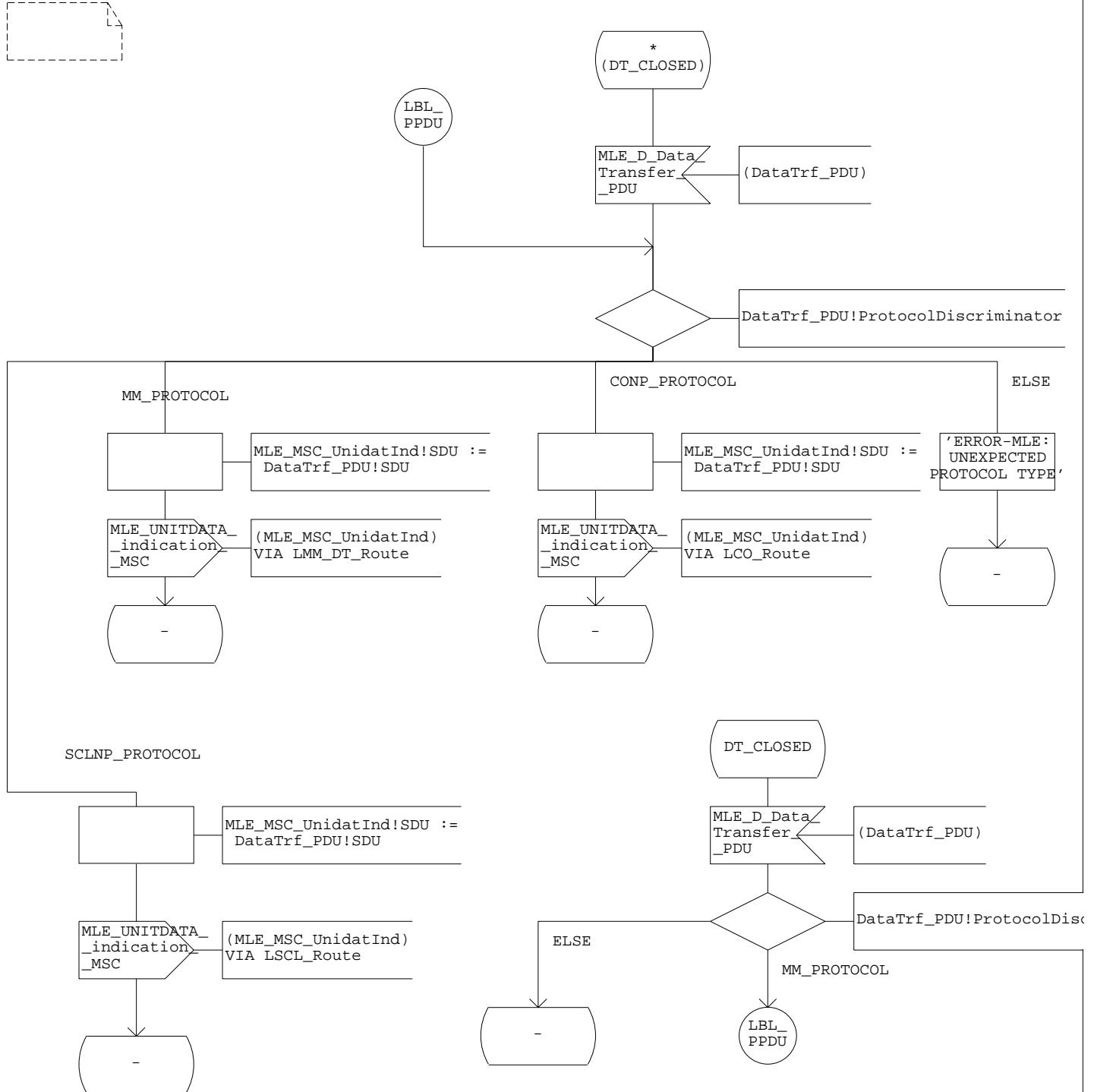


Process Data_transfer

16 (25)

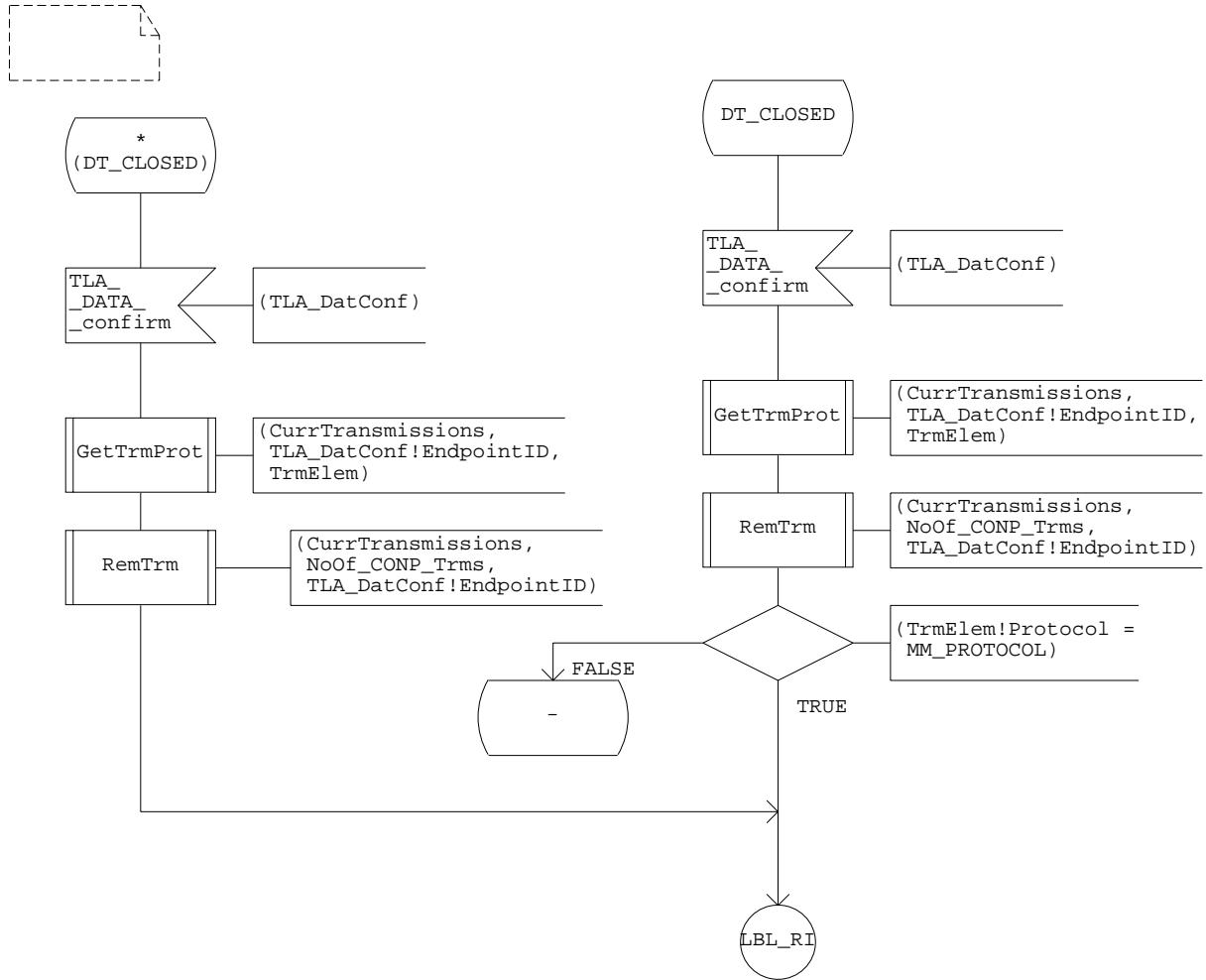






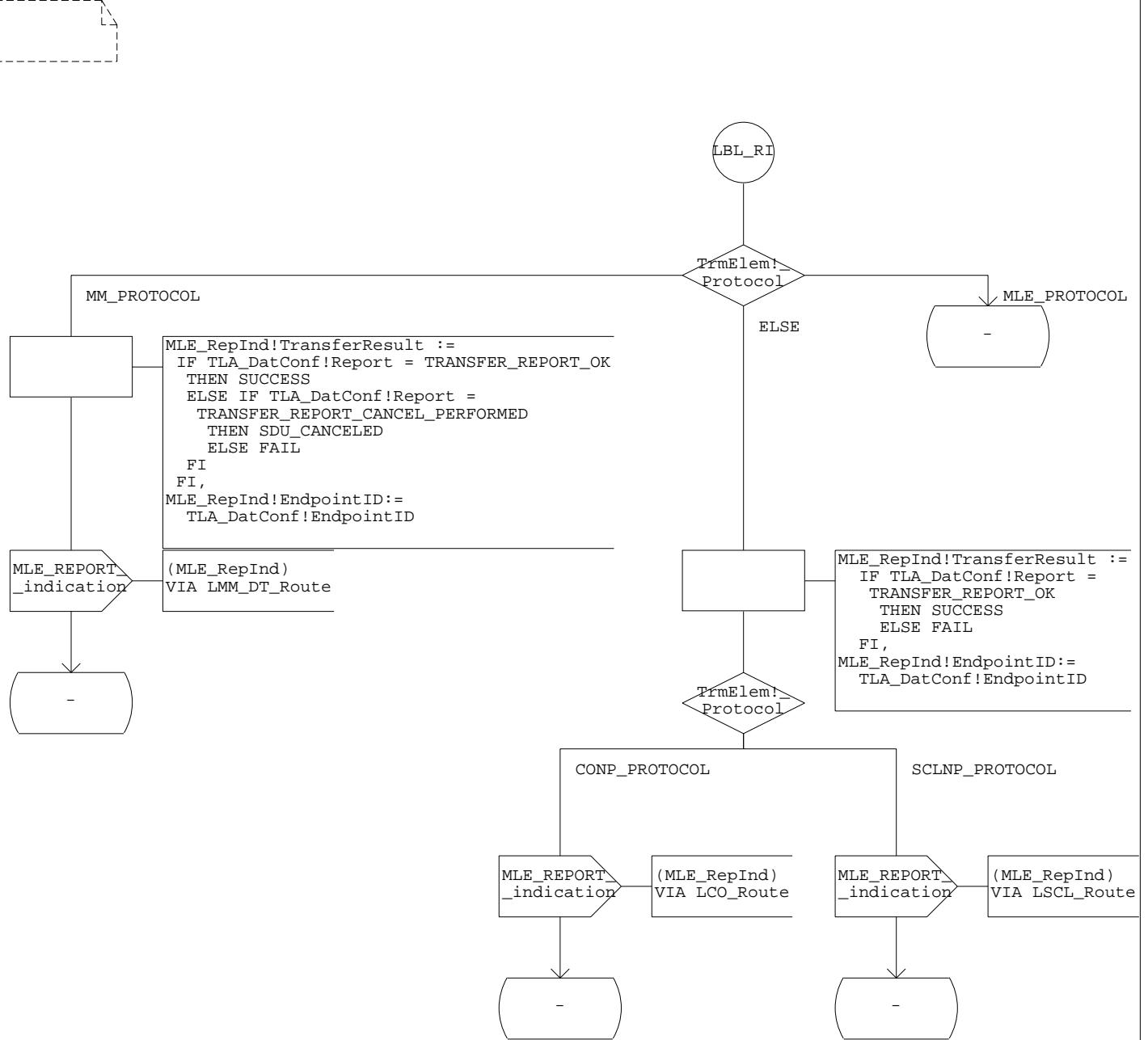
Process Data_transfer

19 (25)



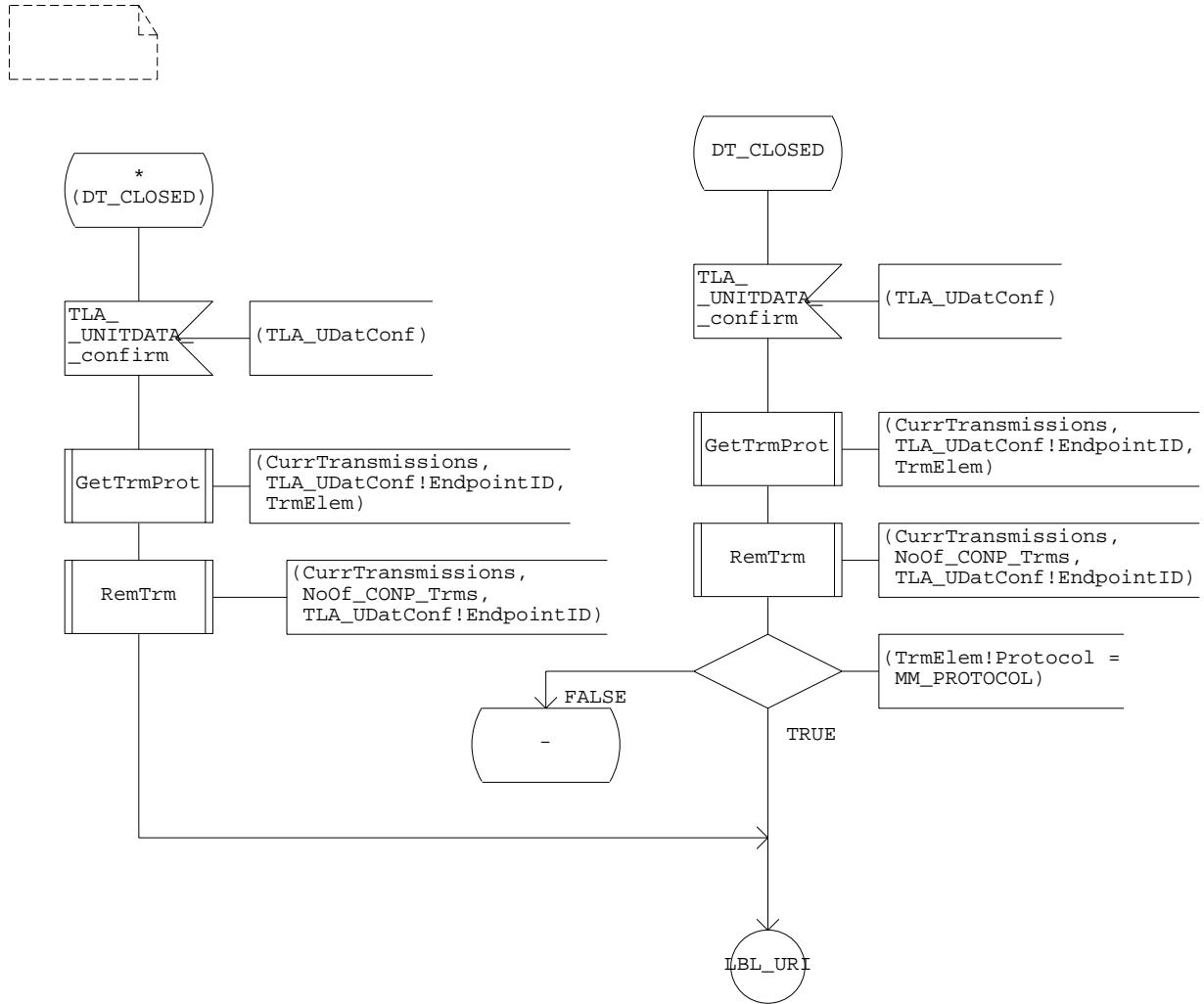
Process Data_transfer

20 (25)



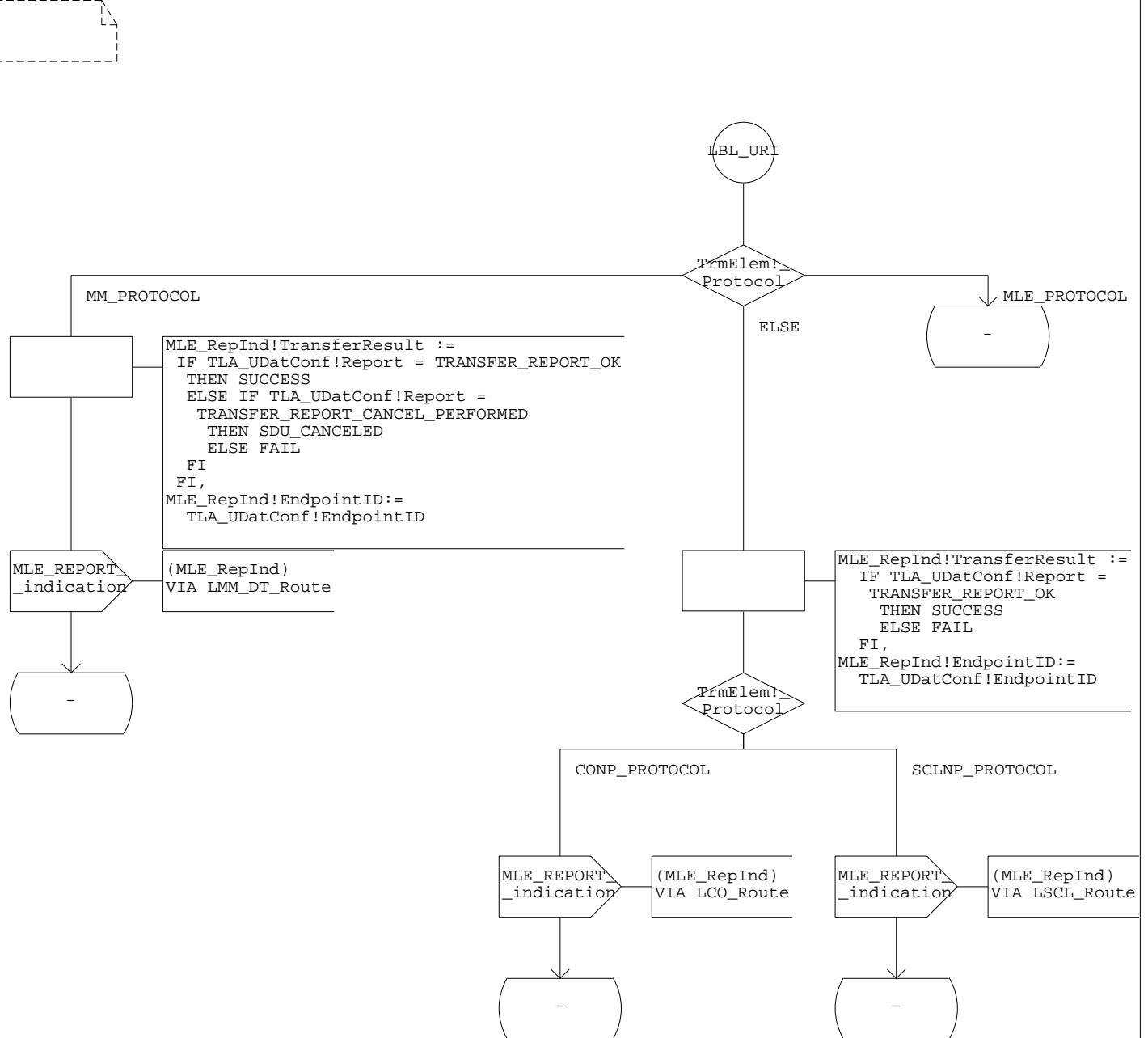
Process Data_transfer

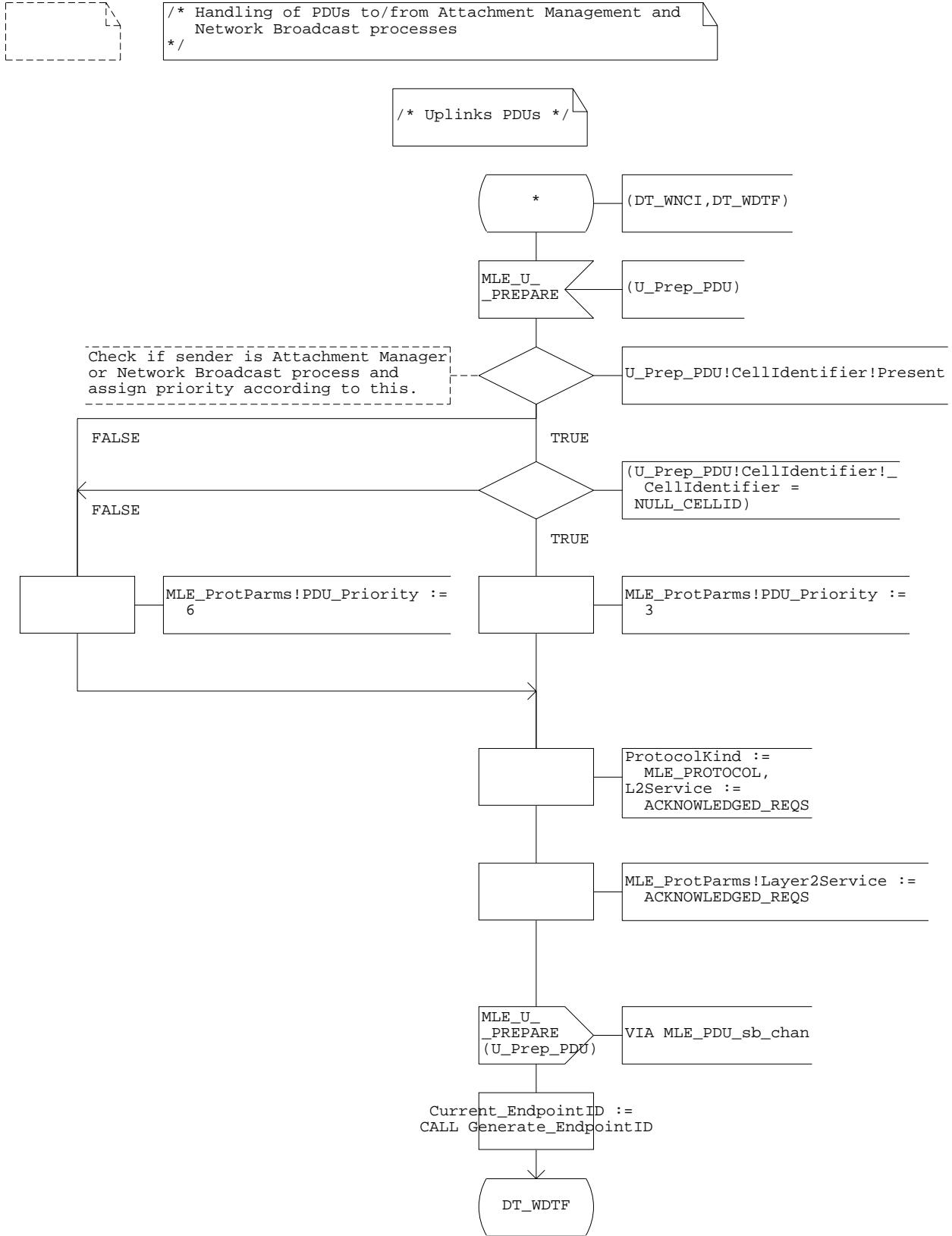
21 (25)

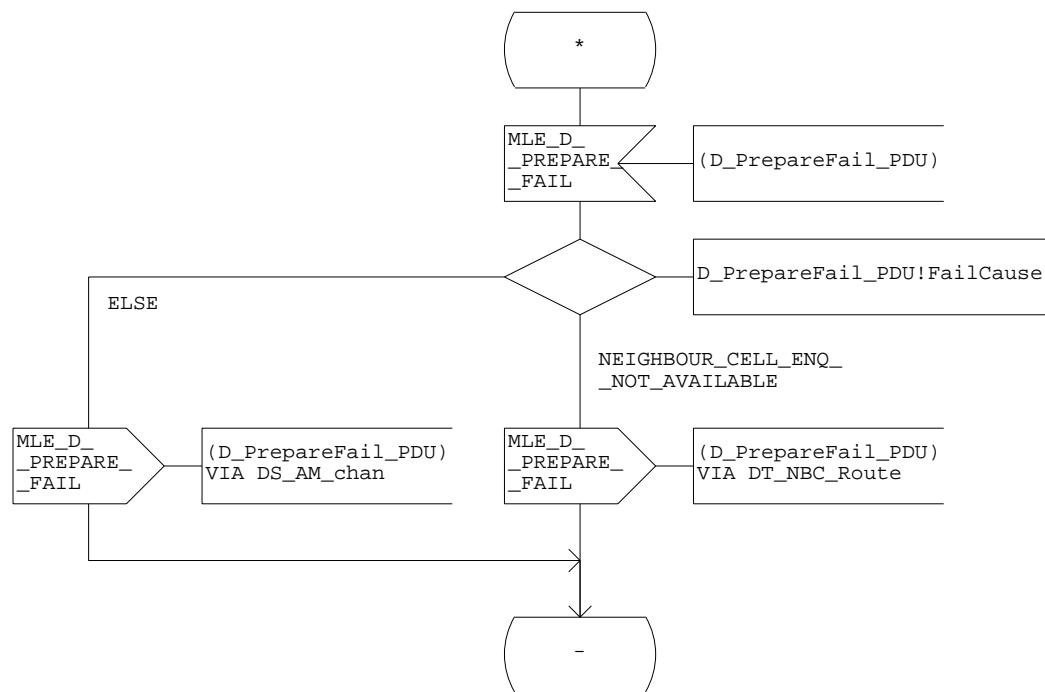


Process Data_transfer

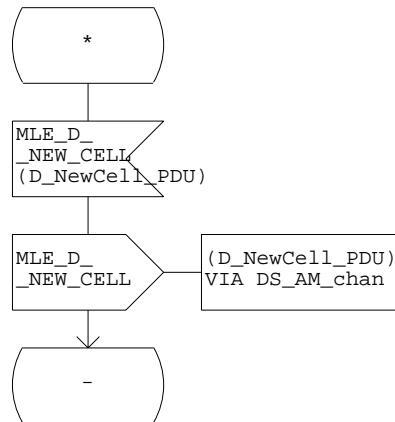
22 (25)





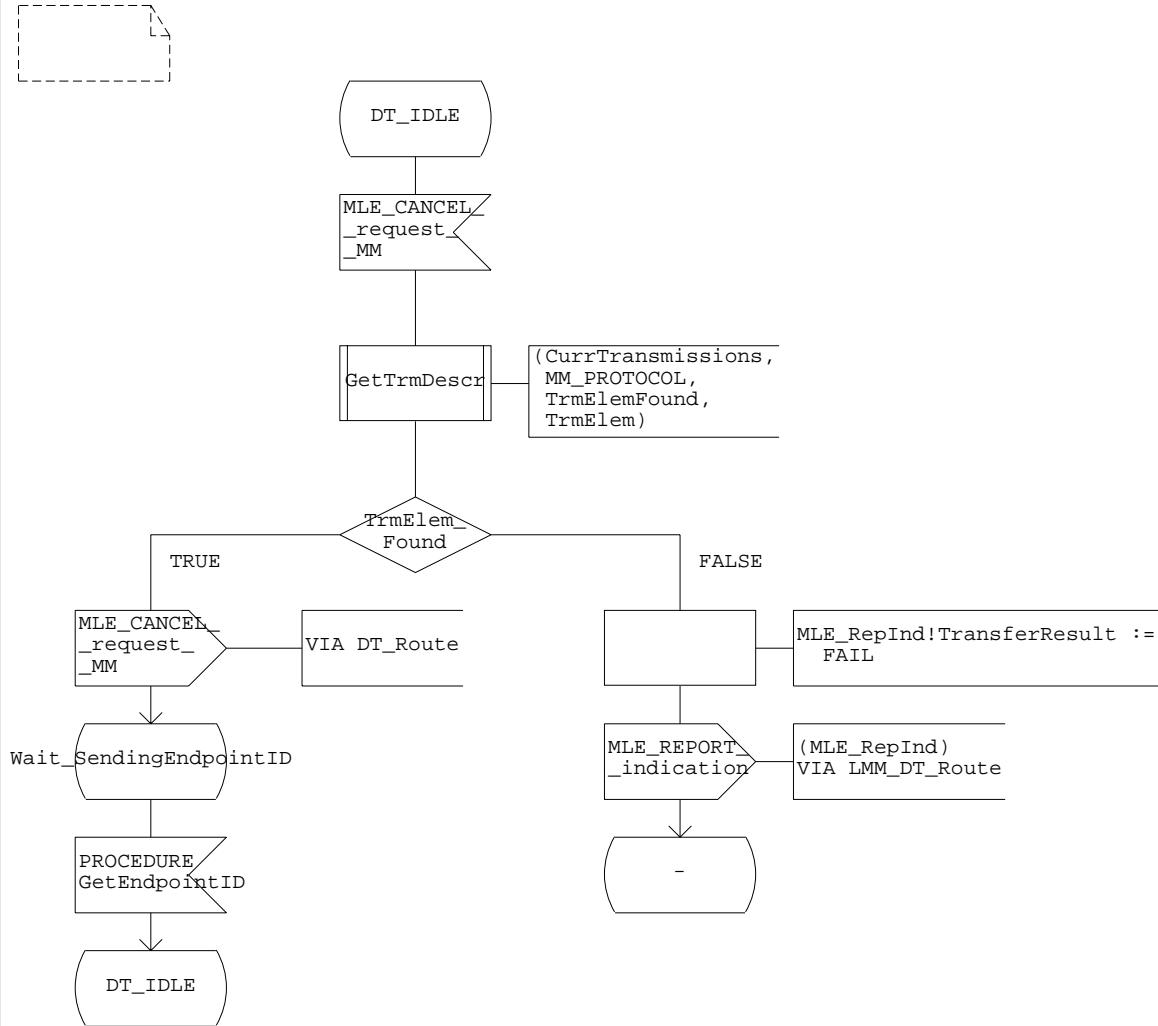


```
/* Transfer NEW CELL
   PDU to attachment
   process
*/
```



Process Data_transfer

25 (25)



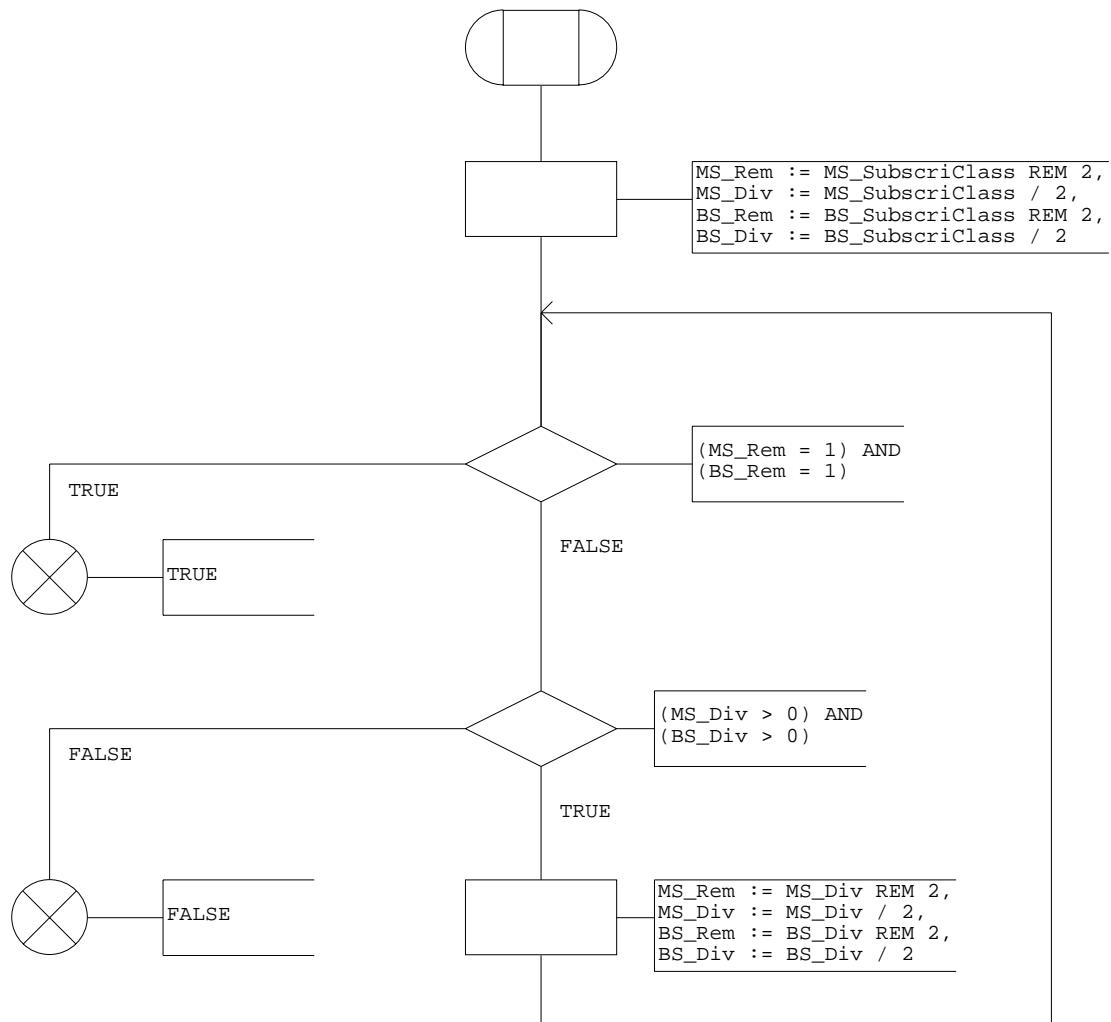
Procedure is_comelem

1(1)

```
; FPAR
  IN MS_SubscriClass SubscriberClassType;
  IN BS_SubscriClass SubscriberClassType;
 RETURNS Boolean;
```

```
/* Check if the MS has any subscriber class common with
   the subscriber classes supported by the current serving
   cell and if so return TRUE.
 */
```

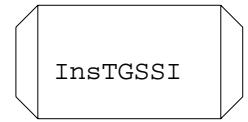
```
DCL
  MS_Rem Integer,
  MS_Div Integer,
  BS_Rem Integer,
  BS_Div Integer;
```



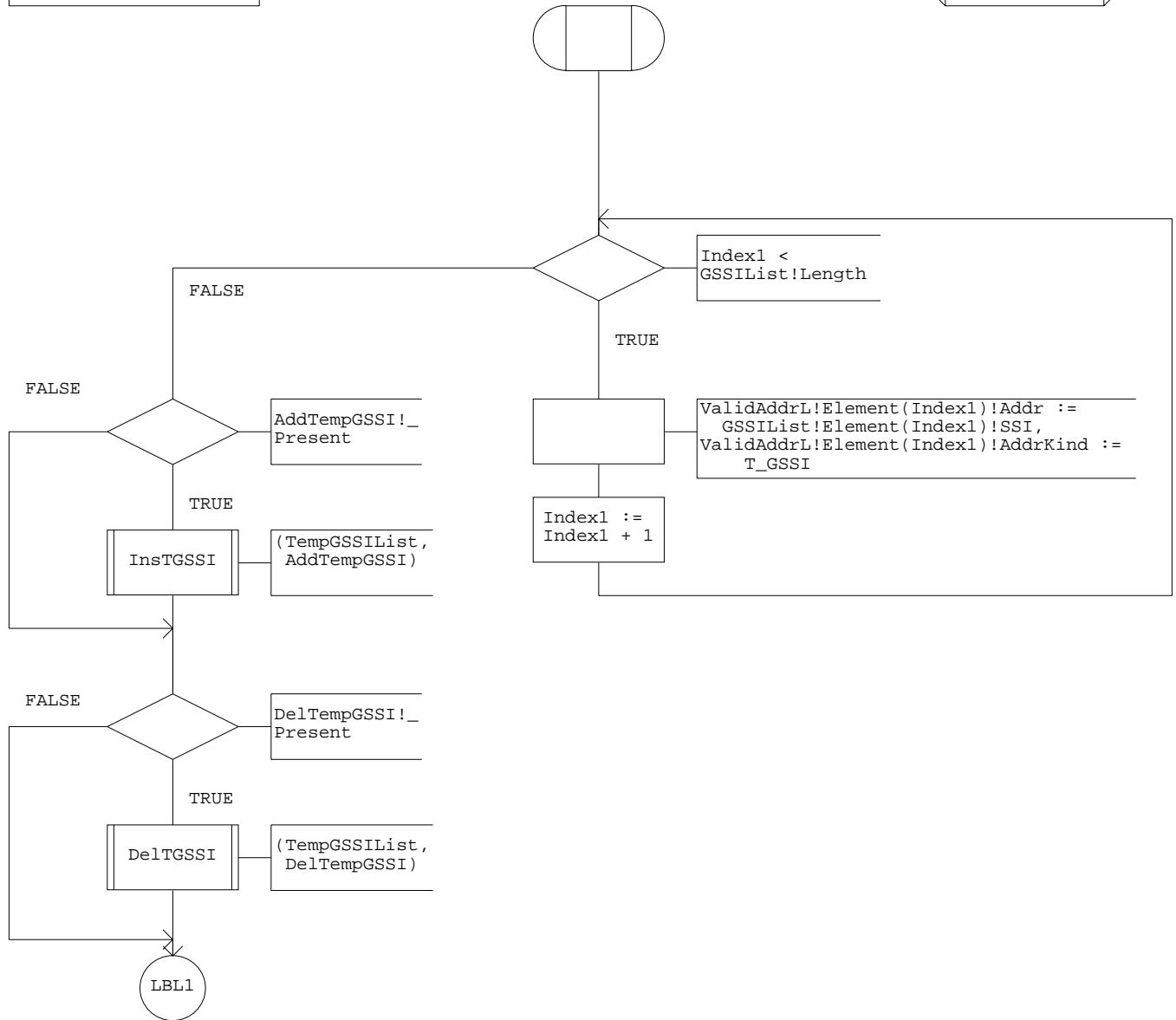
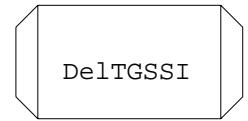
Procedure <>Process Data_transfer>> mk_ValidAddrList

1 (4)

```
; FPAR
IN/OUT ValidAddrL ValidAddressesType,
IN ISSI SSI_Type2,
IN ASSI SSI_Type2,
IN GSSIList TSI_ListType,
IN/OUT TempGSSIList TSI_ListType,
IN TMI TMI_Type,
IN AddTempGSSI GSSI_Type2,
IN DelTempGSSI GSSI_Type2;
```



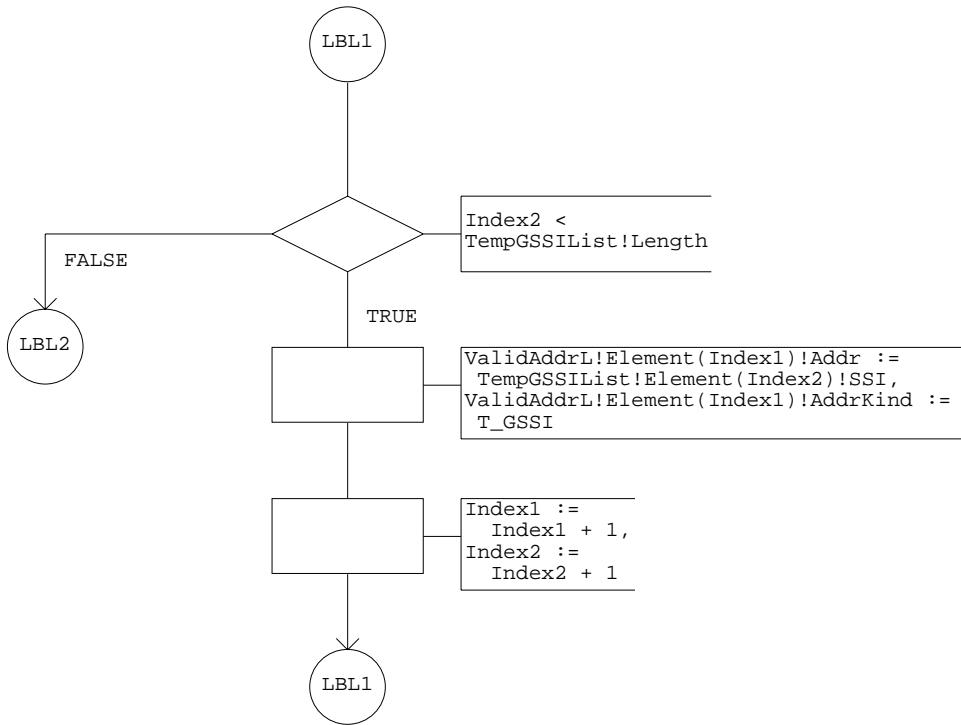
```
DCL
Index1 Natural := 0;
Index2 Natural := 0;
```



Procedure <>Process Data_transfer>> mk_ValidAddrList

2(4)

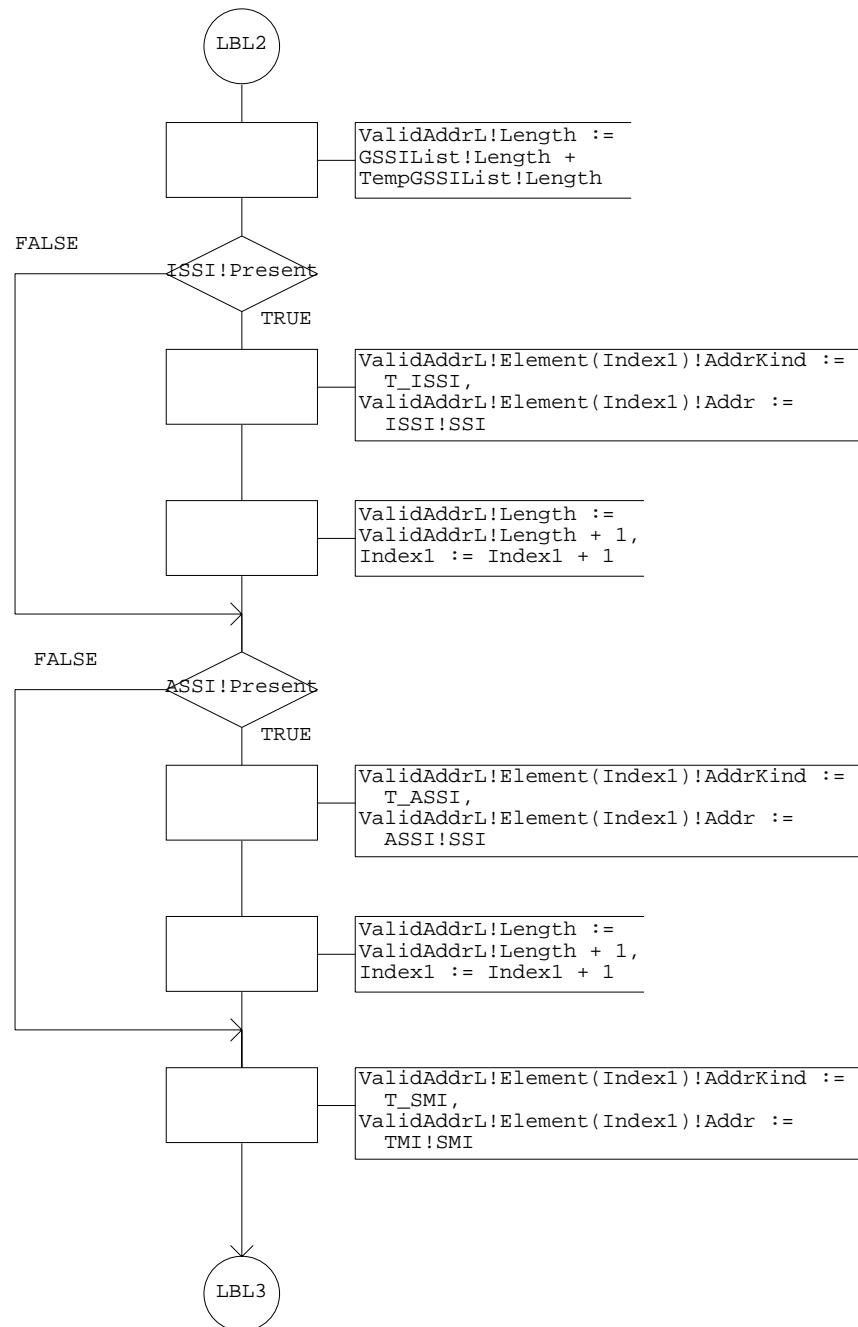
```
; FPAR
  IN/OUT ValidAddrL ValidAddressesType,
  IN ISSI SSI_Type2,
  IN ASSI SSI_Type2,
  IN GSSIList TSI_ListType,
  IN/OUT TempGSSIList TSI_ListType,
  IN TMI TMI_Type,
  IN AddTempGSSI GSSI_Type2,
  IN DelTempGSSI GSSI_Type2;
```



Procedure <<Process Data_transfer>> mk_ValidAddrList

3(4)

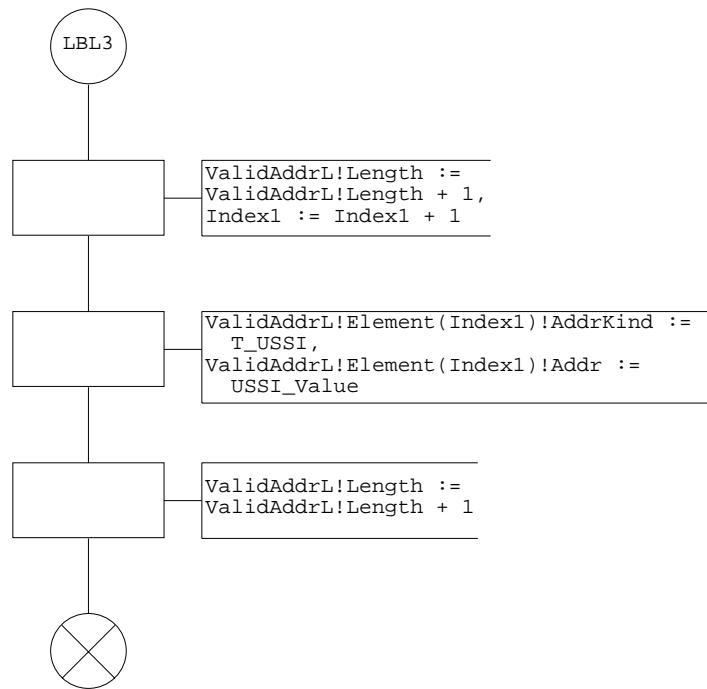
```
; FPAR
  IN/OUT ValidAddrL ValidAddressesType,
  IN ISSI SSI_Type2,
  IN ASSI SSI_Type2,
  IN GSSIList TSI_ListType,
  IN/OUT TempGSSIList TSI_ListType,
  IN TMI TMI_Type,
  IN AddTempGSSI GSSI_Type2,
  IN DelTempGSSI GSSI_Type2;
```



Procedure <>Process Data_transfer>> mk_ValidAddrList

4(4)

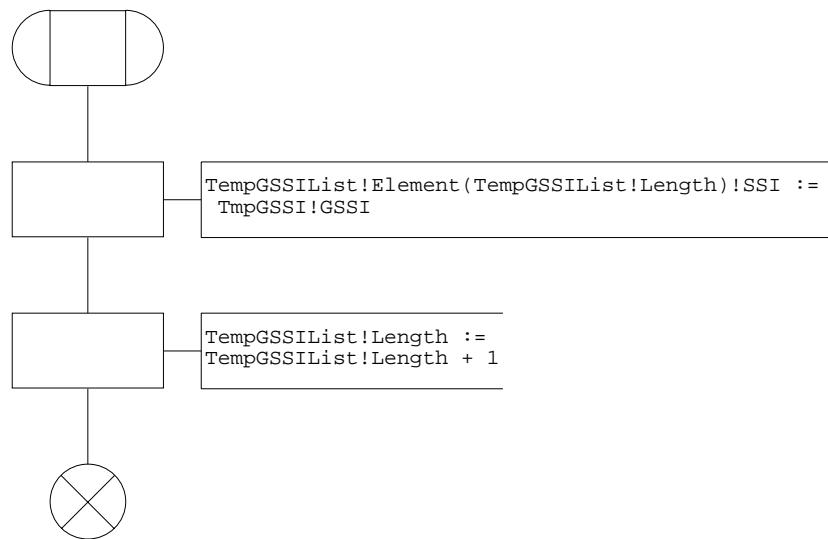
```
; FPAR
IN/OUT ValidAddrL ValidAddressesType,
IN ISSI SSI_Type2,
IN ASSI SSI_Type2,
IN GSSIList TSI_ListType,
IN/OUT TempGSSIList TSI_ListType,
IN TMI TMI_Type,
IN AddTempGSSI GSSI_Type2,
IN DelTempGSSI GSSI_Type2;
```



```
Procedure <<Process Data_transfer/Procedure mk_ValidAddrList>> InstTGSSI
```

```
1(1)
```

```
; FPAR  
IN/OUT TempGSSIList TSI_ListType;  
IN TmpGSSI GSSI_Type2;
```

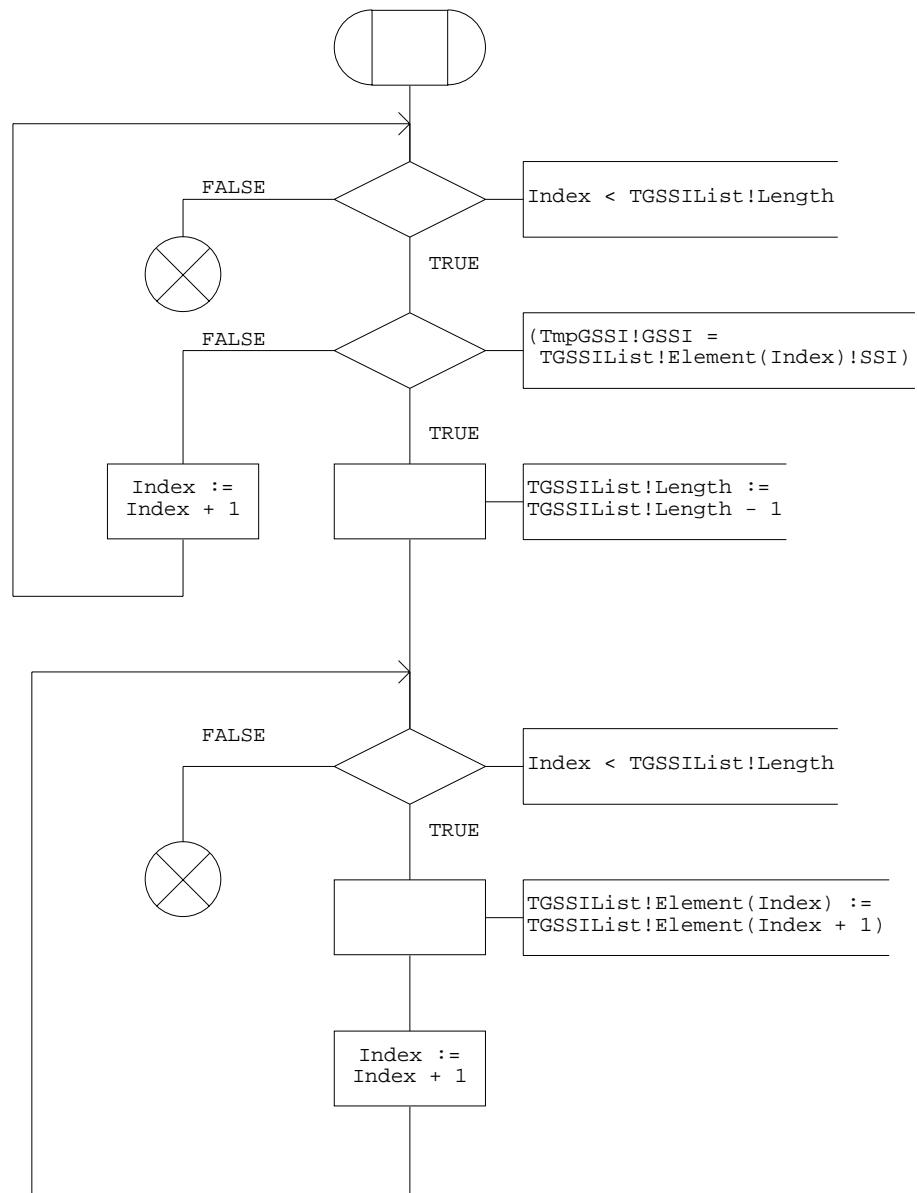


Procedure <>Process Data_transfer/Procedure mk_ValidAddrList>> DelTGSSI

1(1)

```
; FPAR  
IN/OUT TGSSIList TSI_ListType,  
IN TmpGSSI GSSI_Type2;
```

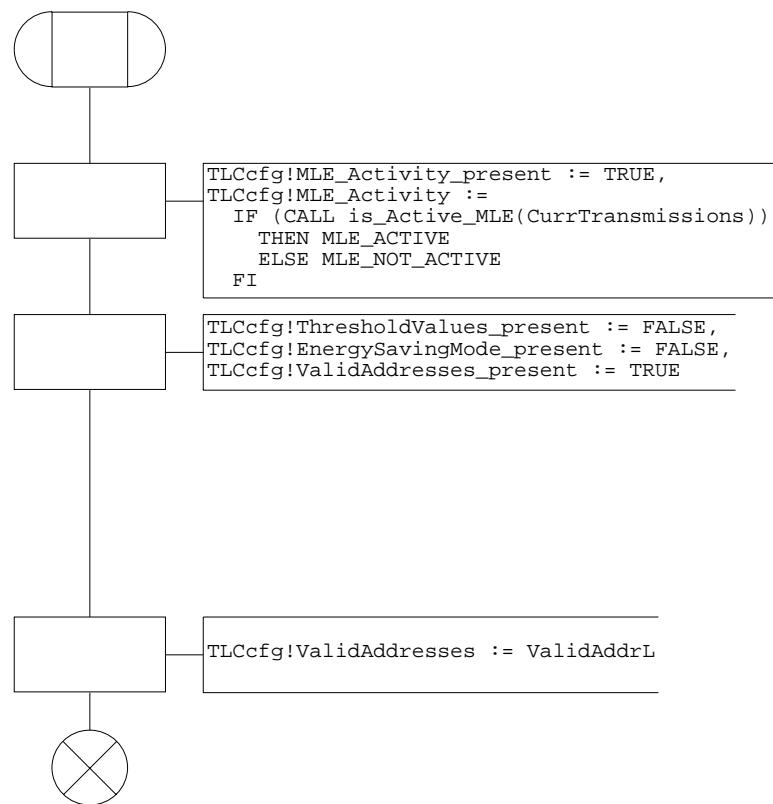
```
DCL  
Index Natural := 0;
```



Procedure InitTLCconfig

1(1)

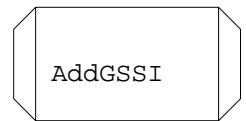
```
; FPAR
  IN/OUT TLCcfg  TLC_ConfigureRequestType;
  IN ValidAddrL  ValidAddressesType;
```



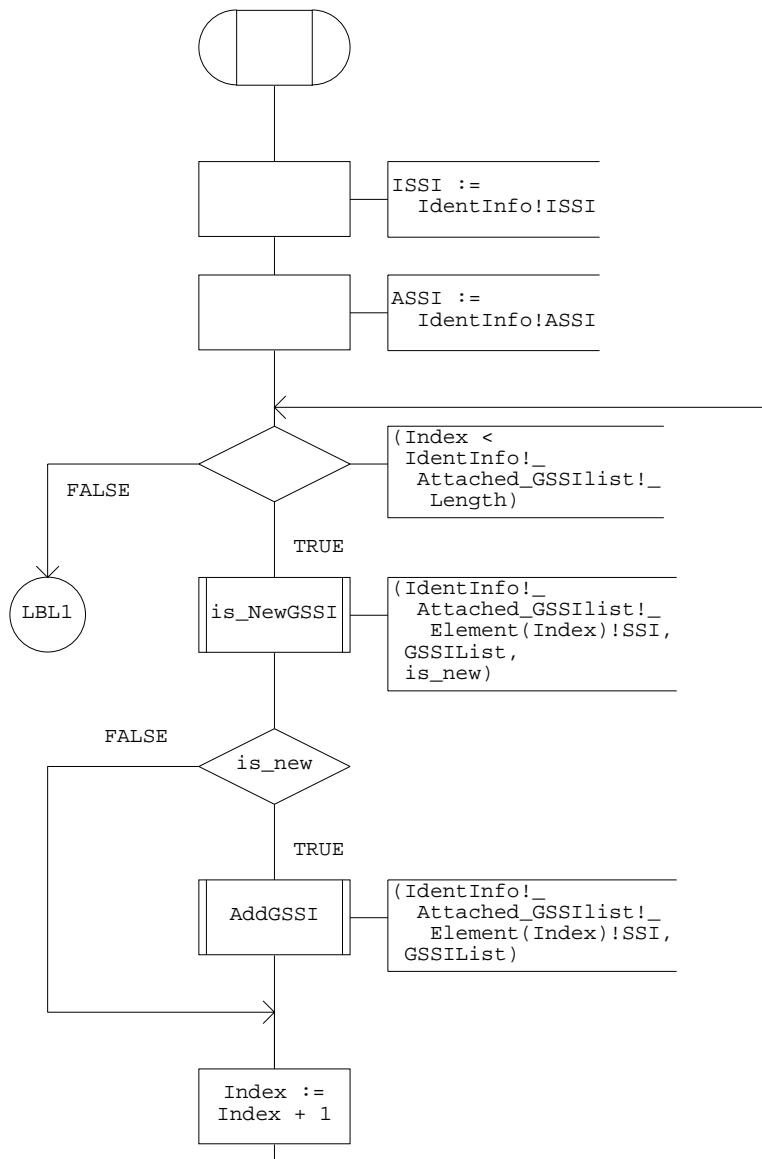
Procedure <<Process Data_transfer>> UpdAddrs

1 (2)

```
; FPAR
  IN IdentInfo MLE_MM_IdentitiesType;
  IN/OUT ISSI SSI_Type2,
  IN/OUT ASSI SSI_Type2,
  IN/OUT GSSIList TSI_ListType;
```



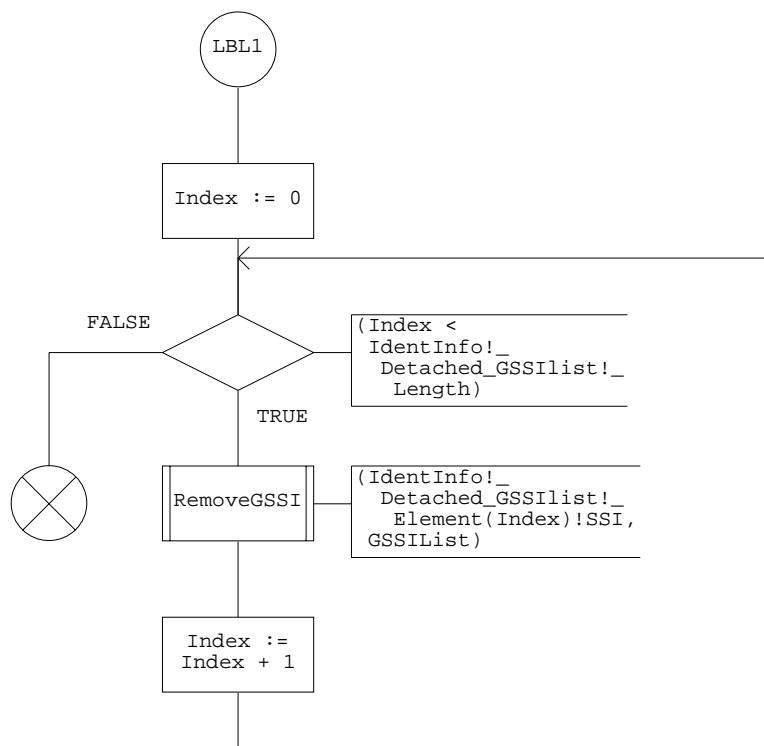
```
DCL
  Index Natural := 0,
  is_new Boolean;
```



Procedure <<Process_Data_transfer>> UpdAddrs

2(2)

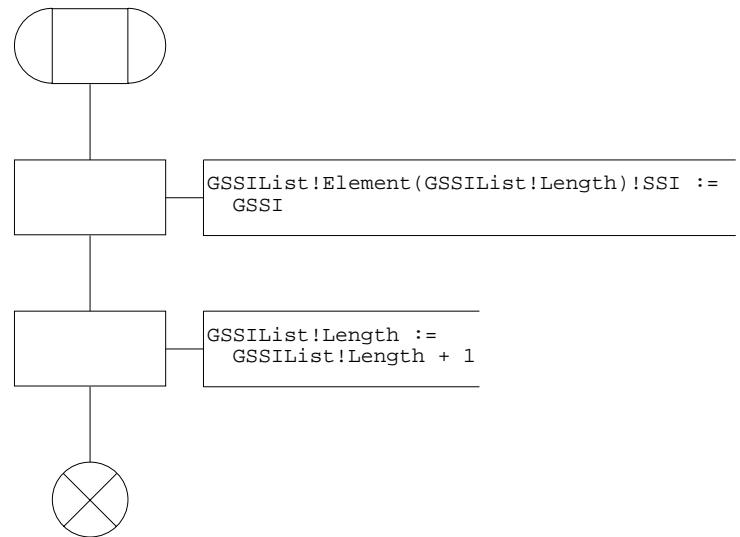
```
; FPAR
  IN IdentInfo MLE_MM_IdentitiesType;
  IN/OUT ISSI SSI_Type2,
  IN/OUT ASSI SSI_Type2,
  IN/OUT GSSIList TSI_ListType;
```



Procedure <<Process Data_transfer/Procedure UpdAddrs>> AddGSSI

1(1)

```
; FPAR  
  IN GSSI GSSI_Type,  
  IN/OUT GSSIList TSI_ListType;
```

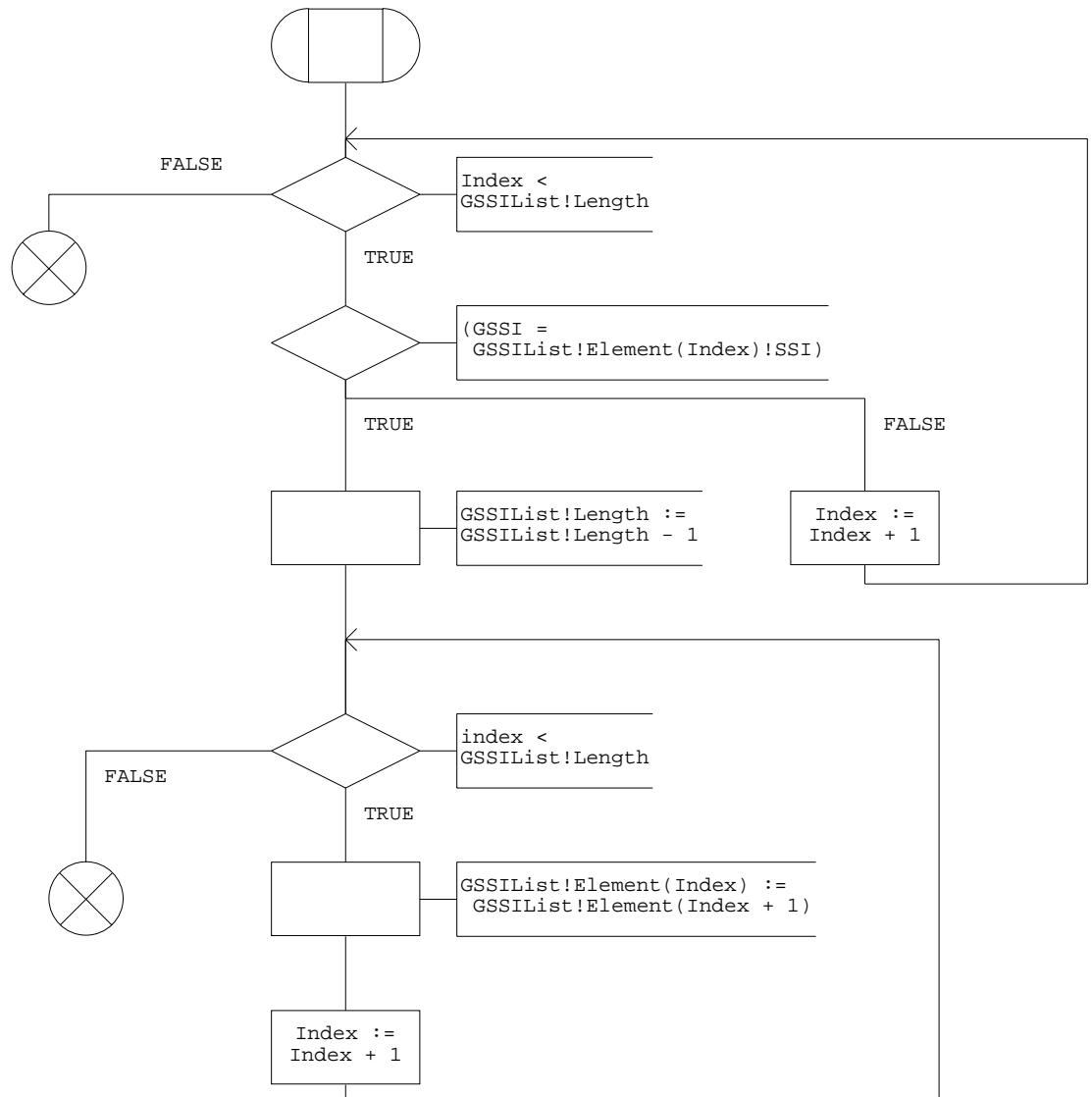


Procedure <<Process Data_transfer/Procedure UpdAddrs>> RemoveGSSI

1(1)

```
; FPAR  
  IN GSSI GSSI_Type,  
  IN/OUT GSSIList  TSI_ListType;
```

```
DCL  
  Index Natural := 0;
```

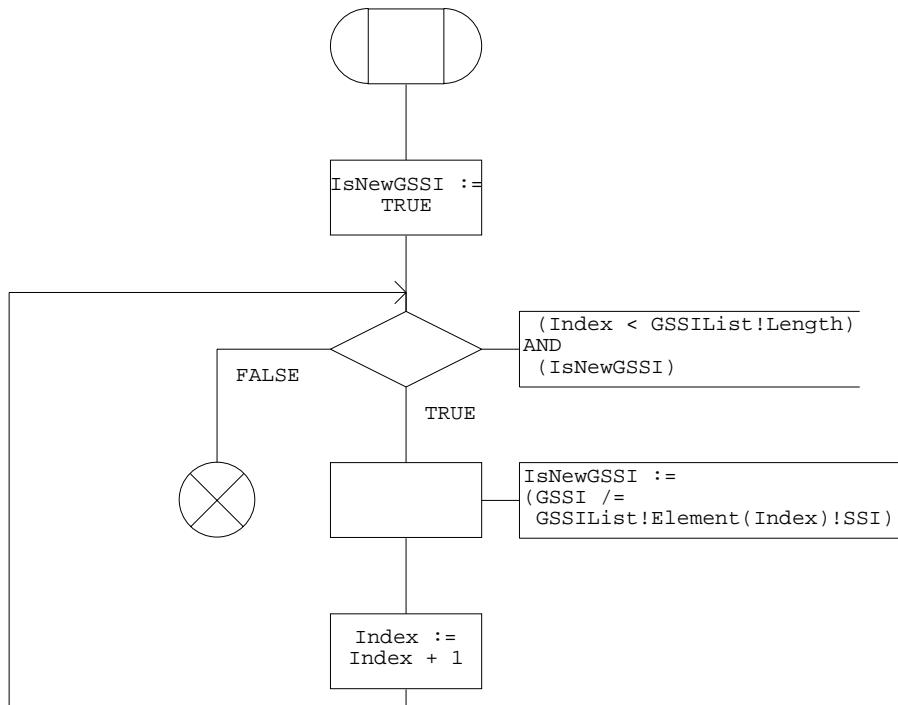


```
Procedure <<Process Data_transfer/Procedure UpdAddrs>> is_NewGSSI
```

```
1(1)
```

```
; FPAR-----\  
  IN GSSI GSSI_Type,  
  IN GSSILIST TSI_ListType,  
  IN/OUT IsNewGSSI Boolean;
```

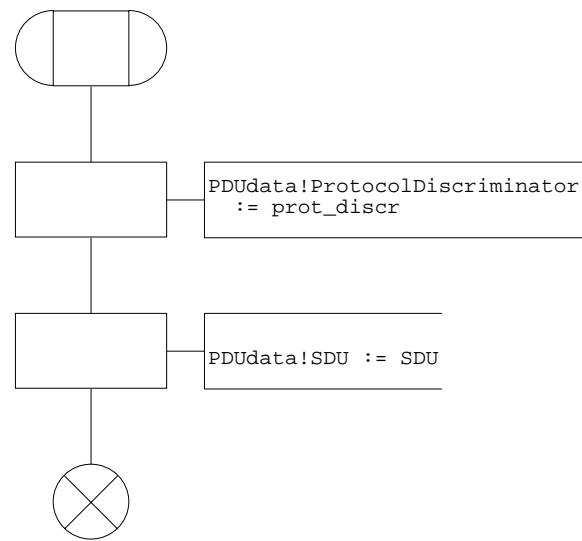
```
DCL  
Index Natural := 0;
```



```
Procedure <<Process Data_transfer>> mk_dataPDU
```

```
1(1)
```

```
; FPAR
  IN prot_discr  MLE_ProtocolDiscriminatorType,
  IN SDU MLE_SDU_Type,
  IN/OUT PDUsdata MLE_DataTransferPDUType;
```

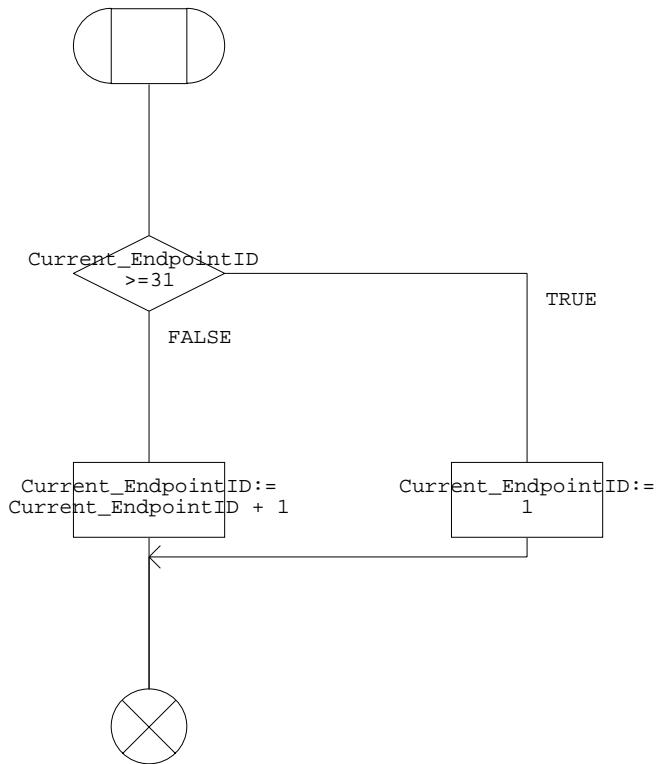


Procedure Generate_EndpointID

1(1)

```
;FPAR  
IN/OUT Current_EndpointID Endpoint_ID_Type;
```

```
/*  
This procedure is acting as an counter within the range of 0 to  
31. The range of value is outside the scope of FTS and is presented  
for validation us only. The value 0 is reserved for null Endpoint ID.  
*/
```



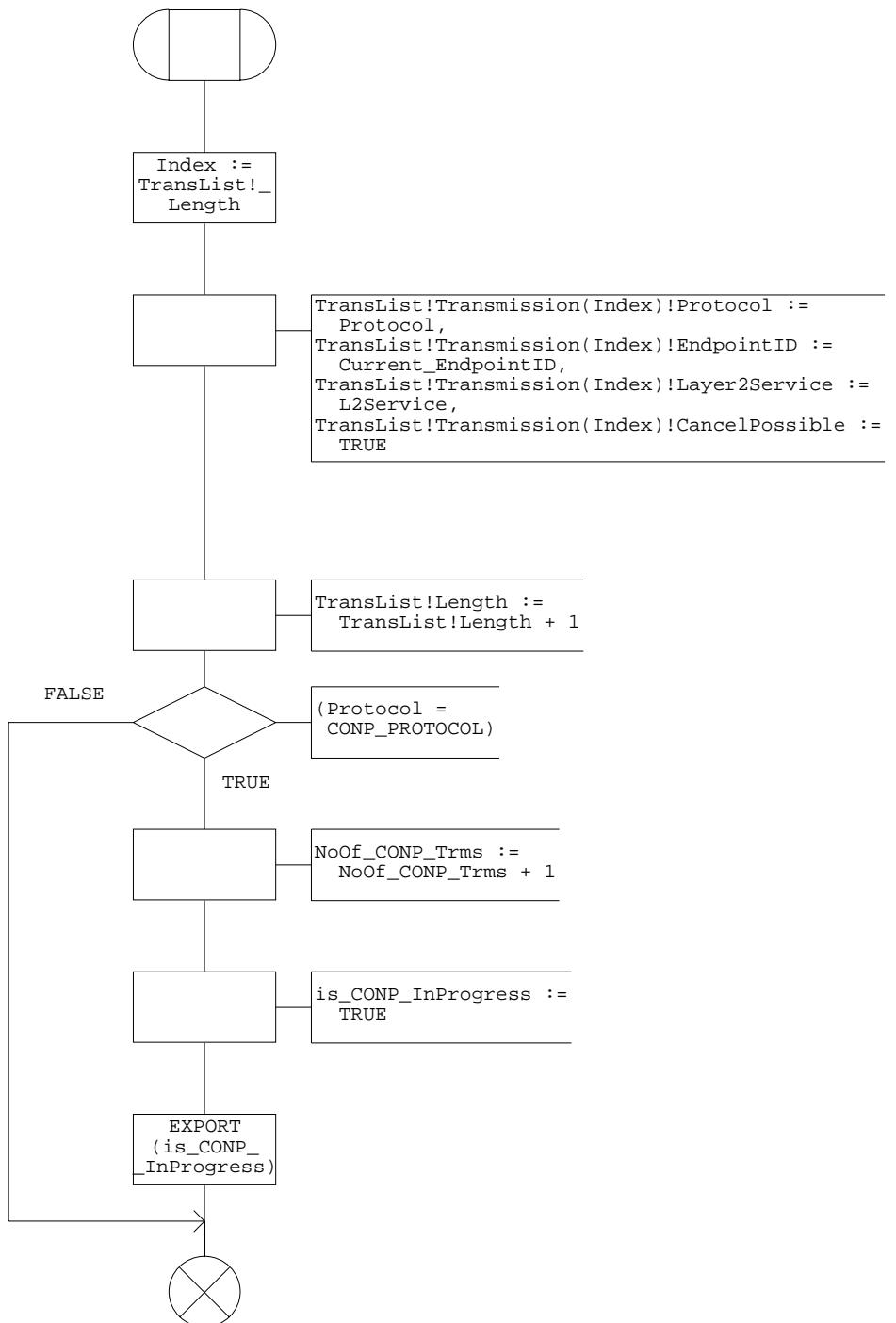
Procedure <<Process Data_transfer>> InsTrm

1(1)

```
; FPAR
  IN/OUT TransList TransmissionListType,
  IN/OUT NoOf_CONP_Trms Natural,
  IN EndpointId Endpoint_ID_Type,
  IN Protocol MLE_ProtocolDiscriminatorType,
  IN L2Service Layer2ServiceType;
```

/* Add a new Transmission descriptor to the list of on-going transmissions */

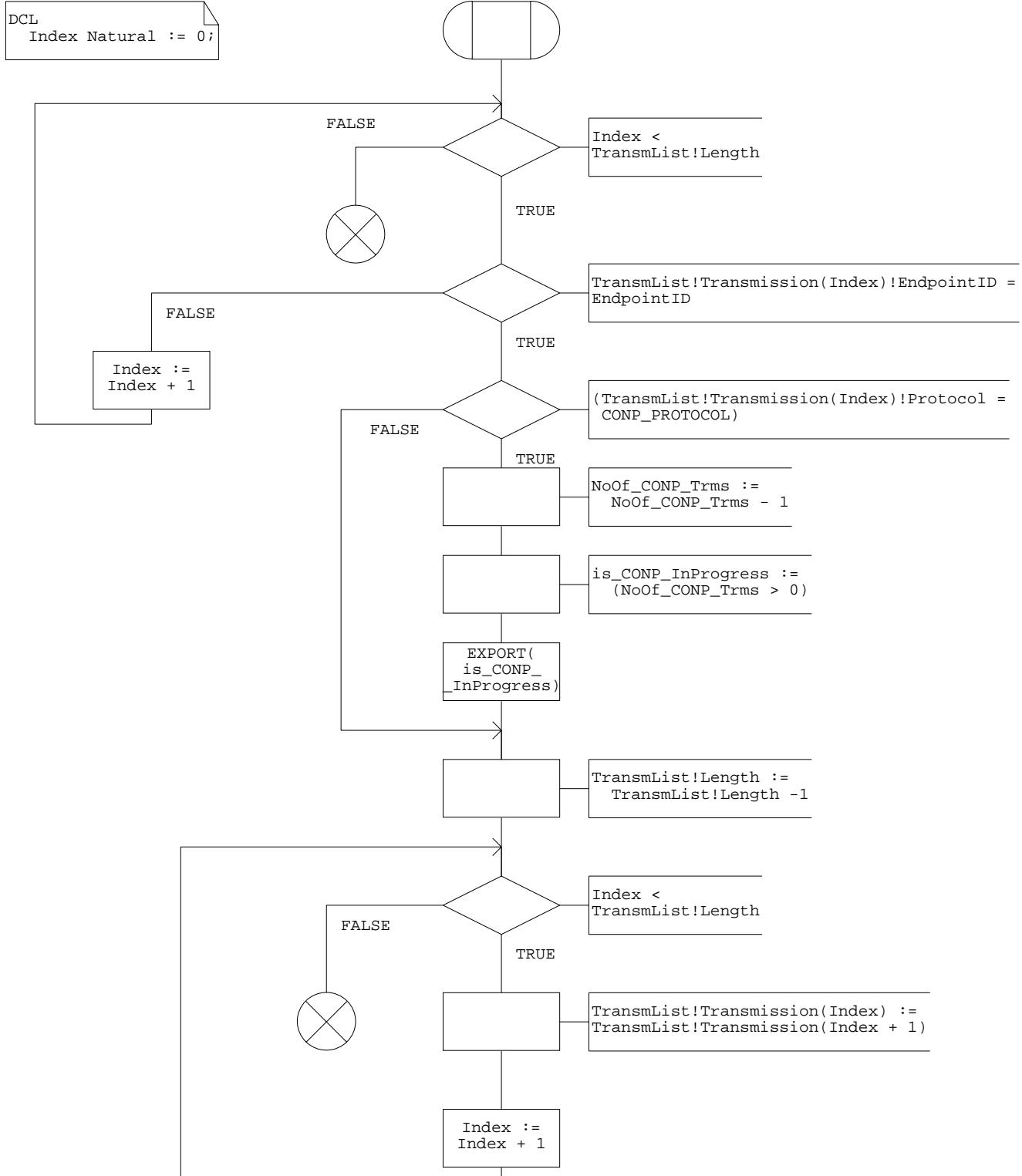
DCL
Index Natural;



Procedure <<Process Data_transfer>> RemTrm

1(1)

```
; FPAR
  IN/OUT TransmList TransmissionListType,
  IN/OUT NoOf_CONP_Trms Natural,
  IN EndpointID Endpoint_ID_Type;
```

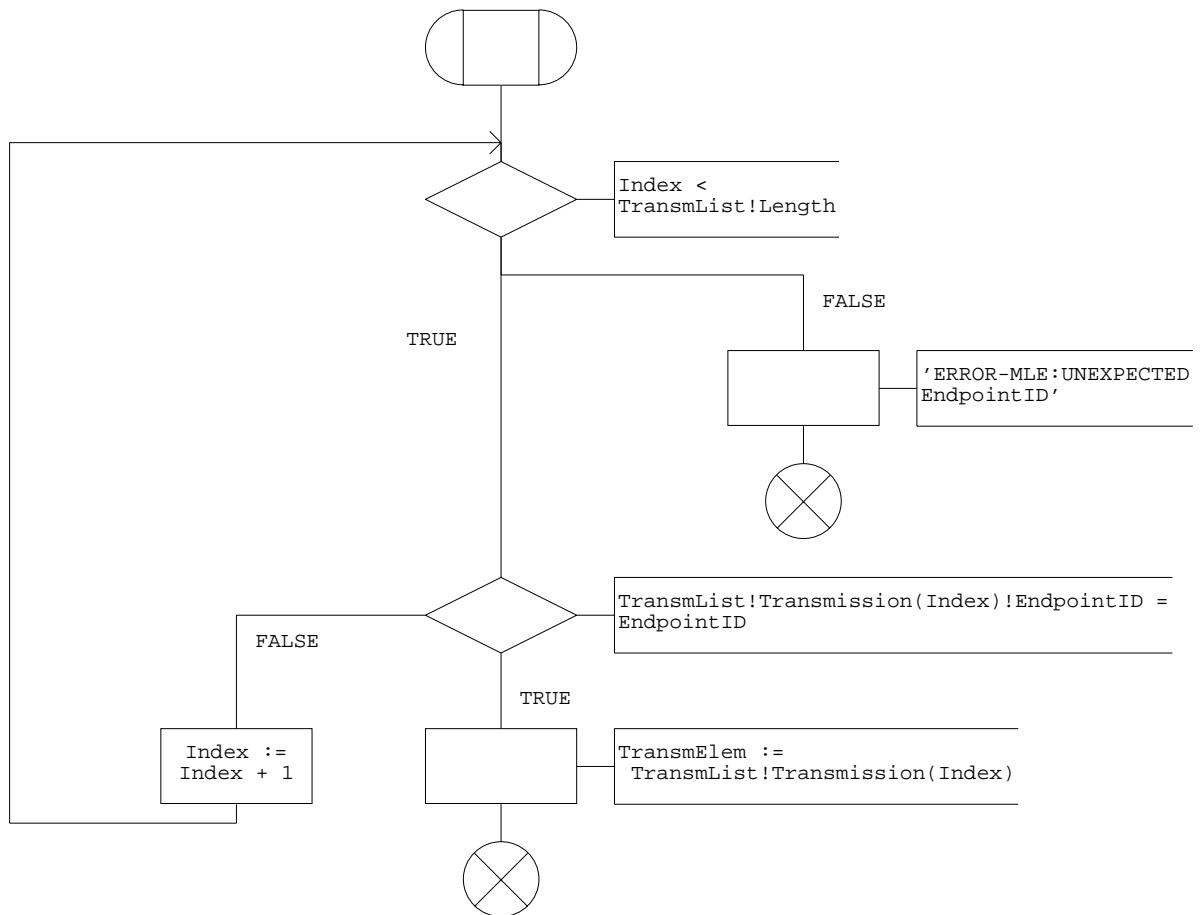


Procedure <<Process Data_transfer>> GetTrmProt

1(1)

```
; FPAR
  IN TransmList TransmissionListType,
  IN EndpointID Endpoint_ID_Type,
  IN/OUT TransmElem TransmissionDescrType;
```

```
DCL
  Index Natural := 0;
```



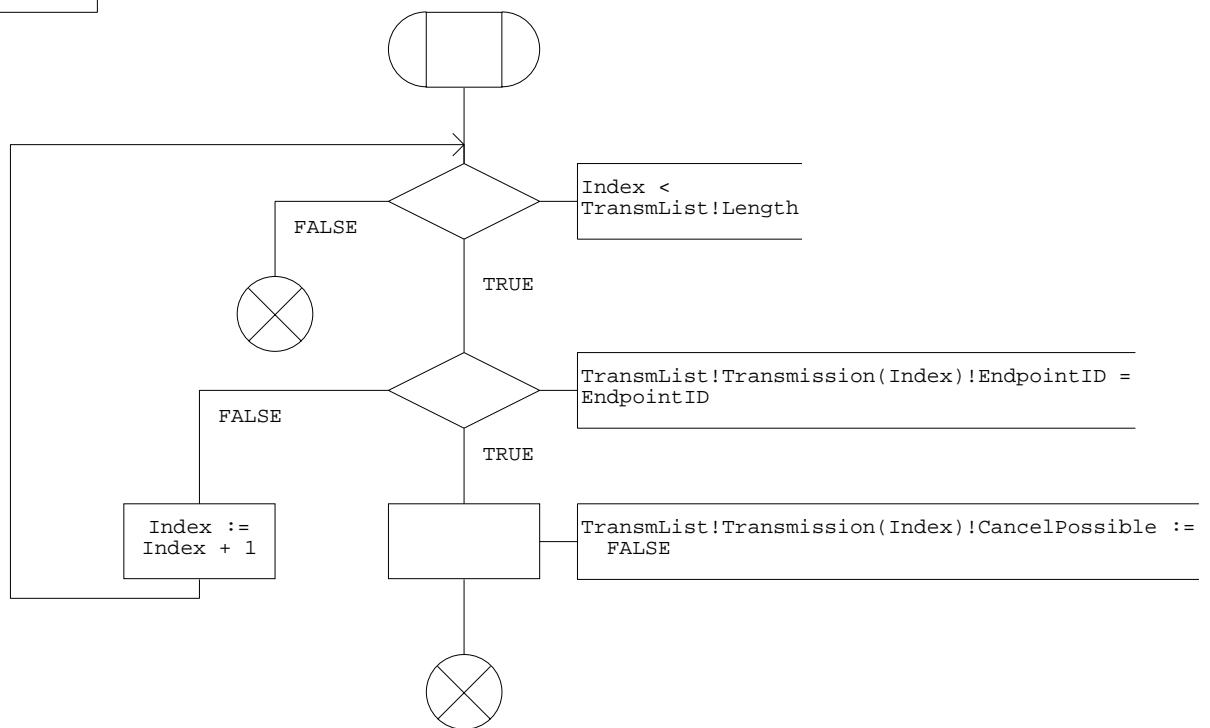
Procedure TrmCancelNoLonger

1(1)

```
; FPAR
  IN/OUT TransmList TransmissionListType;
  IN EndpointID Endpoint_ID_Type;
```

```
/* Set the Cancel Possible flag to false for
   the transmission descriptor with the
   specified EndpointID.
```

```
DCL
  Index Natural := 0;
```

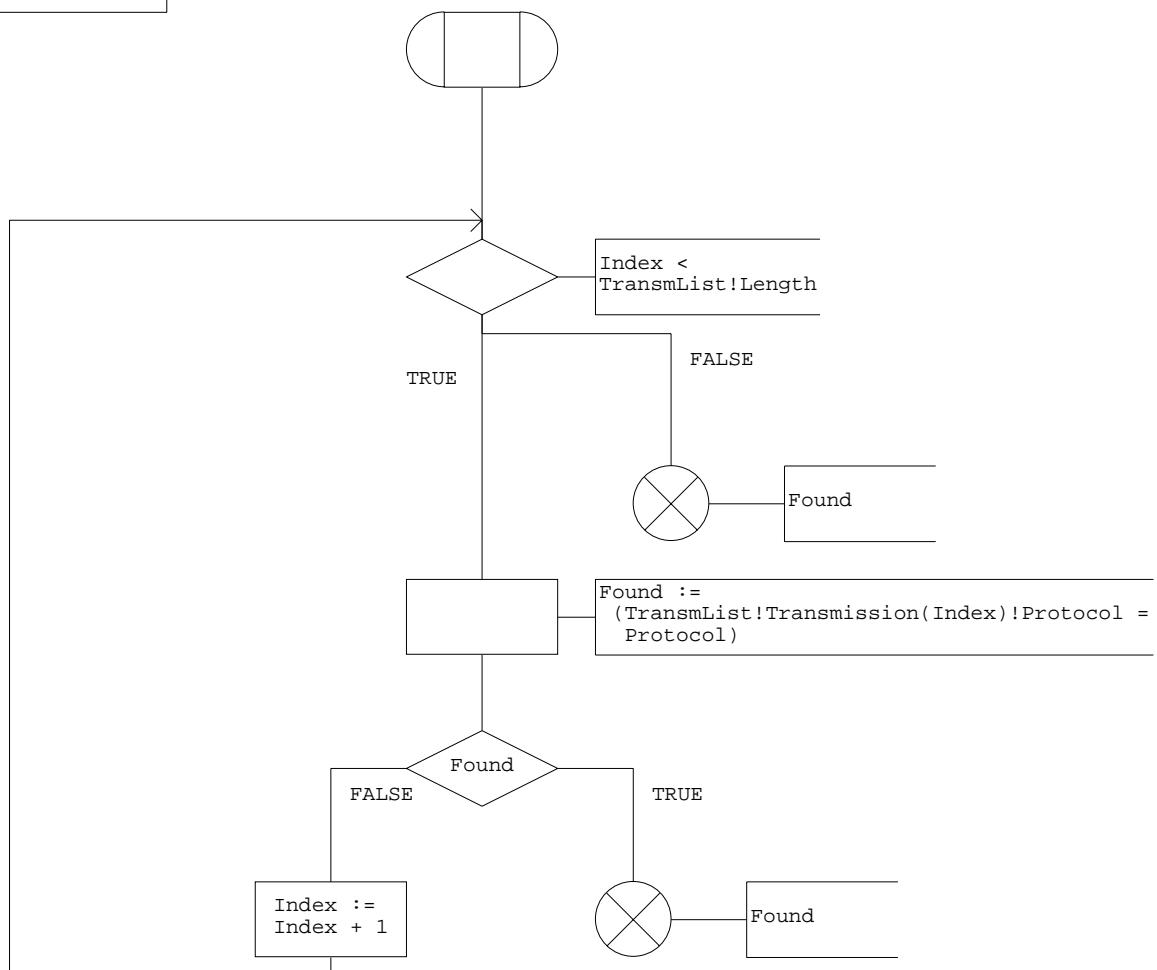


Procedure is_TrmProt

1(1)

```
; FPAR
  IN TransmList TransmissionListType,
  IN Protocol MLE_ProtocolDiscriminatorType;
RETURNS Boolean;
```

```
DCL
  Index Natural := 0,
  Found Boolean := FALSE;
```



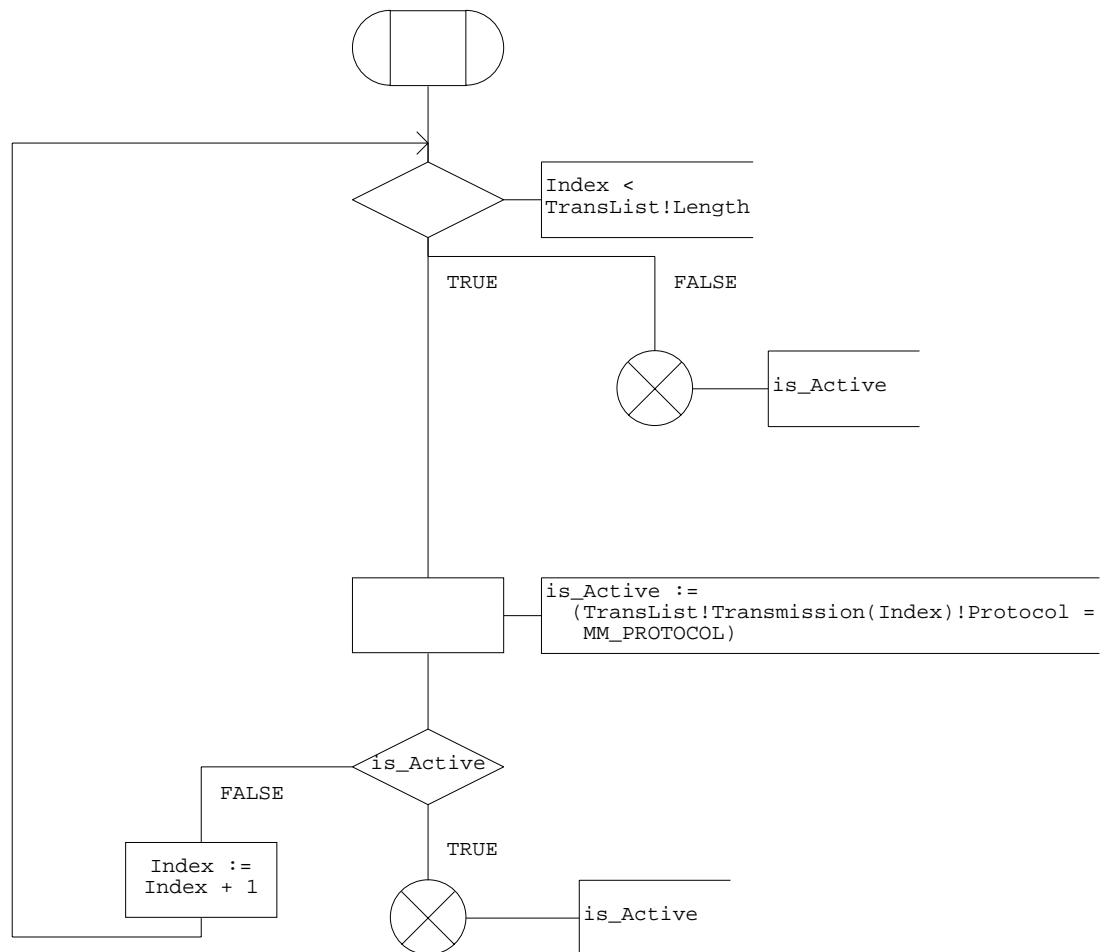
Procedure is_Active_MLE

1(1)

```
; FPAR
  IN/OUT TransList TransmissionListType;
RETURNS
  Boolean;
```

```
/* Check if there are any MM calls in
   progress and return TRUE if so. This is used to
   notify layer 2 in the TLC_CONFIGURE SP.
*/
```

```
DCL
  Index Natural := 0,
  is_Active Boolean := FALSE;
```



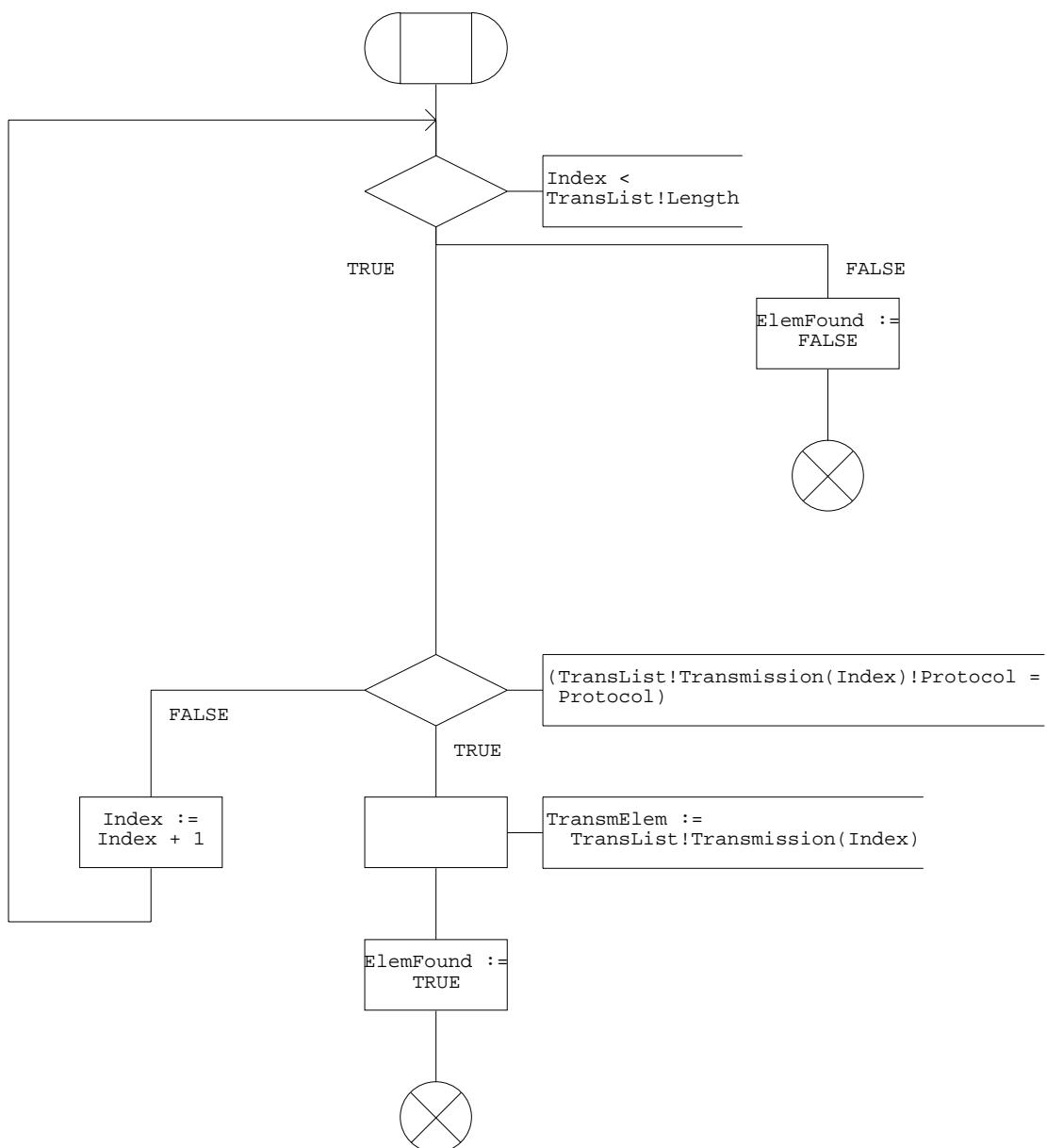
Procedure GetTrmDescr

1(1)

```
; FPAR
  IN/OUT TransList TransmissionListType,
  IN Protocol MLE_ProtocolDiscriminatorType,
  IN/OUT ElemFound Boolean,
  IN/OUT TransmElem TransmissionDescrType;
```

```
/* Get the transmission element descriptor
   for the specified protocol entity if
   any. If none is present the ElemFound is
   set to FALSE otherwise the element is
   returned in TransmElem
*/
```

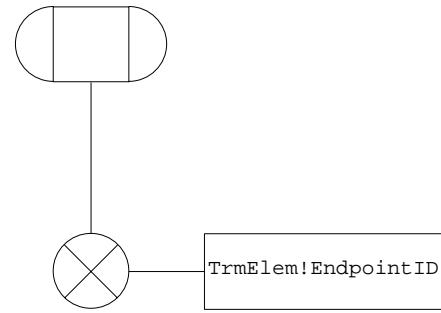
```
DCL
Index Natural := 0;
```



Procedure GetEndpointID

1(1)

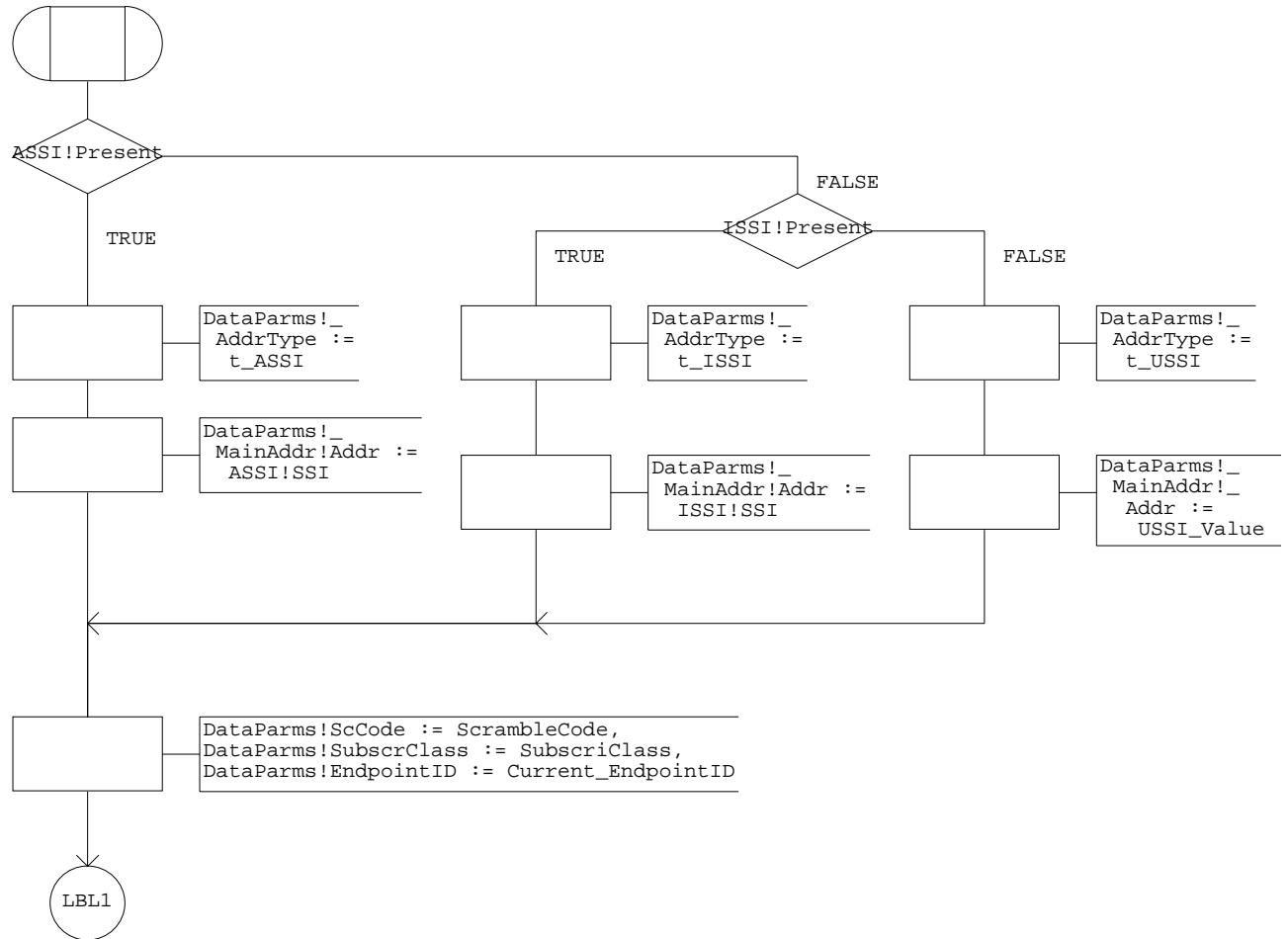
```
;RETURNS Endpoint_ID_Type;
```



EXPORTED Procedure GetSendDataParms

1 (2)

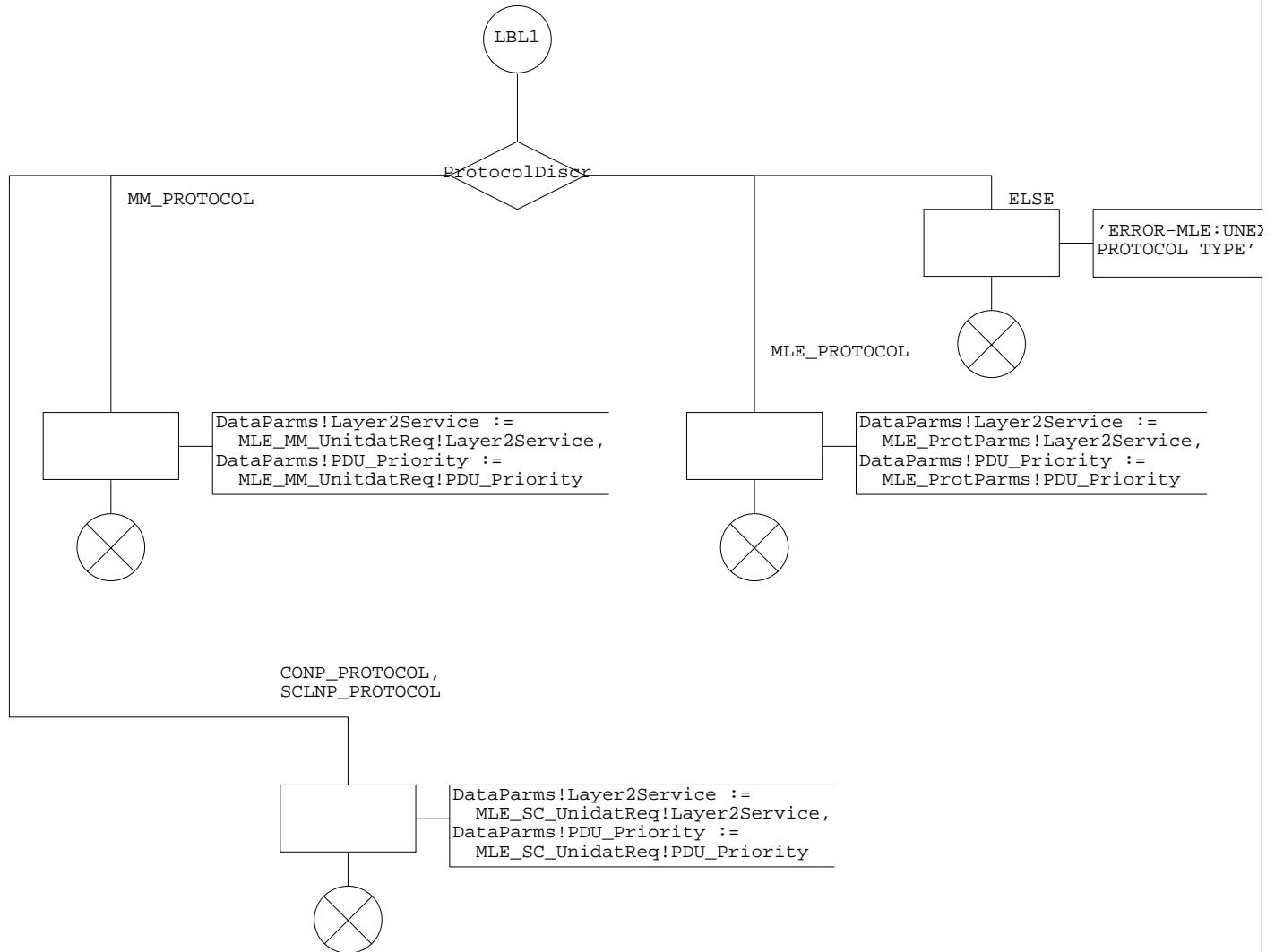
```
; FPAR
  IN/OUT DataParms DataReqInfoType,
  IN ProtocolDiscr MLE_ProtocolDiscriminatorType;
```



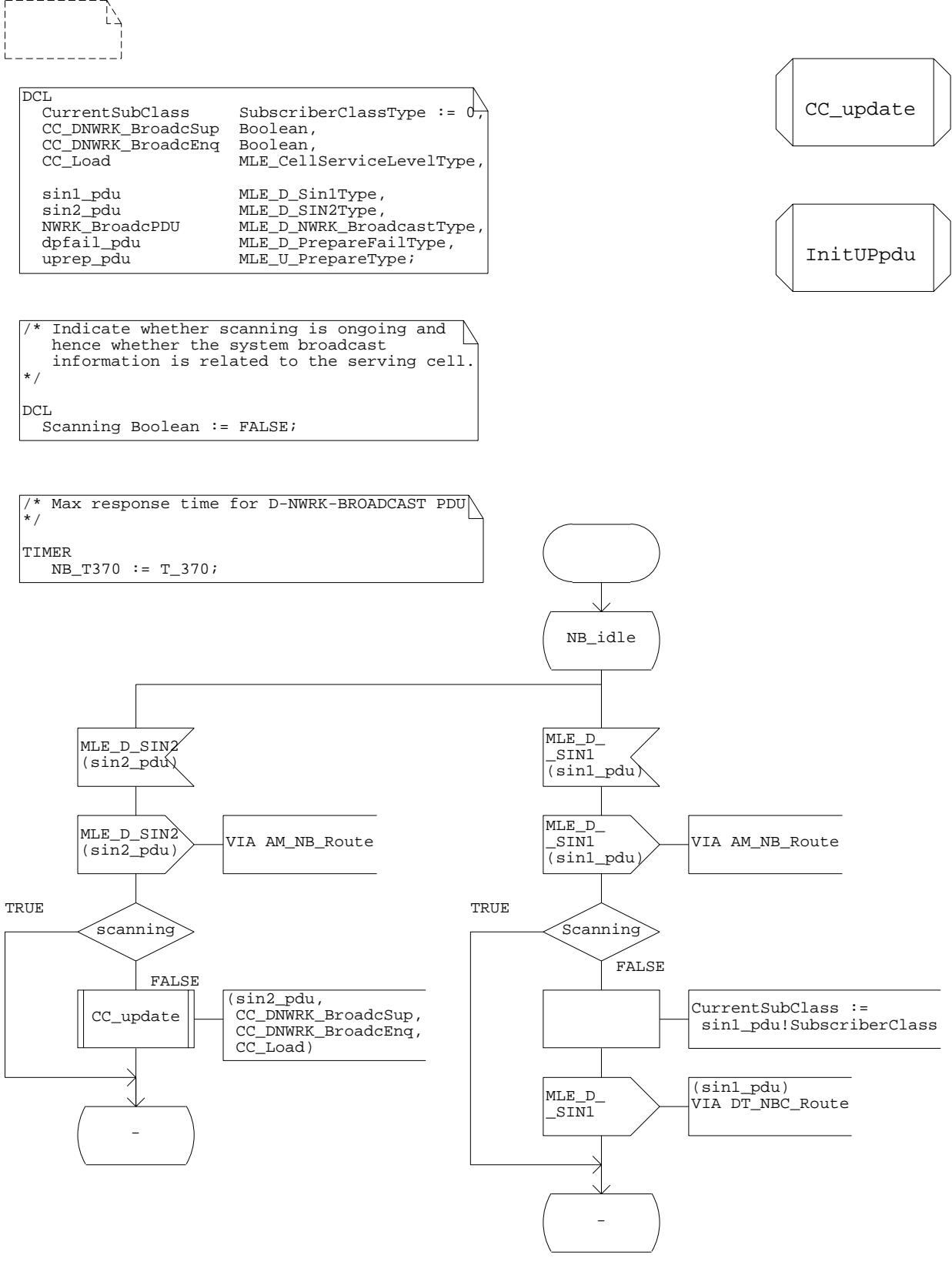
EXPORTED Procedure GetSendDataParms

2 (2)

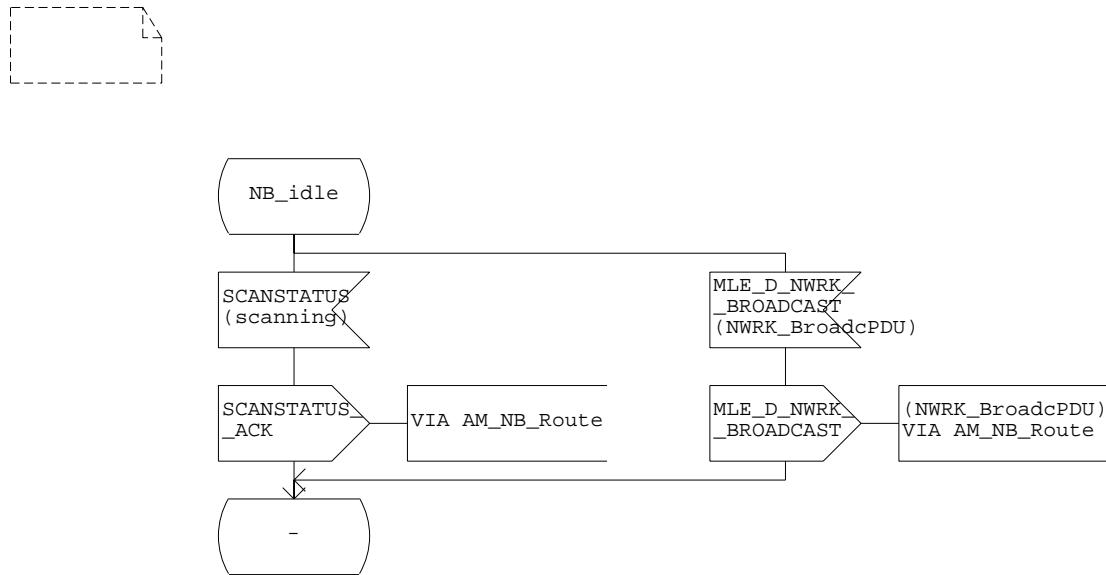
```
; FPAR
IN/OUT DataParms DataReqInfoType,
IN ProtocolDiscr MLE_ProtocolDiscriminatorType;
```



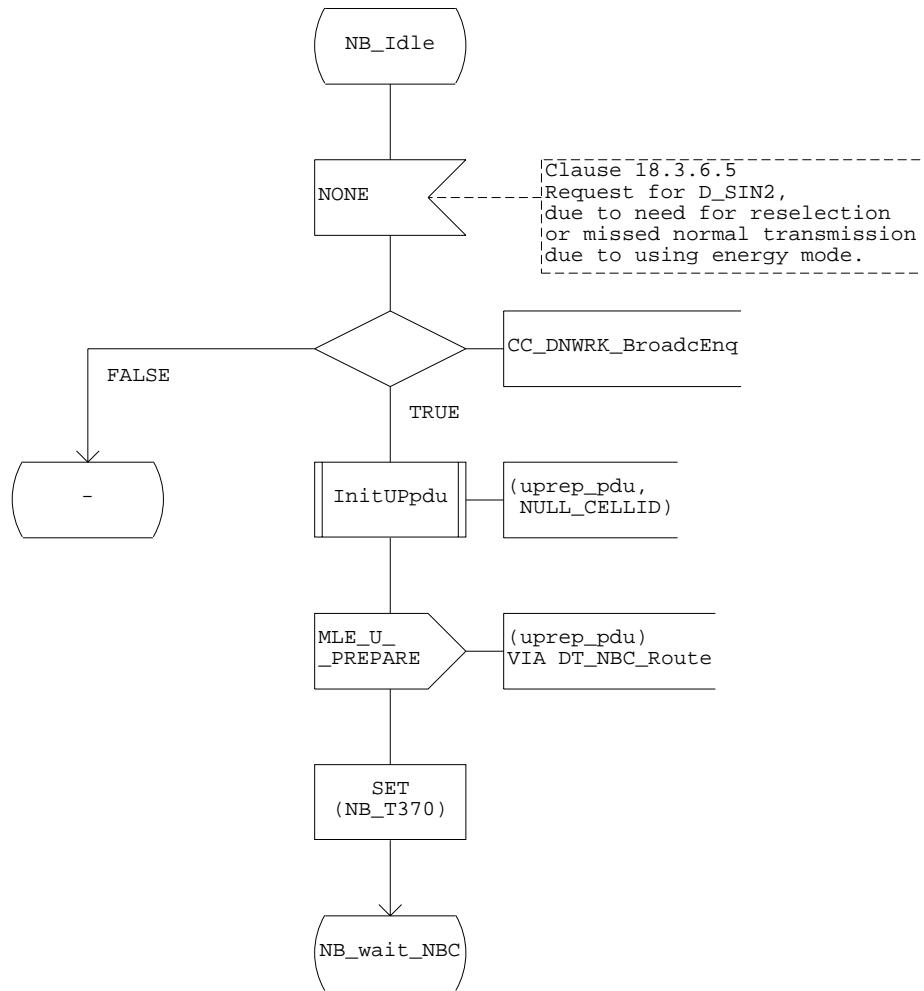
Process <<Substructure MLE_MS_Protocol_Type/Block Data_Services>> Network_Broadcast 1(4)



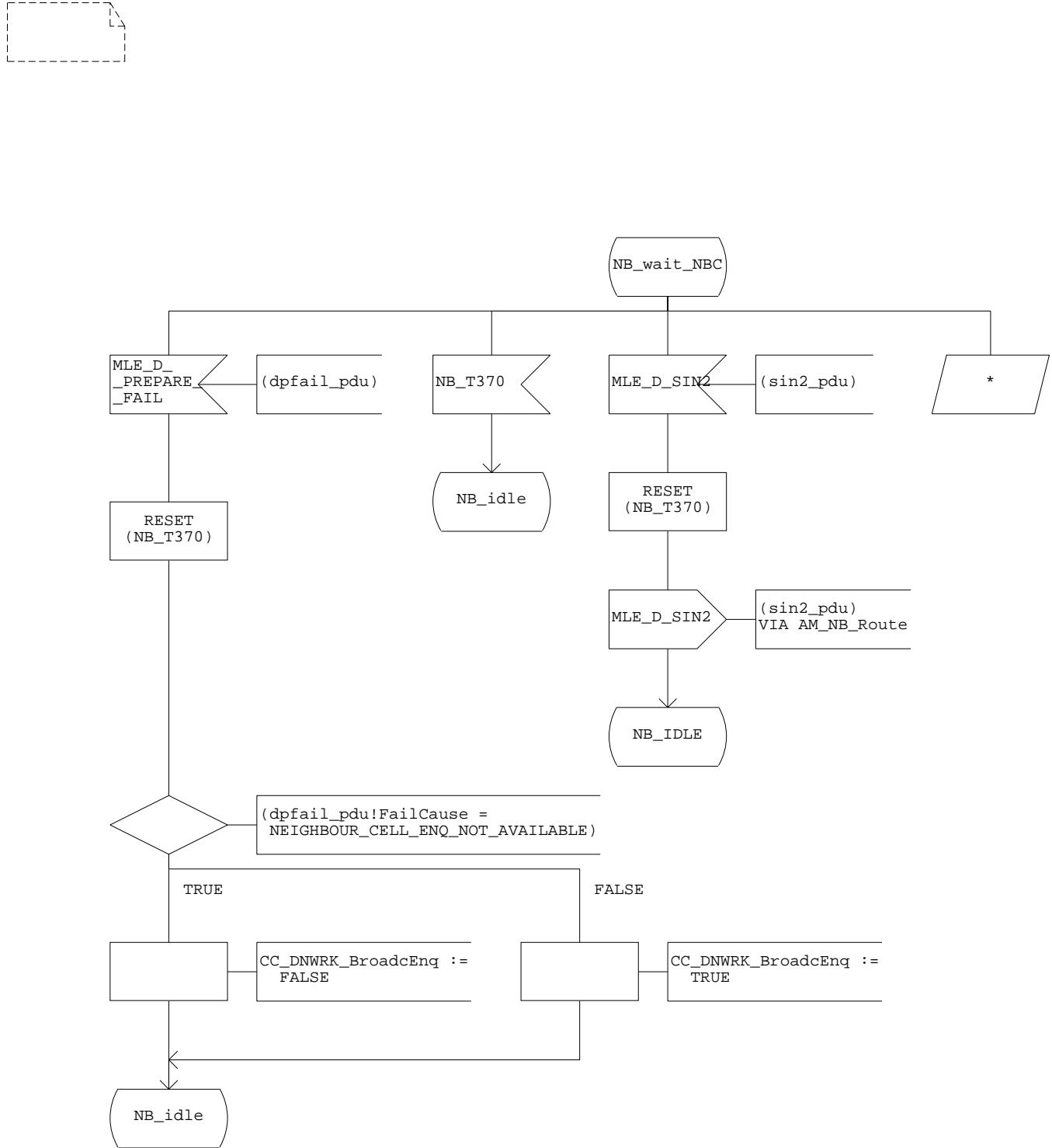
Process <<Substructure MLE_MS_Protocol_Type/Block Data_Services>> Network_Broadcast 2(4)



Process <<Substructure MLE_MS_Protocol_Type/Block Data_Services>> Network_Broadcast 3(4)



Process <<Substructure MLE_MS_Protocol_Type/Block Data_Services>> Network_Broadcast 4(4)



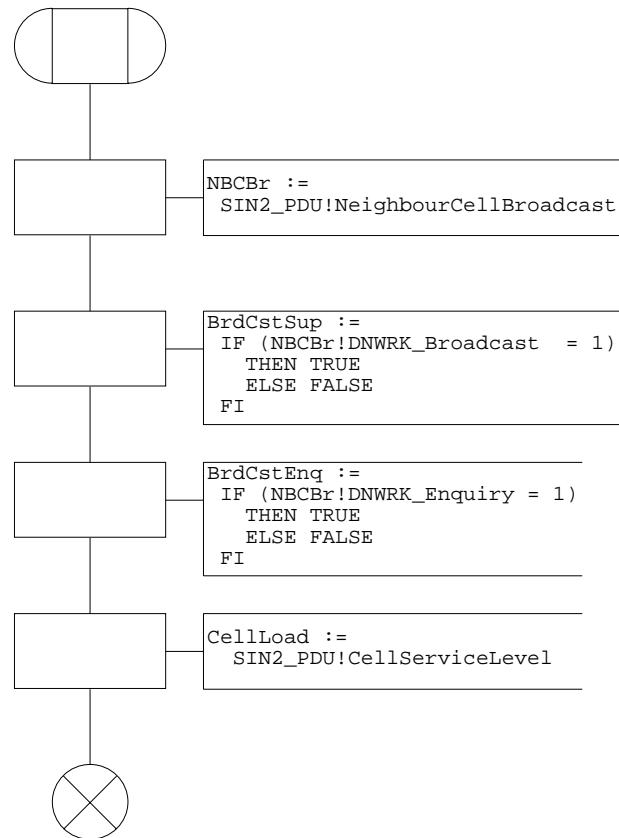
Procedure CC_update

1(1)

```
; FPAR
  IN/OUT SIN2_PDU  MLE_D_SIN2Type,
  IN/OUT BrdCstSup Boolean,
  IN/OUT BrdCstEng Boolean,
  IN/OUT CellLoad MLE_CellServiceLevelType;
```

```
/* Update the information of the current cell
   based on the information received in a
   MLE_D_SIN2 pdu.
*/
```

```
DCL
  NBCBr NeighbourCellBroadcastType;
```



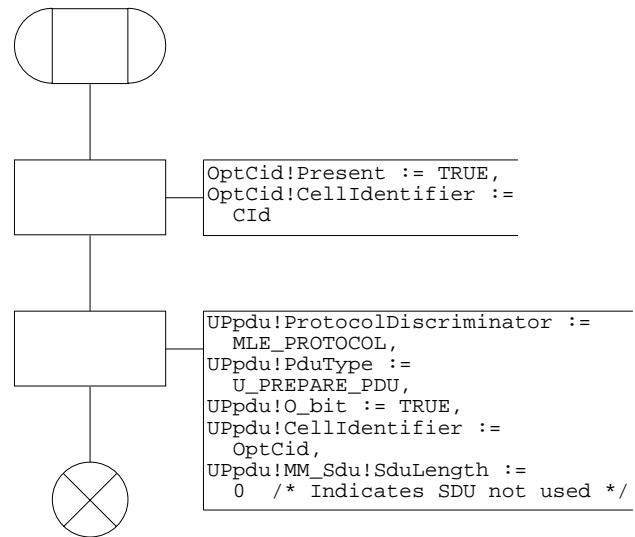
Procedure InitUP pdu

1(1)

```
; FPAR  
IN/OUT UP pdu MLE_U_PrepType,  
IN CId CellIdentifierType;
```

```
/* InitUP pdu initialises the fields of the U_PREPARE pdu */
```

```
DCL  
OptCid CellIdentifierType2;
```



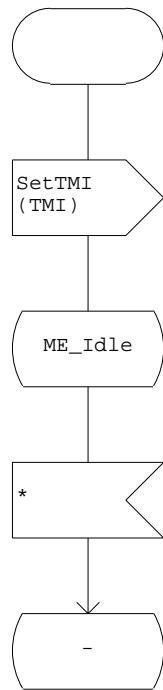
Process Management_Entity

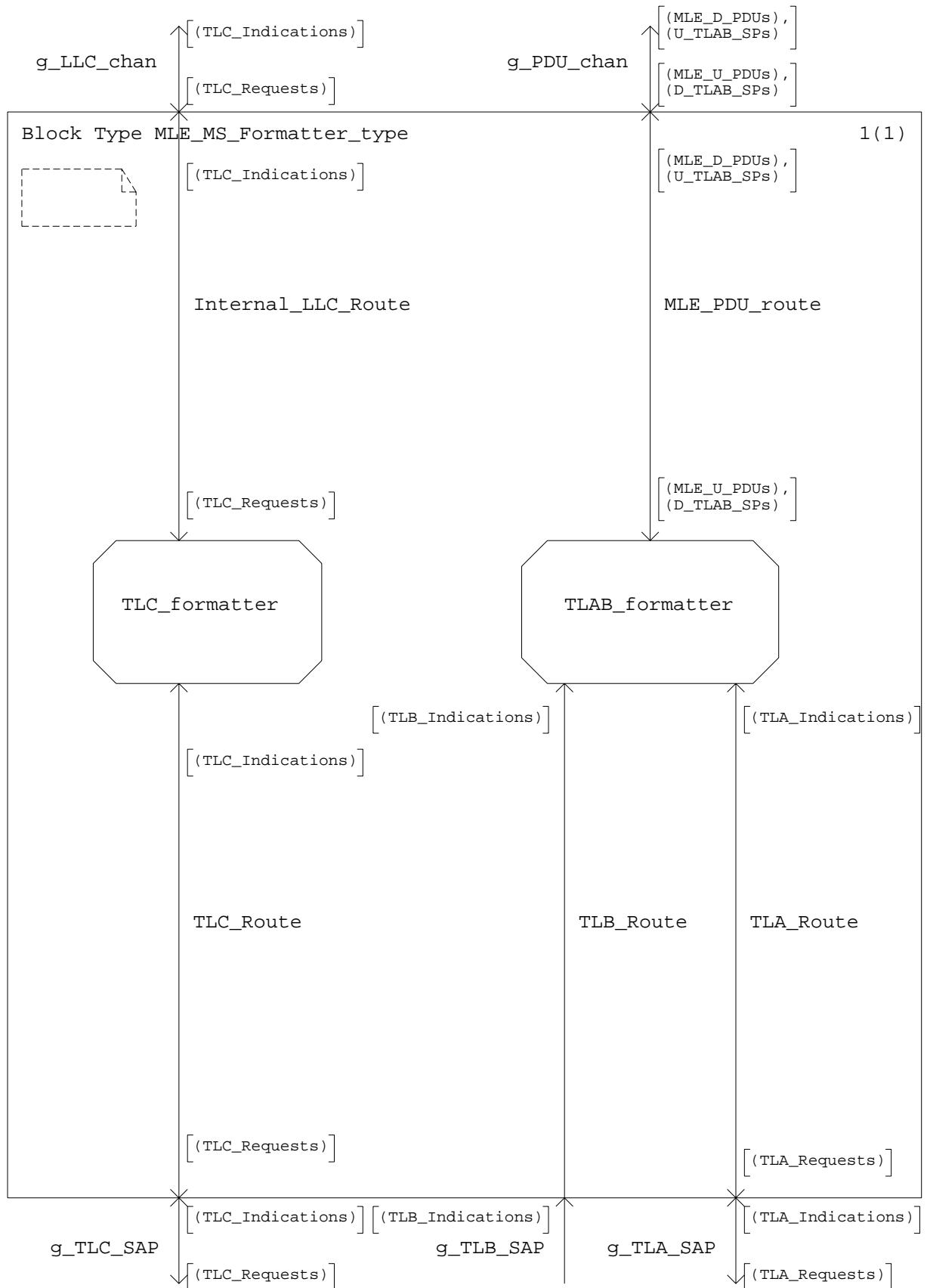
1(1)



```
/* The MS has a unique TMI  
which is held by the  
Management entity  
*/  
  
SYNONYM TMI TMI_Type =  
(. 0,0,0 .);
```

```
/* No behaviour requirements  
are specified for the  
Management Entity  
*/
```





Process TLAB_formatter

1(14)



```
/* *** LOCAL DATA TYPE DEFINITIONS ***/
```

```
/* Data type for QoS checking results */
NEWTYPE QoS_ResultType
  LITERALS
    QOS_ACCEPTED,
    QOS_NOT_ACCEPTED,
    NEW_QOS
ENDNEWTYPE;
```

Process TLAB_formatter

2(14)

```
/* *** LOCAL SYNONYM DEFINITIONS ***/
```

```
/* Number of bits added to the MLE SDUs when transferred to the LLC SDU. */  
SYNONYM MLE_SDU_addition Natural = 3;
```

```
/* *** IMPORT OF REMOTE PROCEDURES ***/
```

```
/* Procedure for transfer of associated data parameters for data PDUs. */  
IMPORTED PROCEDURE GetSendDataParms;  
  FPAR  
    IN/OUT DataReqInfoType,  
    IN MLE_ProtocolDiscriminatorType;
```

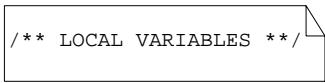
```
/* Procedure for transfer of associated data parameters for data PDUs. */  
IMPORTED PROCEDURE GetEndpointID;  
  RETURNS Endpoint_ID_type;
```

Process TLAB_formatter

3(14)



```
/* *** LOCAL VARIABLE DEFINITIONS ***/
```

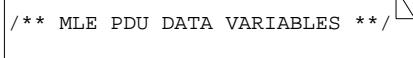


DCL

```
DataToSend      Boolean := FALSE, /* To identify if there are data to be sent
                                  after an AL setup phase has been performed. */
FCS_FlagSetting FCS_FlagType,      /* Indicate the setting of the FCS_Flag in the MS */
                                  /* Endpoint_ID is to be associated to. */
PDU_Kind        MLE_D_PduType,
Prot            MLE_ProtocolDiscriminatorType,
SubscriClass    SubscriberClassType,
NewQoS          QoS_Type,
QoS_Result     QoS_ResultType;
```

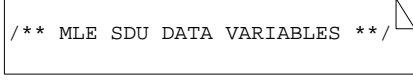
DCL

```
dps DataReqInfoType; /* Variable for transferring information from
                       the Data Transfer process to the formatter
                       concerning the PDU to sent. */
```



DCL

```
Data_PDU        MLE_DataTransferPDUType,
U_Prepare_PDU   MLE_U_PrepareType,
MLE_SIN1Ind    MLE_D_SIN1Type,
MLE_SIN2Ind    MLE_D_SIN2Type;
```



DCL

```
MLE_Cancel    MLE_CancelType;
```

Process TLAB_formatter

4 (14)



```
/** LLC SP DATA VARIABLES **/
```

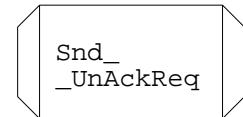
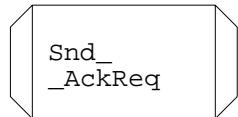
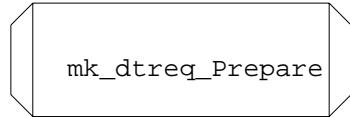
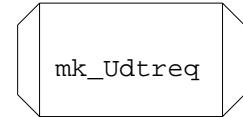
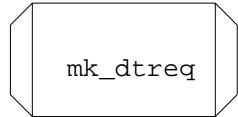
```
DCL
DataConf      TLA_DataConfirmType,
DataInd       TLA_DataIndicationType,
DataReq        TLA_DataRequestType,
UnitDataInd   TLA_UnitdataIndicationType,
UnitDataReq   TLA_UnitdataRequestType,
UnitDataConf  TLA_UnitdataConfirmType,
CancelReq     TLA_CancelRequestType,
SIN2Ind       TLB_BROADCAST2IndicationType,
SIN1Ind       TLB_BROADCAST1IndicationType;
```

Process TLAB_formatter

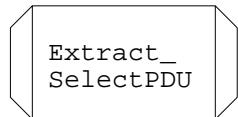
5 (14)



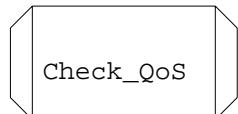
```
/* Initialise SDU descriptors for LLC SPs */
```



```
/* Initialise MLE PDUS */
```



```
/* Miscellaneous procedures */
```



Process TLAB_formatter

6(14)



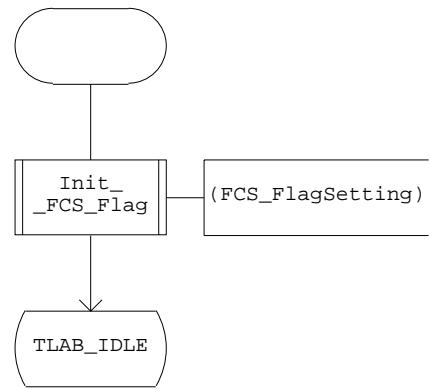
/* PROCEDURE REFERENCES */

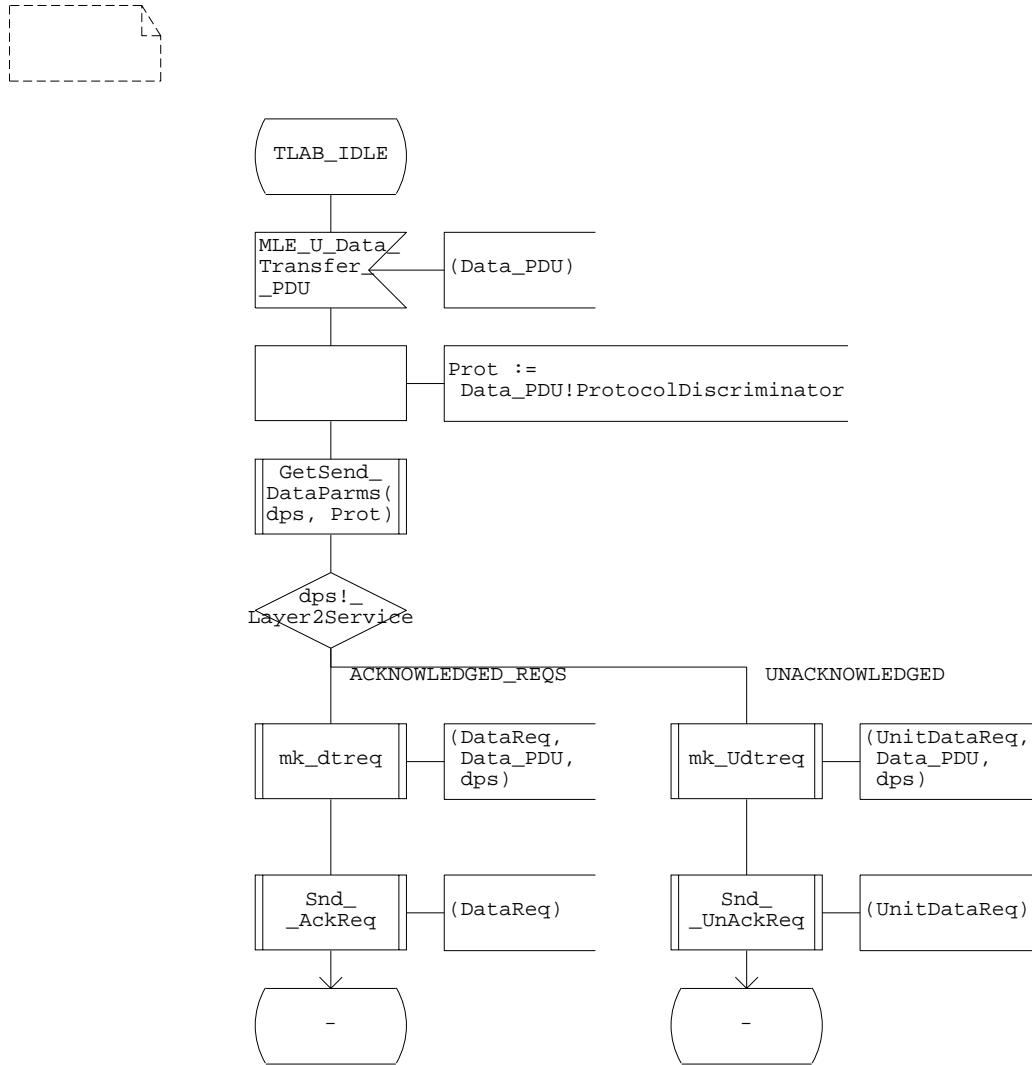
/* Initiate the FCS flag */

Init_
_FCS_Flag

Process TLAB_formatter

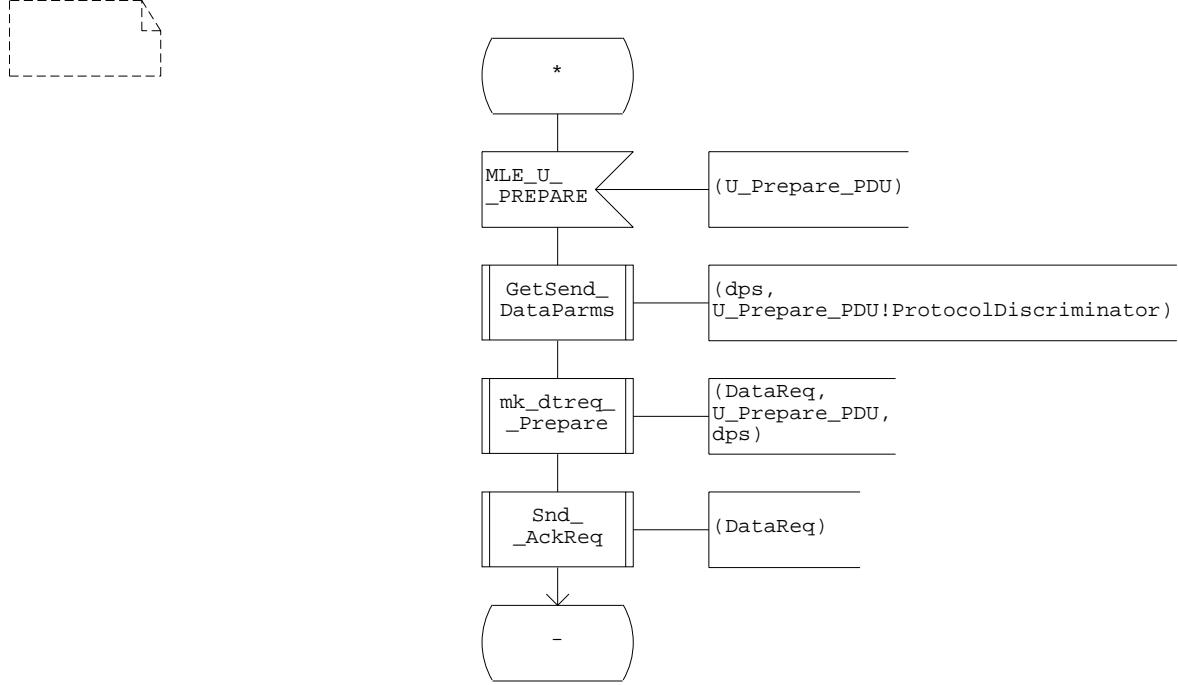
7 (14)





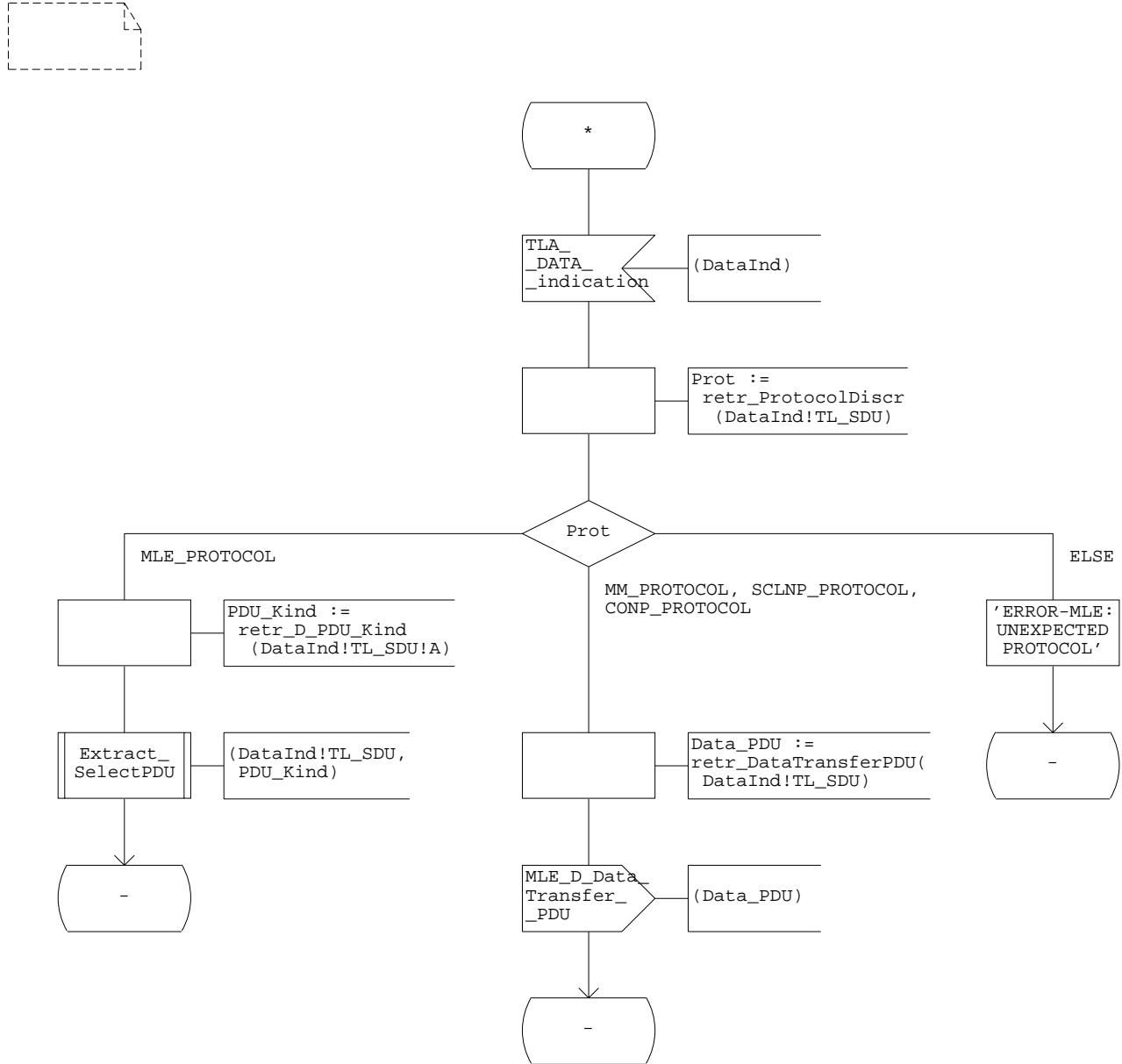
Process TLAB_formatter

9 (14)



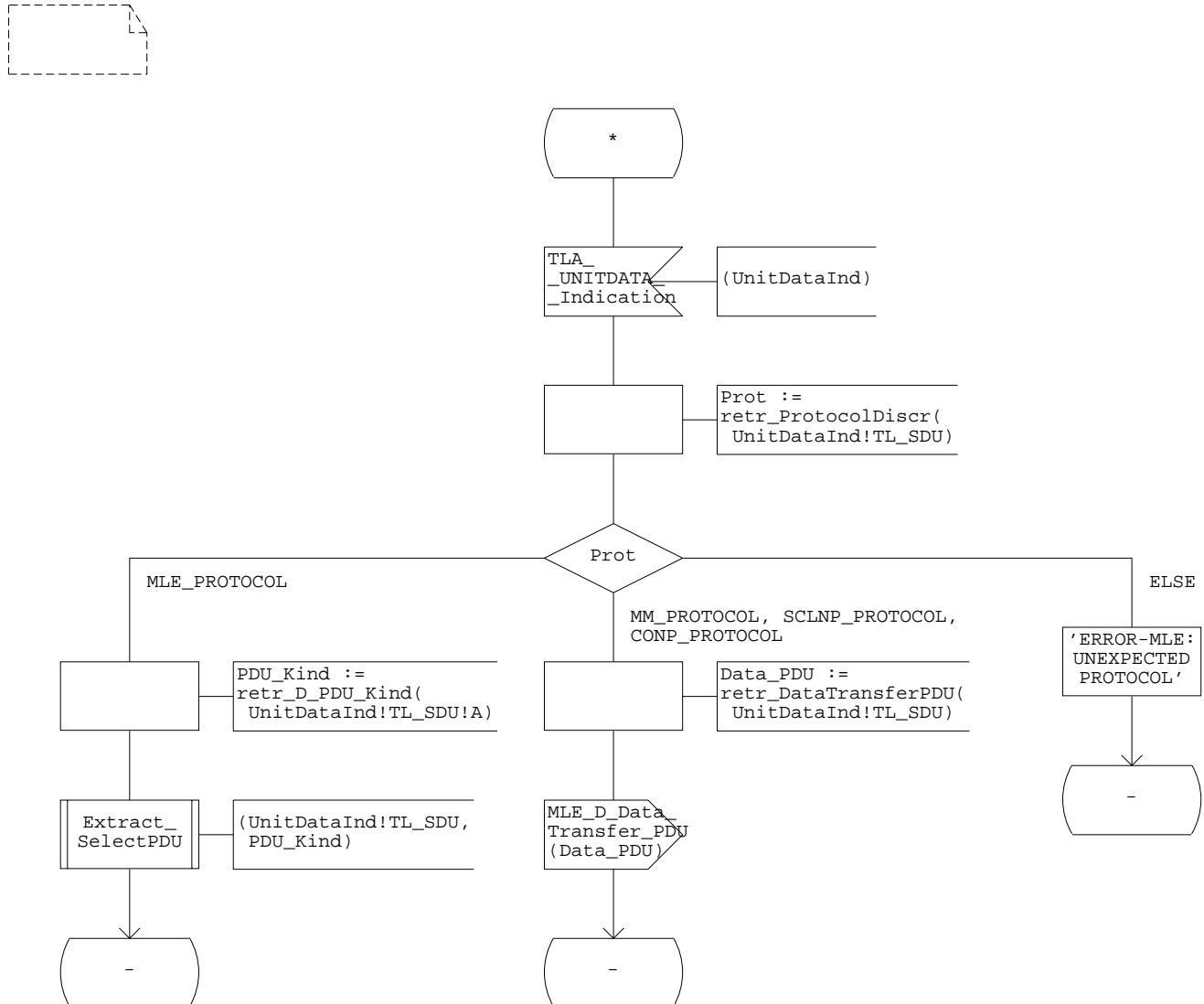
Process TLAB_formatter

10(14)



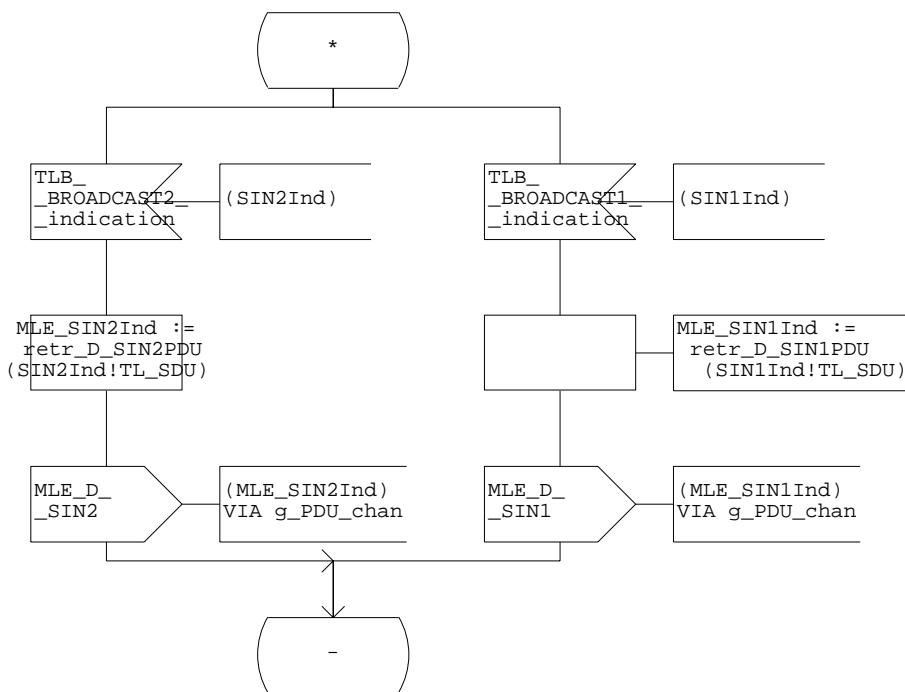
Process TLAB_formatter

11(14)



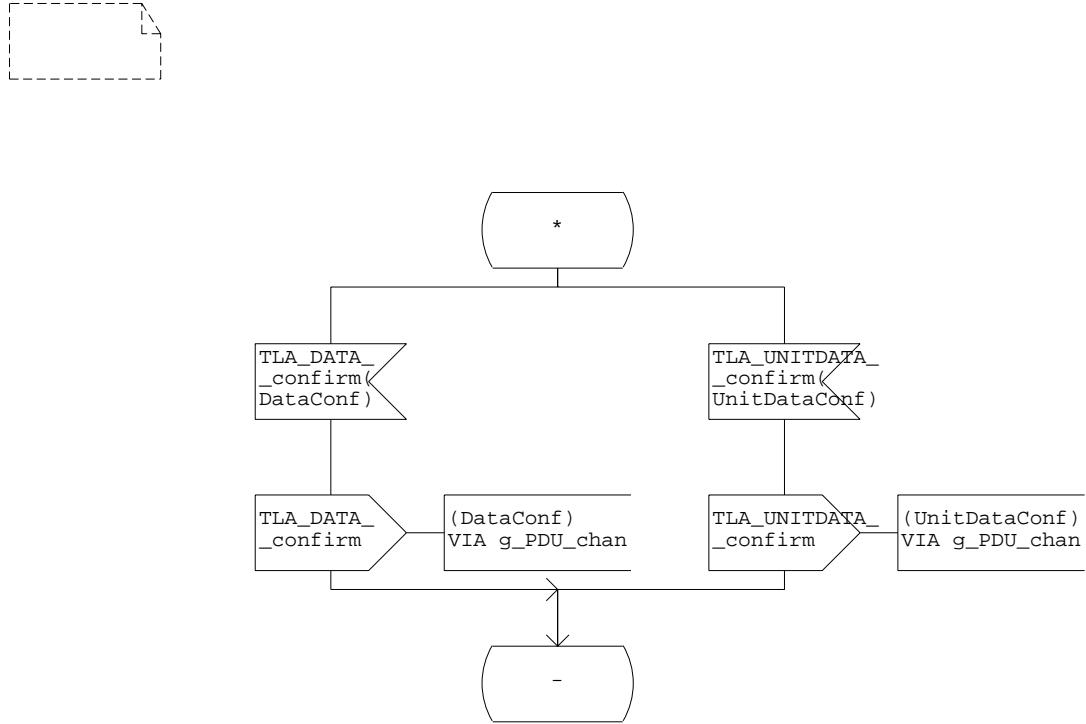
Process TLAB_formatter

12(14)



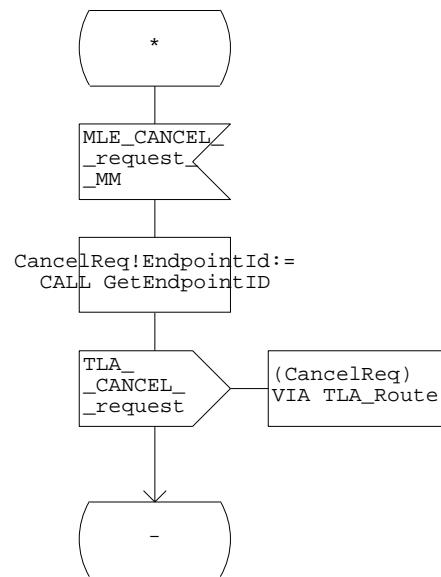
Process TLAB_formatter

13(14)



Process TLAB_formatter

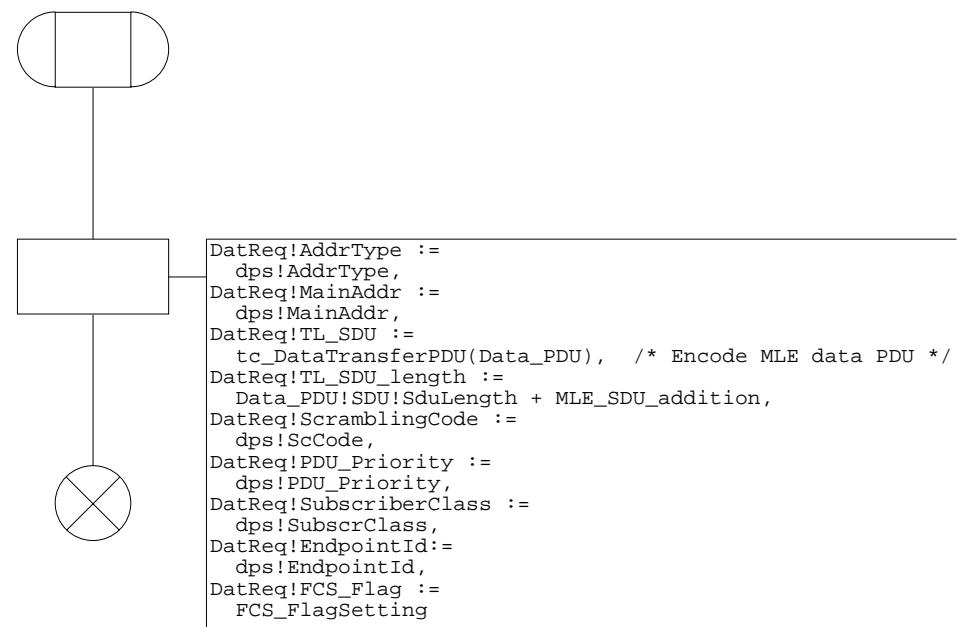
14(14)



Procedure <<Process TLAB_formatter>> mk_dtreq

1(1)

```
; FPAR
IN/OUT DatReq  TLA_DataRequestType,
IN Data_PDU  MLE_DataTransferPDUType,
IN dps DataReqInfoType;
```

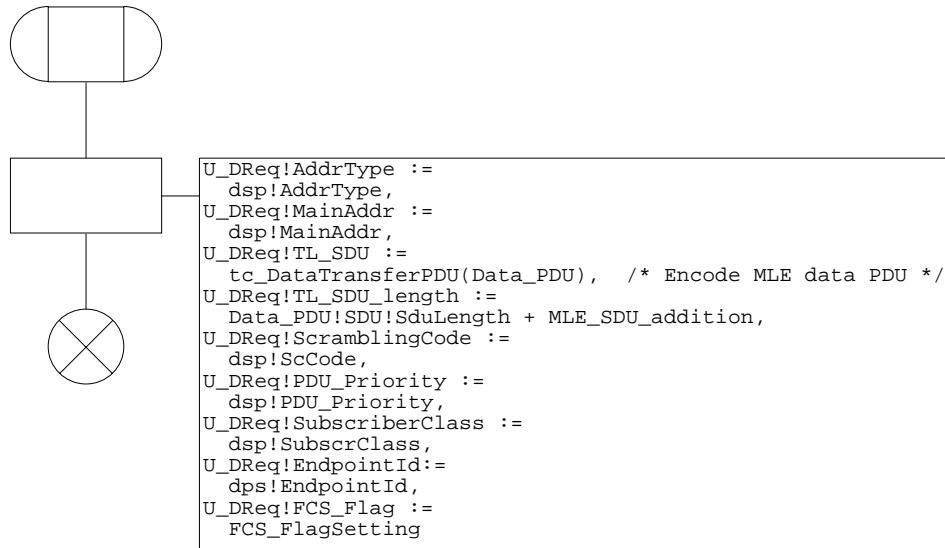


Procedure <<Process TLAB_formatter>> mk_Udtreq

1(1)

```
; FPAR
IN/OUT U_DReq TLA_UnitdataRequestType,
IN Data_PDU MLE_DataTransferPDUType,
IN dsp DataReqInfoType;
```

```
/* Initialise the LLC SDU for
unacknowledged data transfer
using basic link
*/
```



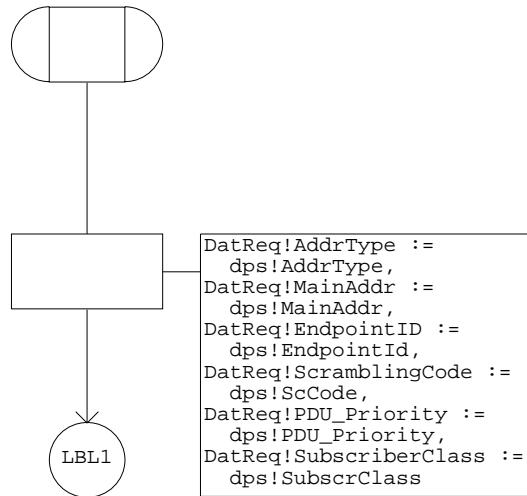
Procedure <<Process TLAB_formatter>> mk_dtreq_Prepare

1(2)

```
; FPAR
IN/OUT DatReq  TLA_DataRequestType,
IN U_Prep_PDU  MLE_U_PrepType,
IN dps DataReqInfoType;
```

```
/* Initialise the request service primitive for
   sending the U-PREPARE pdu.
*/
```

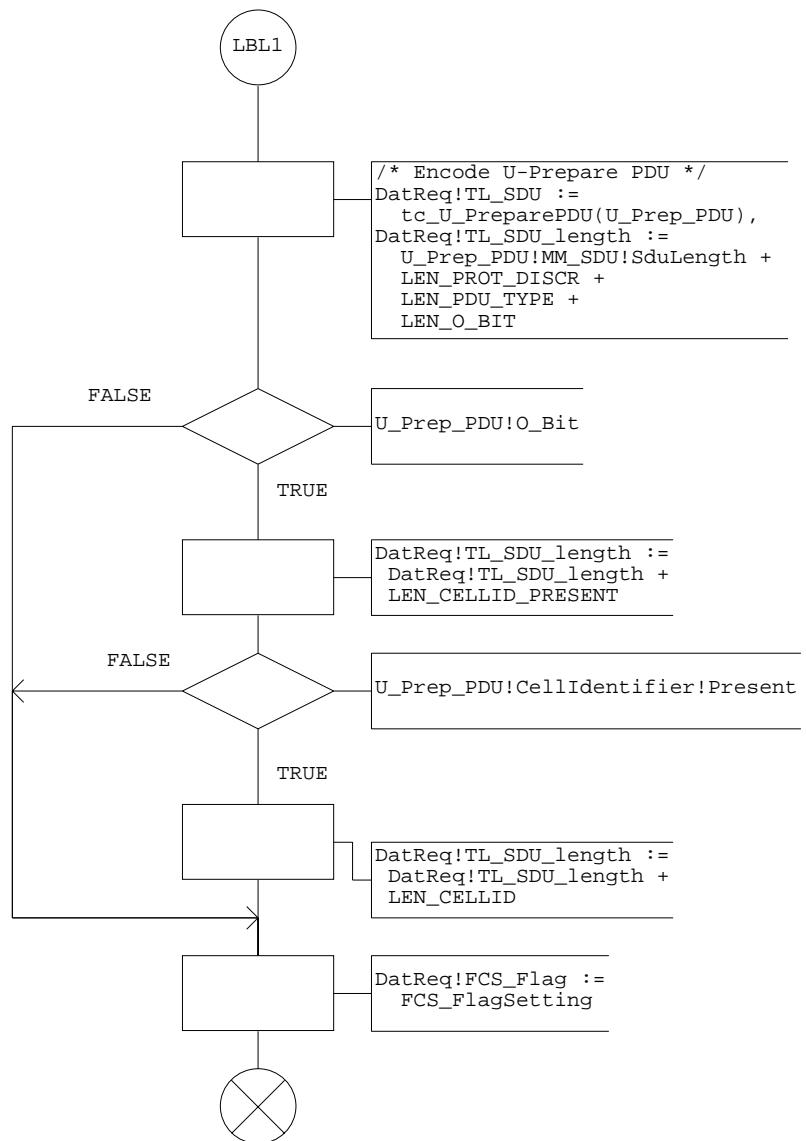
```
/* Length of the fields
   of the Prepare PDU
   in number of Bits
*/
SYNONYM LEN_PROT_DISCR Natural = 3;
SYNONYM LEN_PDU_TYPE Natural = 3;
SYNONYM LEN_O_BIT Natural = 1;
SYNONYM LEN_CELLID_PRESENT Natural = 1;
SYNONYM LEN_CELLID Natural = 5;
```



Procedure <<Process TLAB_formatter>> mk_dtreq_Prepare

2(2)

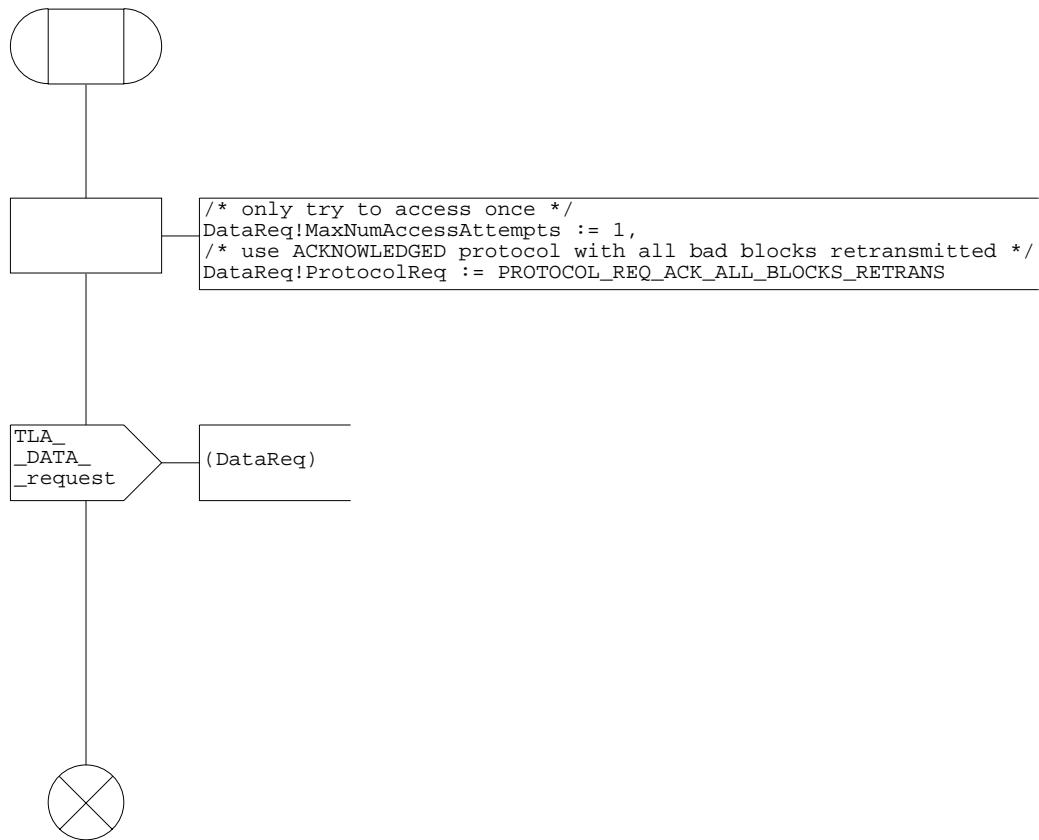
```
; FPAR
IN/OUT DatReq  TLA_DataRequestType,
IN U_Prep_PDU  MLE_U_PrepType,
IN dps DataReqInfoType;
```



Procedure Snd_AckReq

1(1)

```
; FPAR  
IN DataReq TLA_DataRequestType;
```

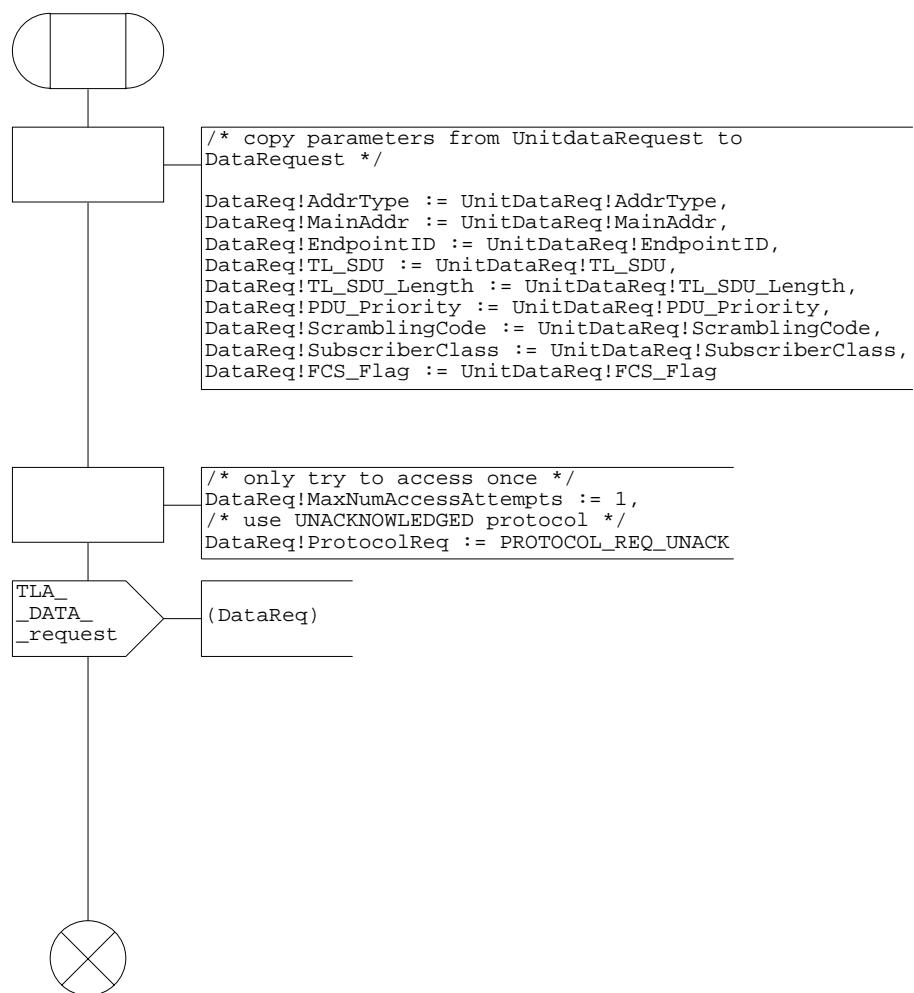


Procedure Snd_UnAckReq

1(1)

```
; FPAR  
IN UnitDataReq TLA_UnitdataRequestType;
```

```
DCL  
DataReq TLA_DataRequestType;
```

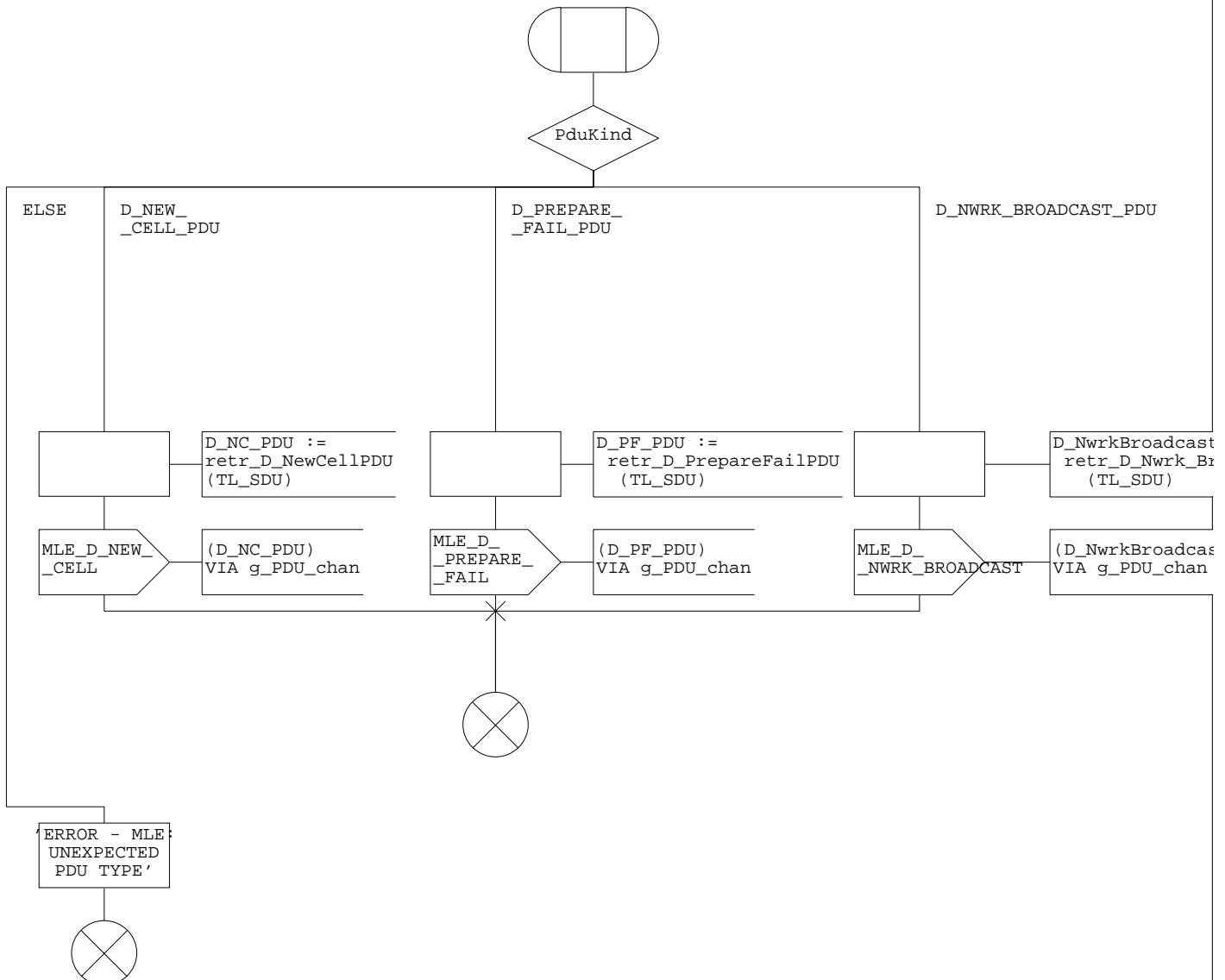


Procedure <<Process TLAB_formatter>> ExtractSelectPDU

1(1)

```
; FPAR
  IN TL_SDU TL_SDU_Type,
  IN PduKind MLE_D_PduType;
DCL
  D_NC_PDU MLE_D_NewCellType,
  D_PF_PDU MLE_D_PrepareFailType,
  D_SIN2_PDU MLE_D_SIN2Type,
  D_NwrkBroadcast_PDU MLE_D_Nwrk_BroadcastType;

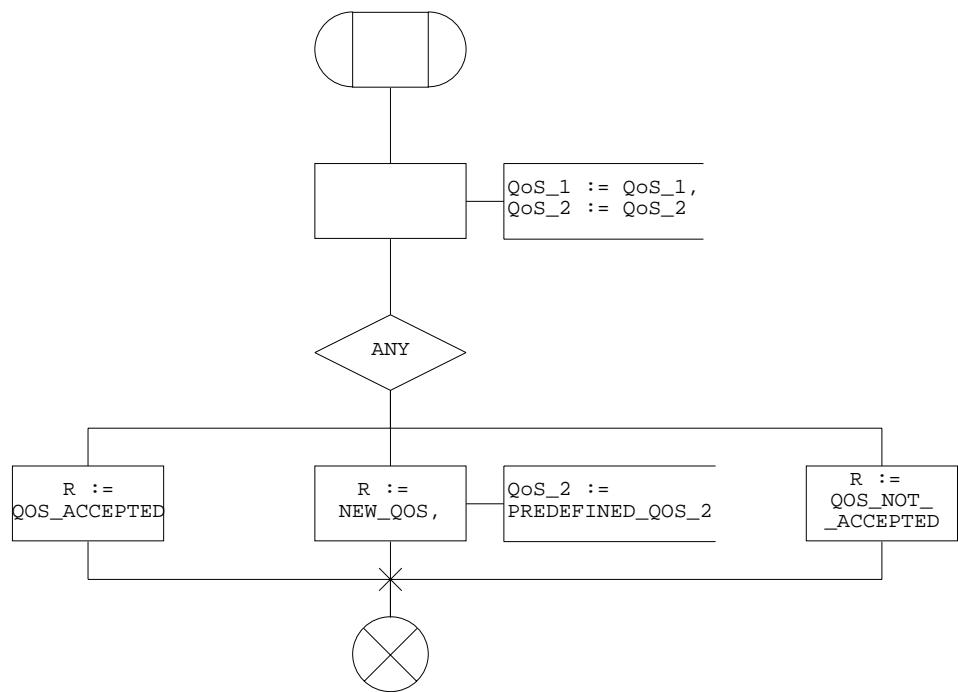
/* Retrieve the Cell selection PDU
   from the TL_SDU and deliver it
   to the Attachment Management
   process.
*/
```



Procedure <<Process TLAB_formatter>> Check_QoS

1(1)

```
;FPAR
  IN QoS_1 QoS_Type,
  IN/OUT QoS_2 QoS_Type,
  IN/OUT R QoS_ResultType;
```

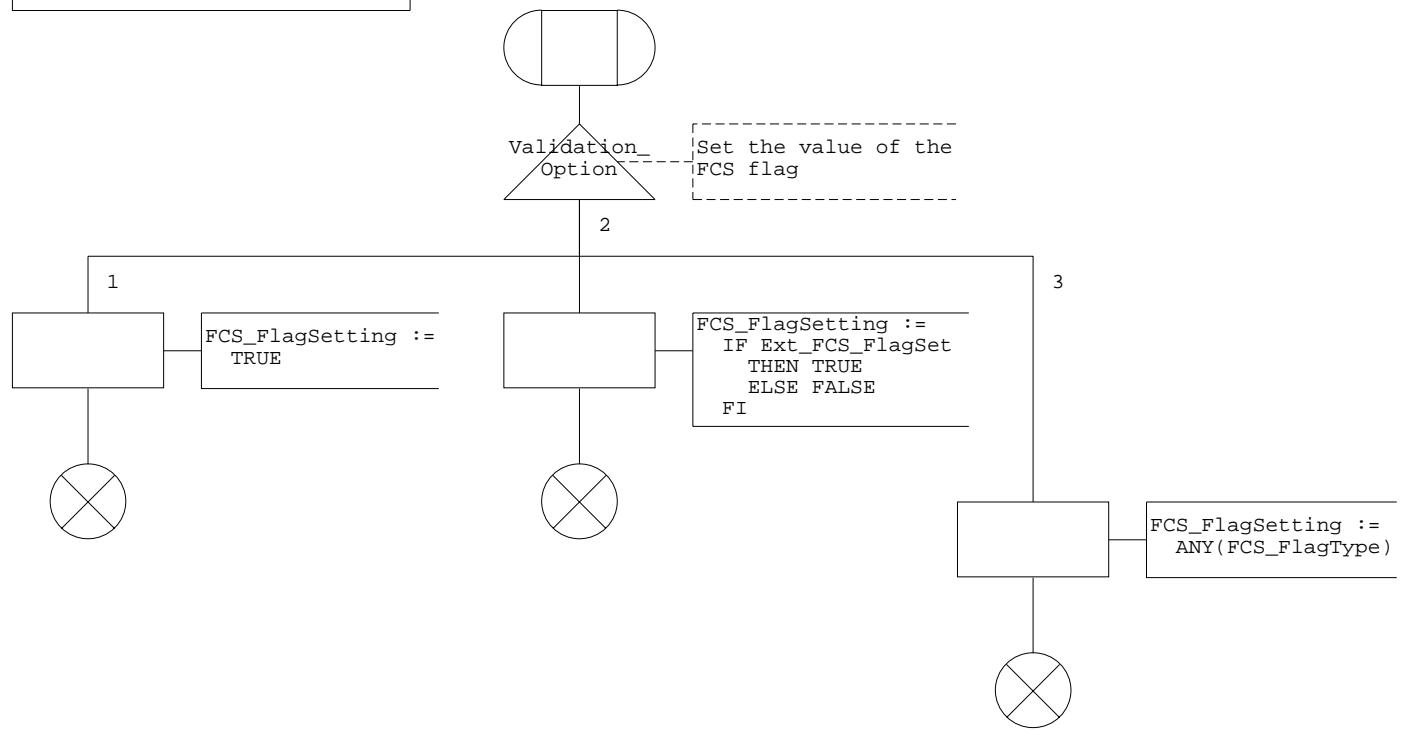


Procedure <<Process TLAB_formatter>> Init_FCS_Flag

1(1)

```
; FPAR  
IN/OUT FCS_FlagSetting FCS_FlagType;
```

```
/* Set the FCS flag according  
to the validation option  
*/
```

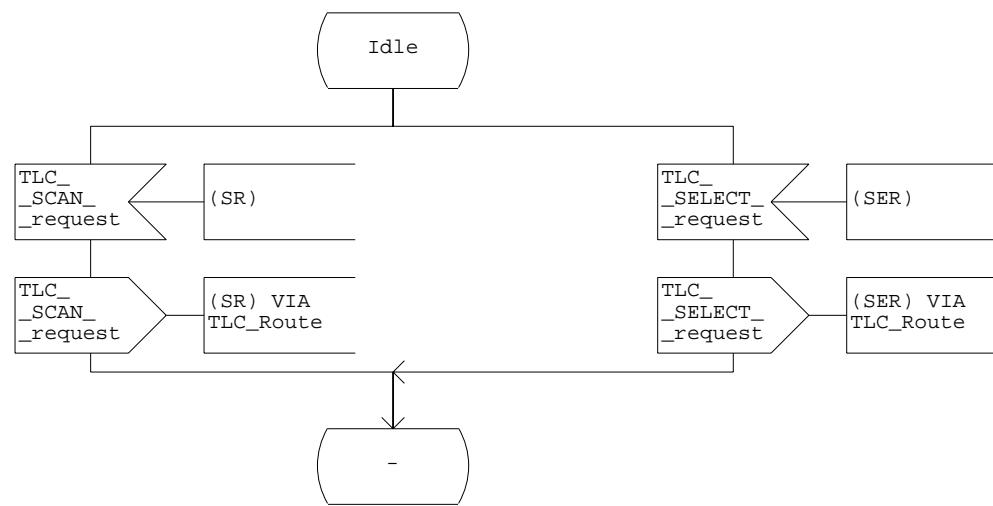
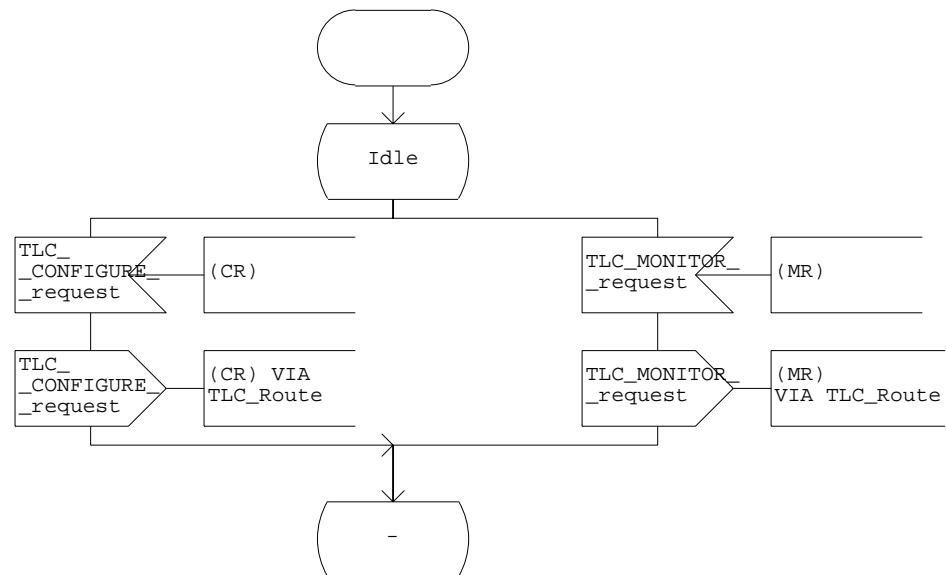


Process TLC_formatter

1 (3)

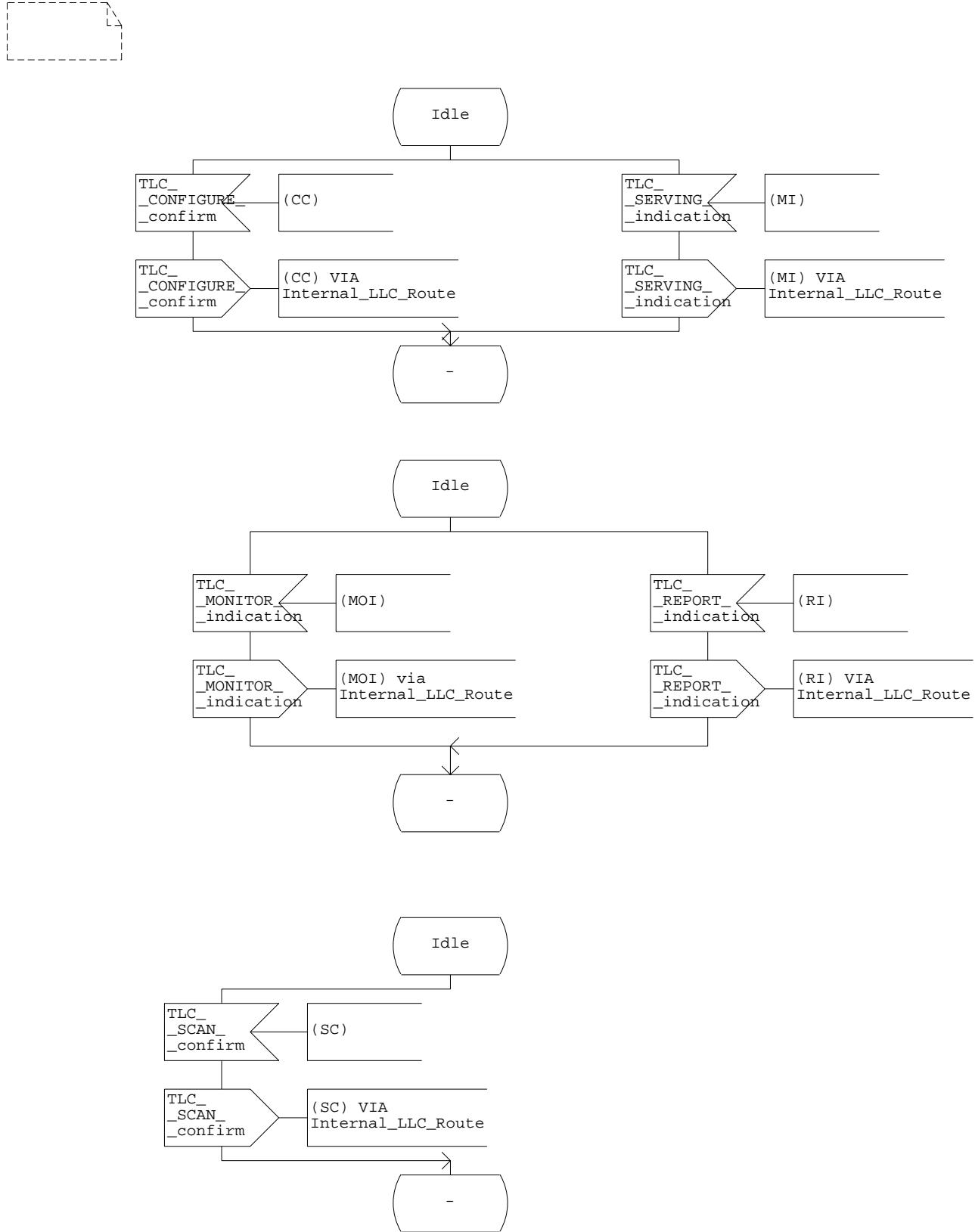


```
DCL
  CC  TLC_ConfigureConfirmType,
  CR  TLC_ConfigureRequestType,
  MI  TLC_ServingIndicationType,
  MOI TLC_MonitorIndicationType,
  MR  TLC_MonitorRequestType,
  RI  TLC_ReportIndicationType,
  SC  TLC_ScanConfirmType,
  SRI TLC_ScanReportIndicationType,
  SR  TLC_ScanRequestType,
  SEC TLC_SelectConfirmType,
  SER TLC_SelectRequestType;
```



Process TLC_formatter

2 (3)



Process TLC_formatter

3 (3)

