T E C H N I S C H E U N I V E R S I T E I T D E L F T

Faculteit der Elektrotechniek

Aard : Afstudeerverslag
Omvang : 118 bladzijden Bijlagen (deel II)
Datum : oktober 1987

Lab/Afd. : Laboratorium voor AUTOMATISCHE VERKEERSSYSTEMEN

Opdrachtnr. : Go/1986/2

Auteur : R.A.F.J. LIE

Titel : The MINEX's layer 3 specification.

Opdrachtnr. : Go/1986/2

Korte inhoud: Dit verslag beschrijft de MINEX's (Mini Exchange) laag 3 processen en signaleringssamenwerking. MINEX is een ISDN-huiscentrale tussen de ISDN-centrale en de abonnee-eindapparatuur.

APPENDIX A: THE TIME SEQUENCES DIAGRAMS

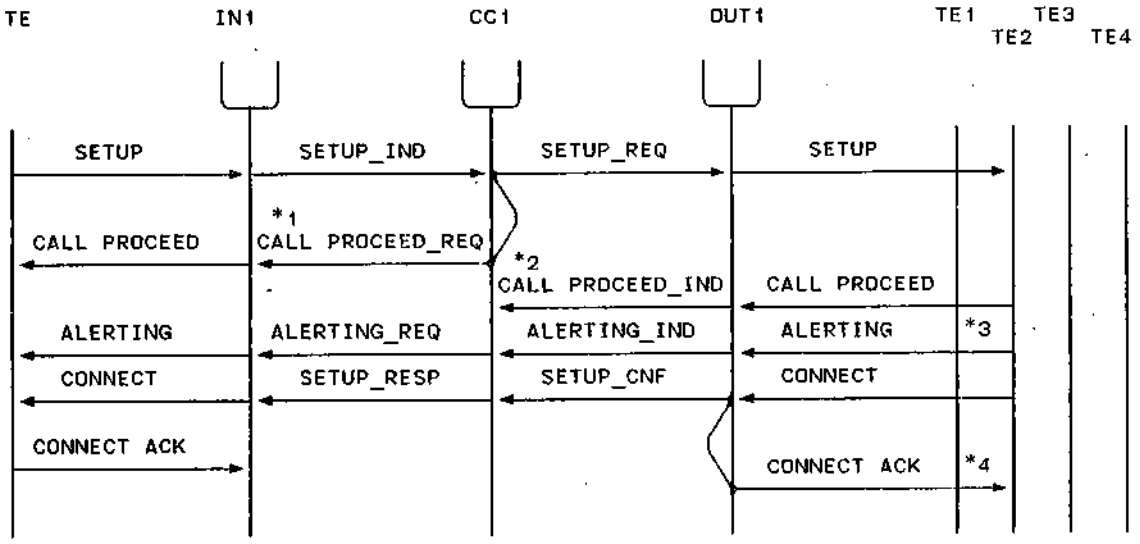


figure.1 Successful call, scenario 1
(internal call - model)

Note: *. An internal call will be a point to point connection; the called party in this figure is TE2, the calling party might be TE1 or TE3 or TE4.

*1. A CALL PROCEED_REQ is generated if channels are available for this new call.

*2. The CALL PROCEED received by OUT1 from TE2 is transferred to the CC1 as CALL PROCEED_IND (switch of MINEX shall be activated); since the calling TE has already received a CALL PROCEED (*1), CC1 should not retransmit CALL PROCEED again to IN1.

*3. As an ALERTING progresses the call further, it is retransmit to the calling TE.

*4. TE2 received CONNECT ACK as response to sending CONNECT.

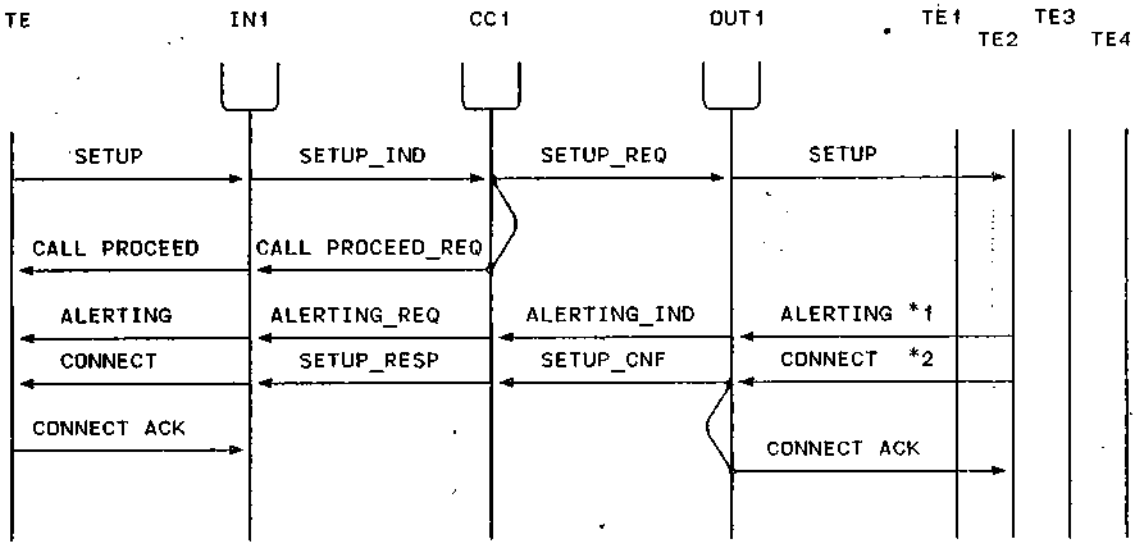


figure.2 Successful call, scenario 2
(internal call - mode1)

Note: *1. It is allowed to respond with ALERTING as first message, this ALERTING should reach the originating TE (switch of MINEX shall be activated by ALERTING_IND).

*2. It is also allowed to respond with CONNECT as first message, this CONNECT shall reach the originating TE (switch of MINEX will be activated).

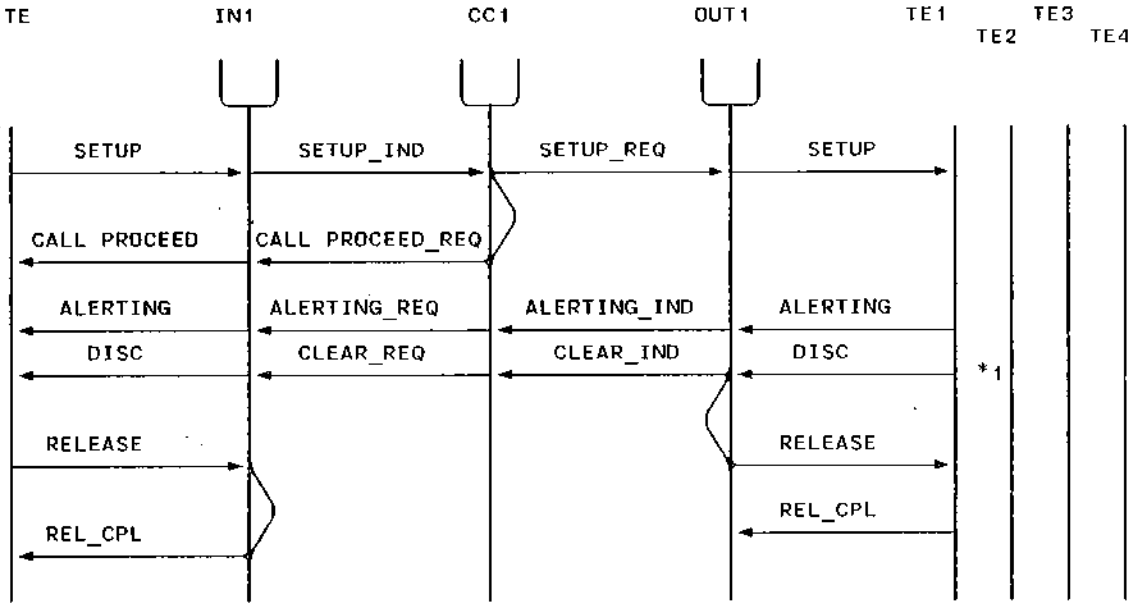


figure.3 Unsuccessful call, scenario 3
(internal call - model, terminated TE initiated disconnection.)

Note: *1. With DISC, TE1 (the called party) is notifying it's intention to release the channel and call reference for this call; OUT1 process generates CLEAR_IND to CC1, which activates the release procedure in IN1 process. A RELEASE is sent to the originator of the clearing procedure (i.e TE1).

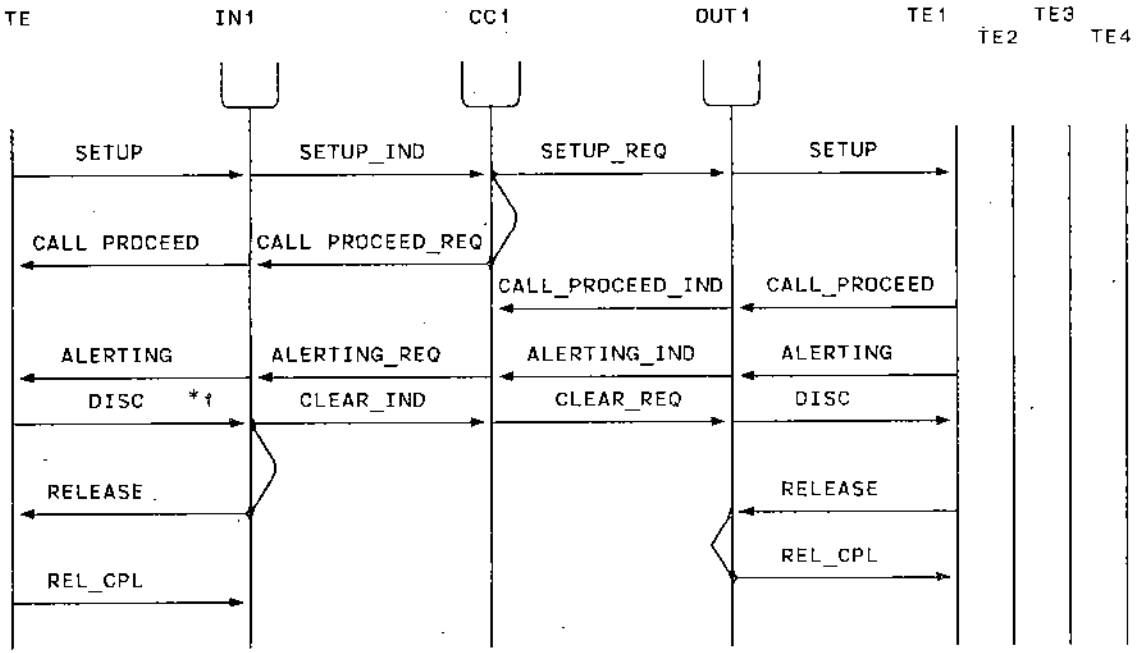


figure.4 Unsuccessful call, scenario 4
(internal call - model, originating TE disconnects before call is active)

Note: *1. With DISC, TE (the calling party) is notifying it's intention
to release the channel and call reference for this call;
IN1 process generates CLEAR_IND to CC1, which activates the release
procedure in OUT1 process.

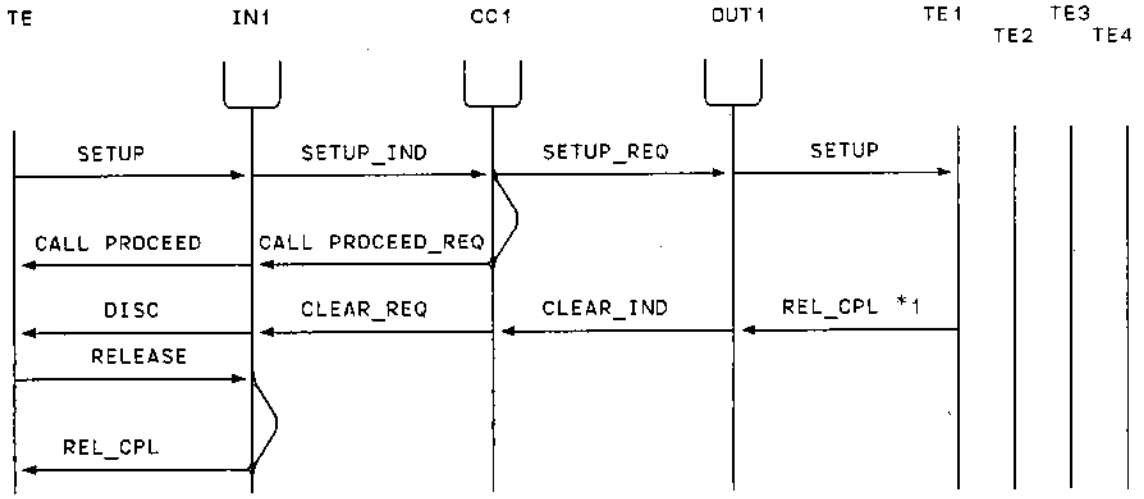


figure.5 Call failure, scenario 5
(internal call - model)

Note: *1. Assumption has been made that the call is offered to a particular TE (e.g TE1), if TE sent REL_CPL's as response to the SETUP, this should be a call reject indication for the OUT1 process; In this case a CLEAR_IND is sent to the CC1 process.
At the originating side, IN1 process activates the release procedure.

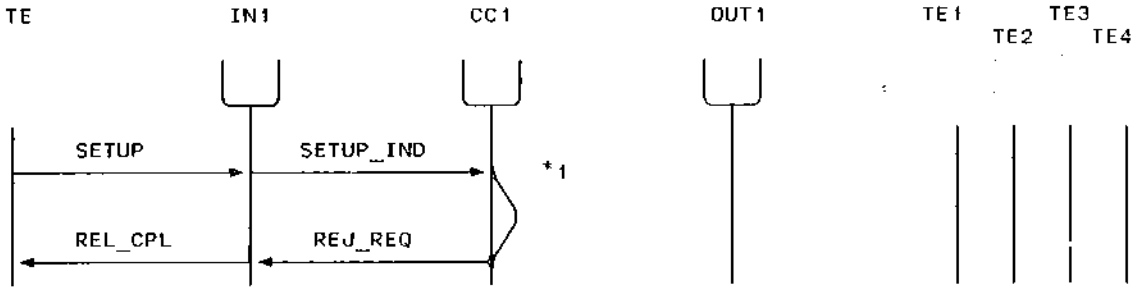


figure.6 Call failure, scenario 6
(internal call - model)

Note: *1. If the MINEX reached its maximum capacity (i.e if both B channels on both S interfaces are used), or a B channel required by the TE is not available; a new call setup will be rejected by means of REJ_REQ.

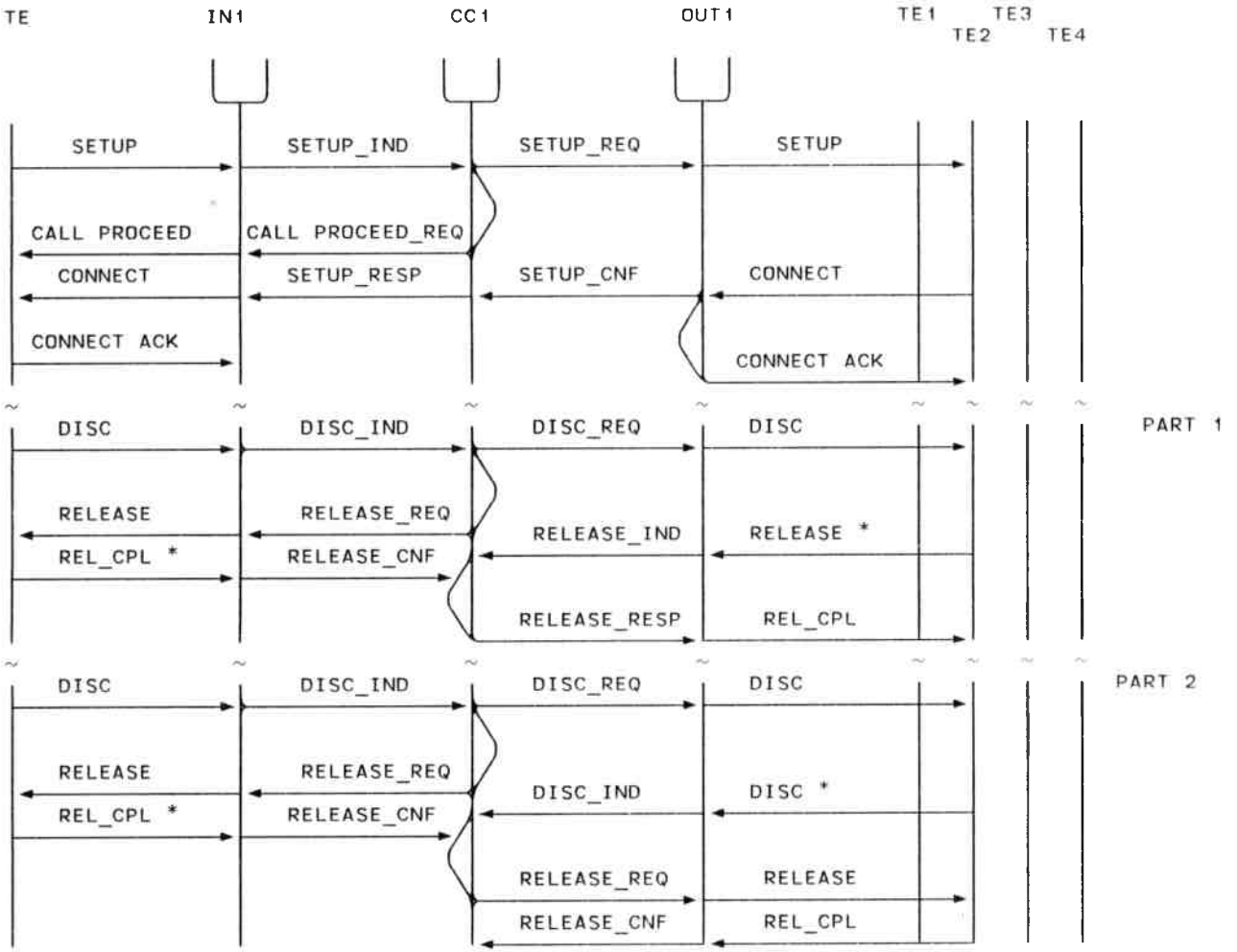


figure.7 Call clearing, scenario 7
(internal call - model, clearing initiated by originating TE.)

Note: *. Situations are outlined when call is active, messages which are marked with a '*' are not necessarily following the time sequences as shown in the figure, an example of the first set: REL_CPL from TE (of originating side) might be generated before a RELEASE is transmitted from TE1 (of the terminating side).

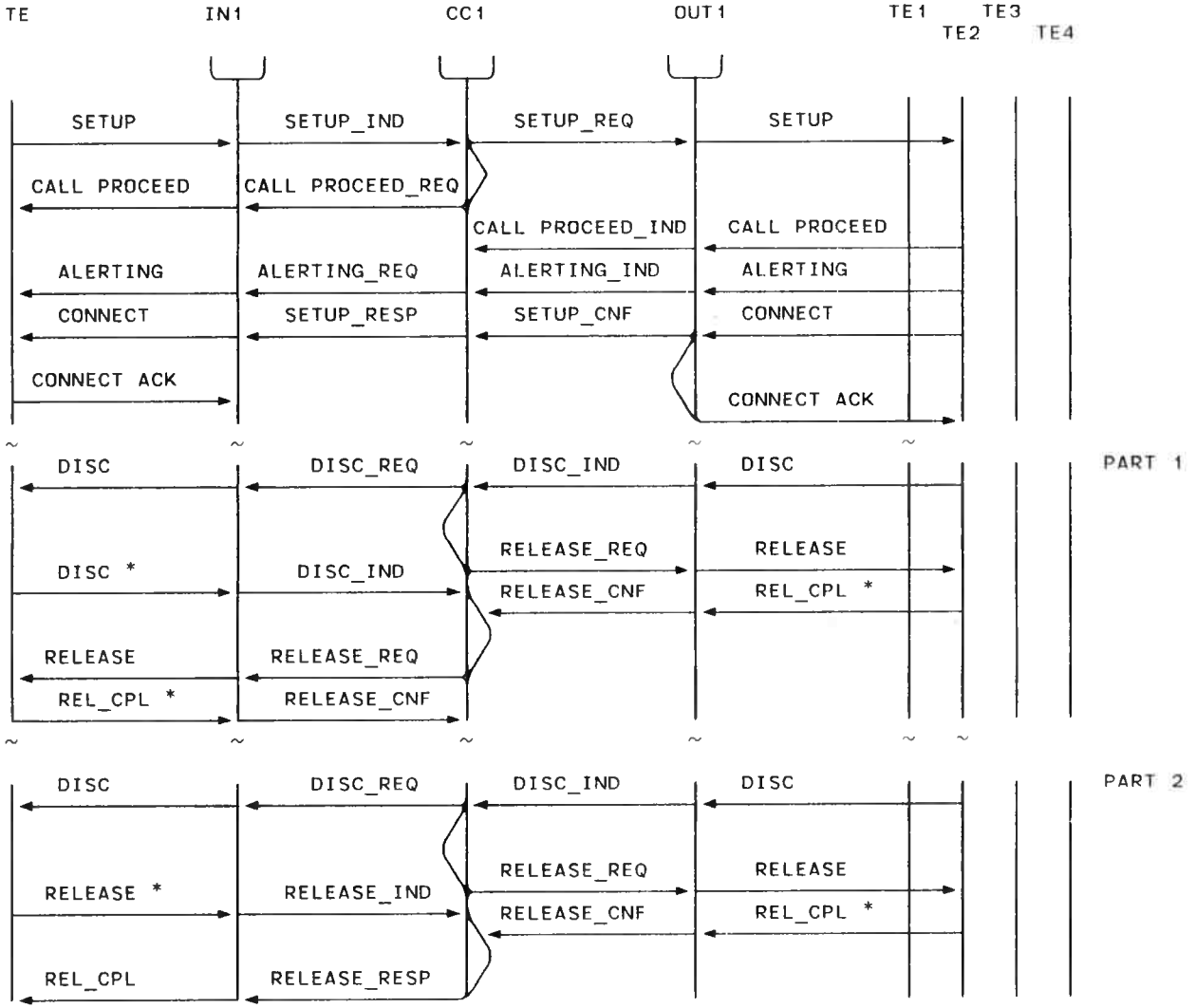


figure.8 Call clearing, scenario 8
(internal call - model, clear by terminating TE)

Note: *.Situations are outlined for an active call, messages which are marked with a '*' are not necessarily following the time sequences as shown in the figure, an example of the first set: REL_CPL from TE (of originating side) might be generated before a RELEASE is transmitted from TE1 (of the terminating side).

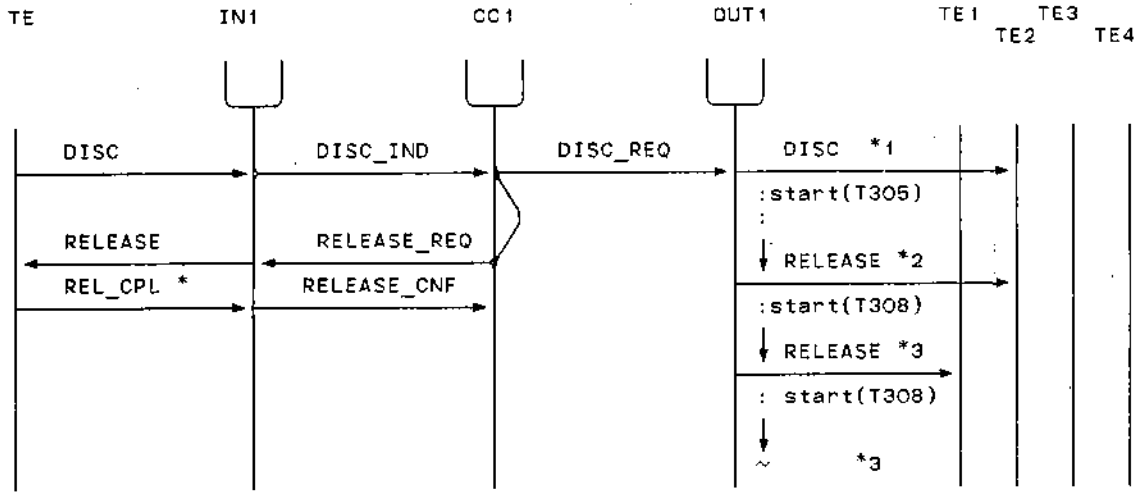


figure.9 Call clearing, scenario 9
(internal call - model, clearing procedure as one of the TE's is "not" responding .)

Note:

This scenario outlines the situation when one of the TE's (either calling or called) does not respond within a time interval.

*1. As the OUT1 (or IN1)process transmits a DISC to the TE, it starts the T305 timer, within this interval the involved OUT1 (or IN1) process is expecting a DISC or RELEASE as response.

*2. If T305 expires before the OUT1 (or IN1) process receives the DISC or RELEASE message, the involved process transmits a RELEASE to the TE, and starts the T308 timer. At this state information transfer on B channel is not reliable (since channel/ time slots might be disconnected by either IN1 or OUT1 process); process conerned is now expecting a REL_CPL to complete the clearing.

*3. If T308 expires without the involved process receiving a REL_CPL, T308 will be restarted. Clearing will be terminated if within the second 308 time interval a REL_CPL is received. If T308 expired for the second time without any response, OUT1 (or IN1) process will assuming that the clearing has been completed; thereby makes the B channels available and releases the CR (Call Reference; CR mentioned here is that issued by the MINEX's CC1 process).

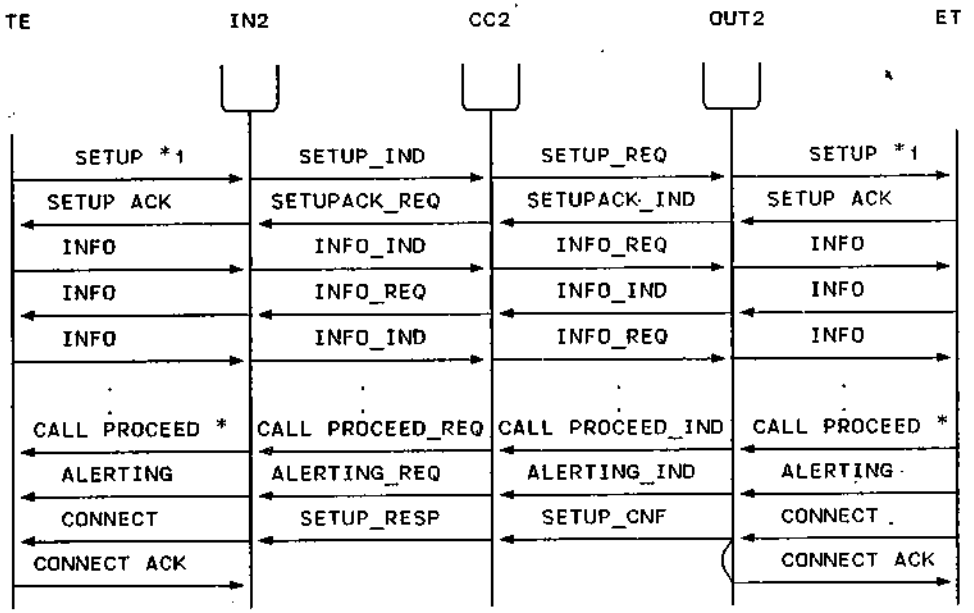


figure.10 Successful call, scenario 10
(overlap sending, external call, calling party side, mode 2)

Note: *1. The TEI and CR used by OUT2 are mapped values, they are not those used by TE (CC2 manages the fixed value mapping).
*. Channel ID info. element in CALL PROCEED, ALERTING and CONNECT messages at the OUT2-ET interface is not necessarily the same as those which are used at the TE-IN2 interface (the choice of CH_ID depends on the availability of the channels on the S and U interfaces, information about of these interfaces is kept in respectively the IN2 and OUT2 processes.

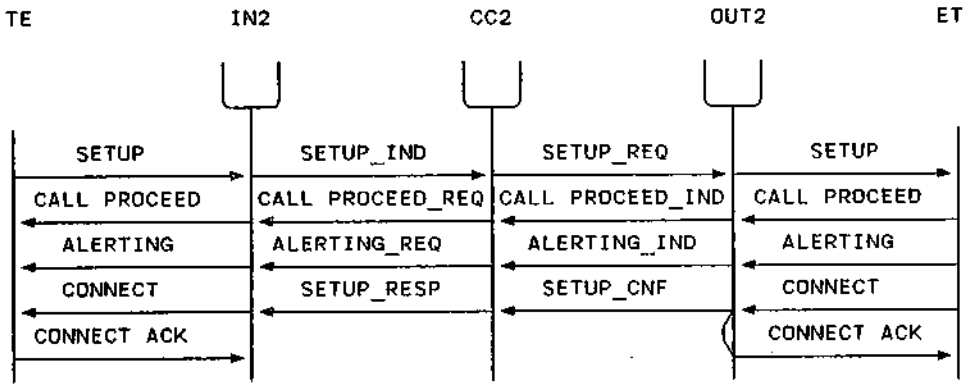


figure.11 Successful call, scenario 11
(en bloc sending, external call, calling party side, mode 2)

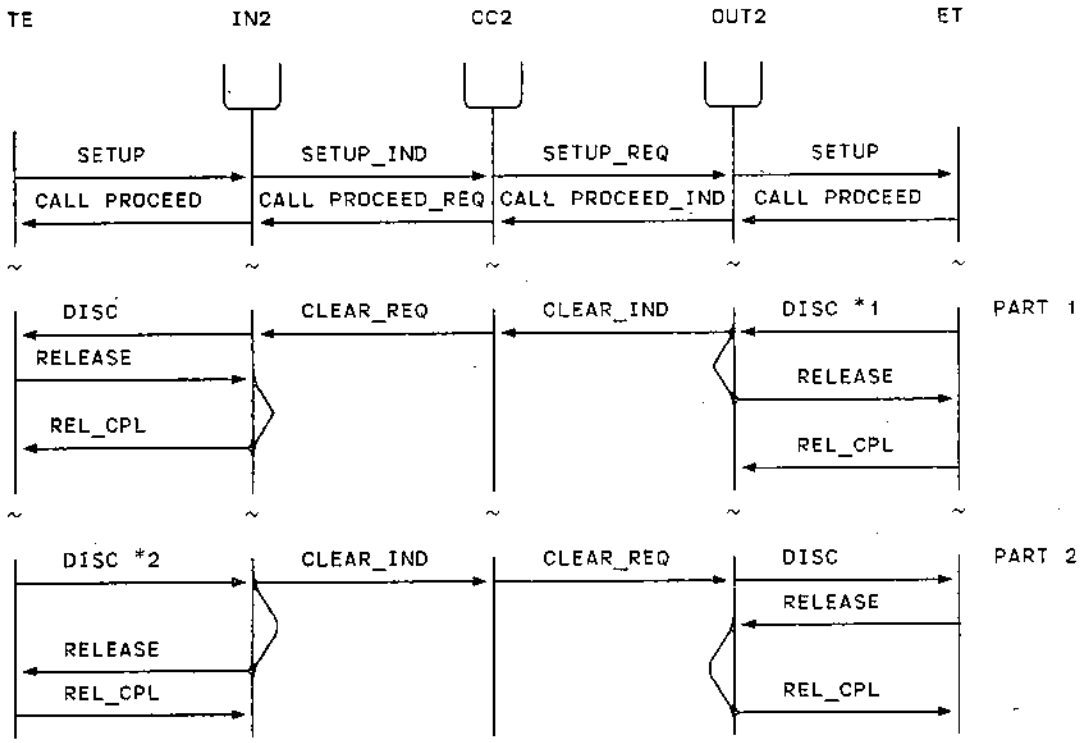


figure.12 Call failure, scenario 12
(external call, mode 2, not in active call state)

Note: *1. Clearing request by ET before call is active.
*2. Clearing request by TE before call is active.

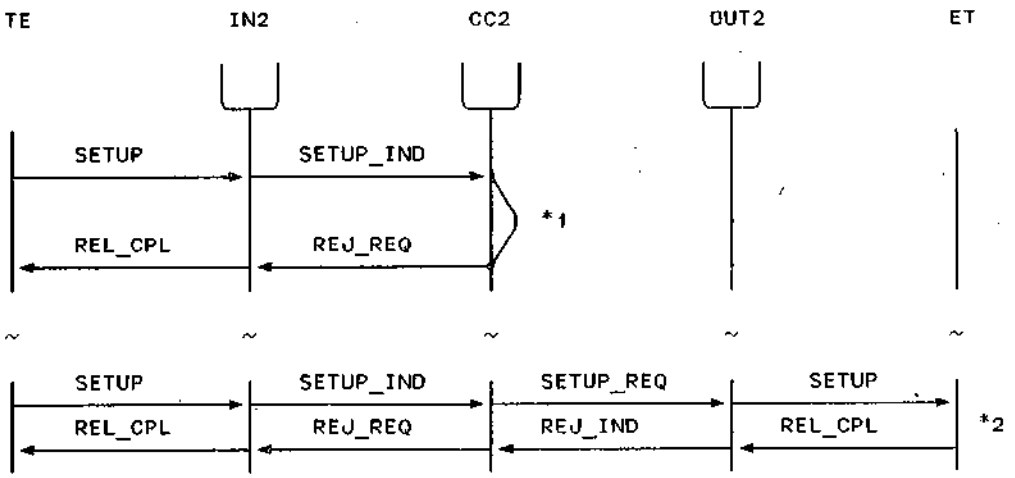


figure.13 Call failure, scenario 13
(external call, calling party side, mode 2)

Note: Failure due to
*1. resources shortage
*2. ET rejects the call

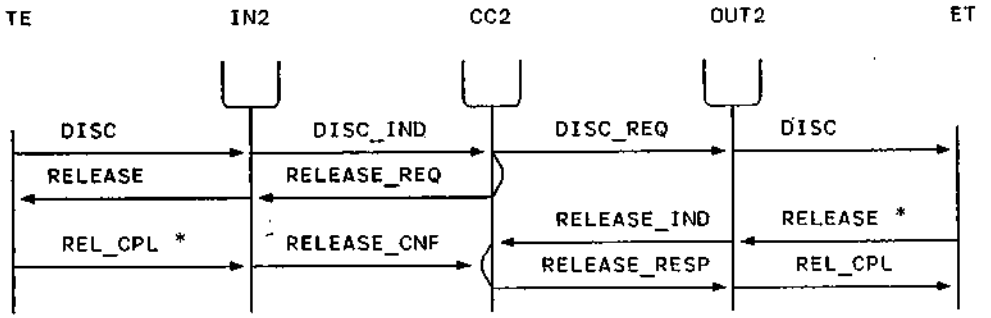


figure.14 Call clearing by user, scenario 14
(external call, calling party side, mode 2)

Note: Clearing of an active call.
*. The sequences of RELEASE and REL_CPL can be changed, since the response time of these messages is not predictable.

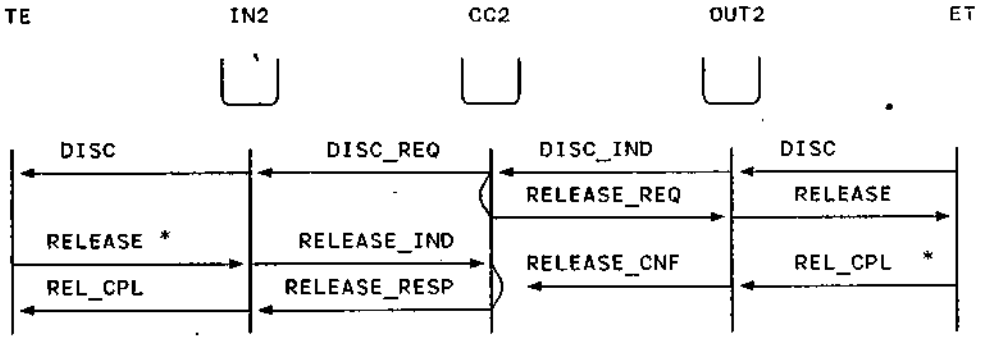


figure.15 Call clearing by ET, scenario 15
(external call, calling party side, mode 2)

Note: Clearing of an active call.
*. As stated in figure.14 the sequence of RELEASE and REL_CPL can be changed.

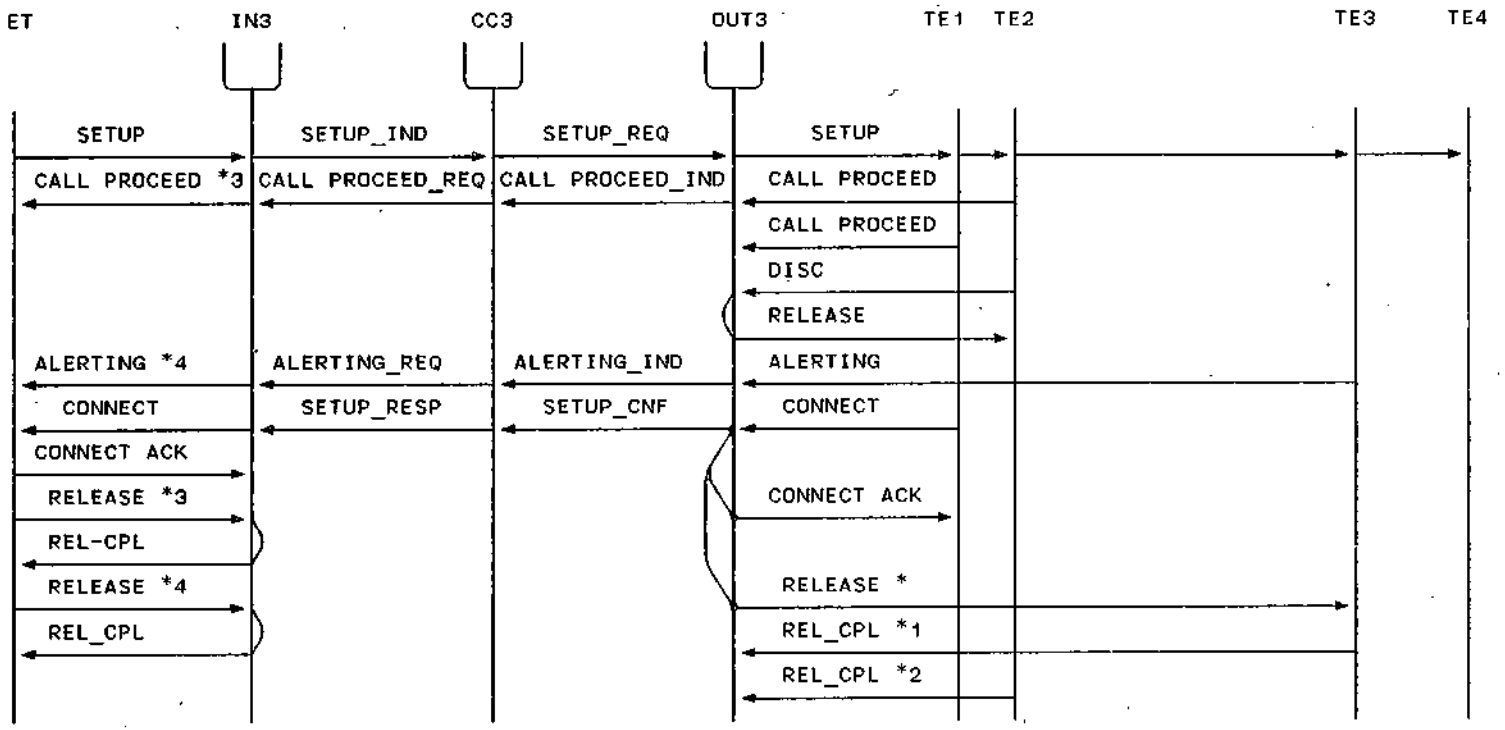


figure.16 Successful call, scenario 16
(external call, called party, mode 3)

Note: * . Release non selected TE's procedure, TE's which are still contending for the call will be notified to give up the attempt through a RELEASE message (see also figure 17).
*1. As response to the release non selected TE procedure.
*2. TE2 completes the disc procedure initiated by itself.

*3,*4. TE1 sends the CONNECT, but ET has also kept status of the TE2 and TE3; due to the release non selected TE's procedure, ET will send RELEASE's to TE2 and TE3. These messages will be received by the IN3 process and work up internally, as response a REL_CPL should be sent back to the ET.

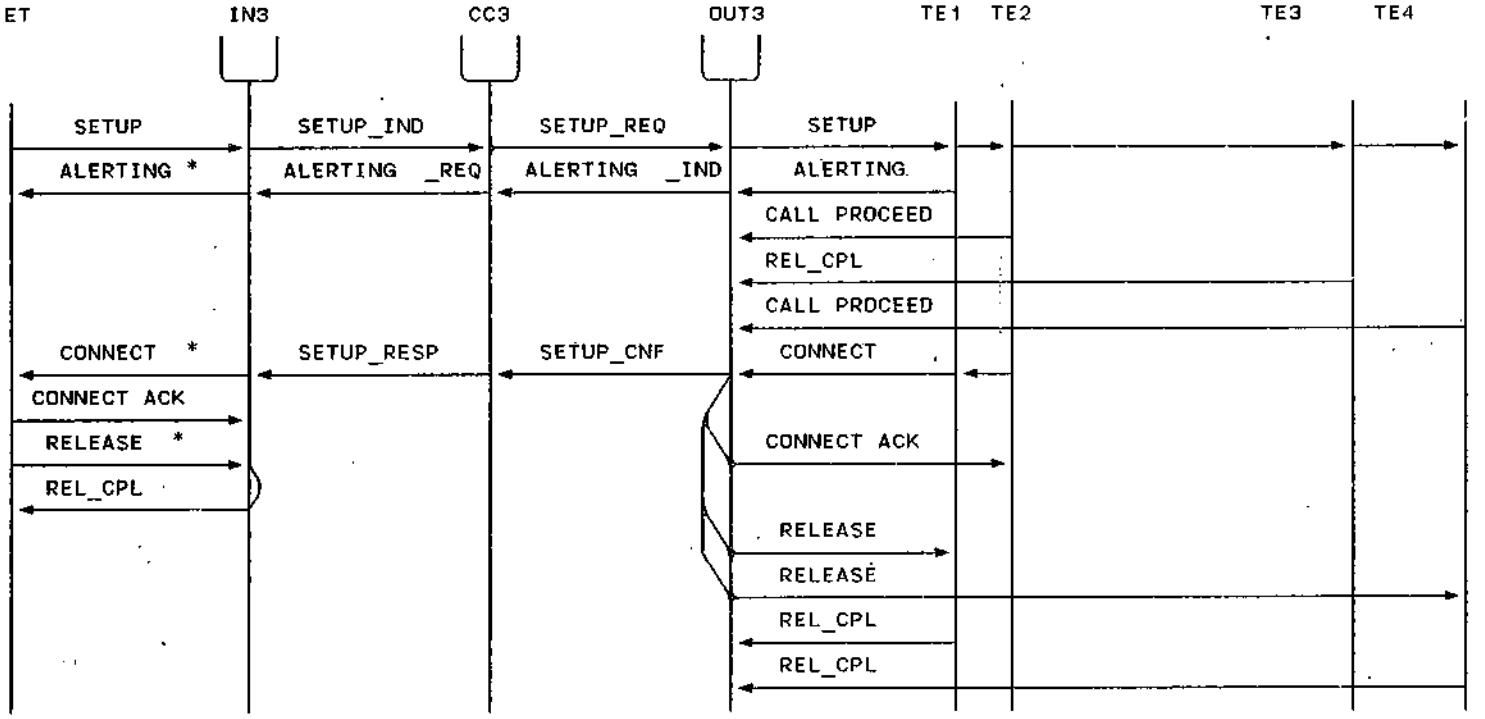


figure.17 Successful call, scenario 17
(external call, called party, mode 3)

Note: * . As in figure.16 CONNECT ACK for TE2 which sent CONNECT, while RELEASE is supposed to be sent to TE1.

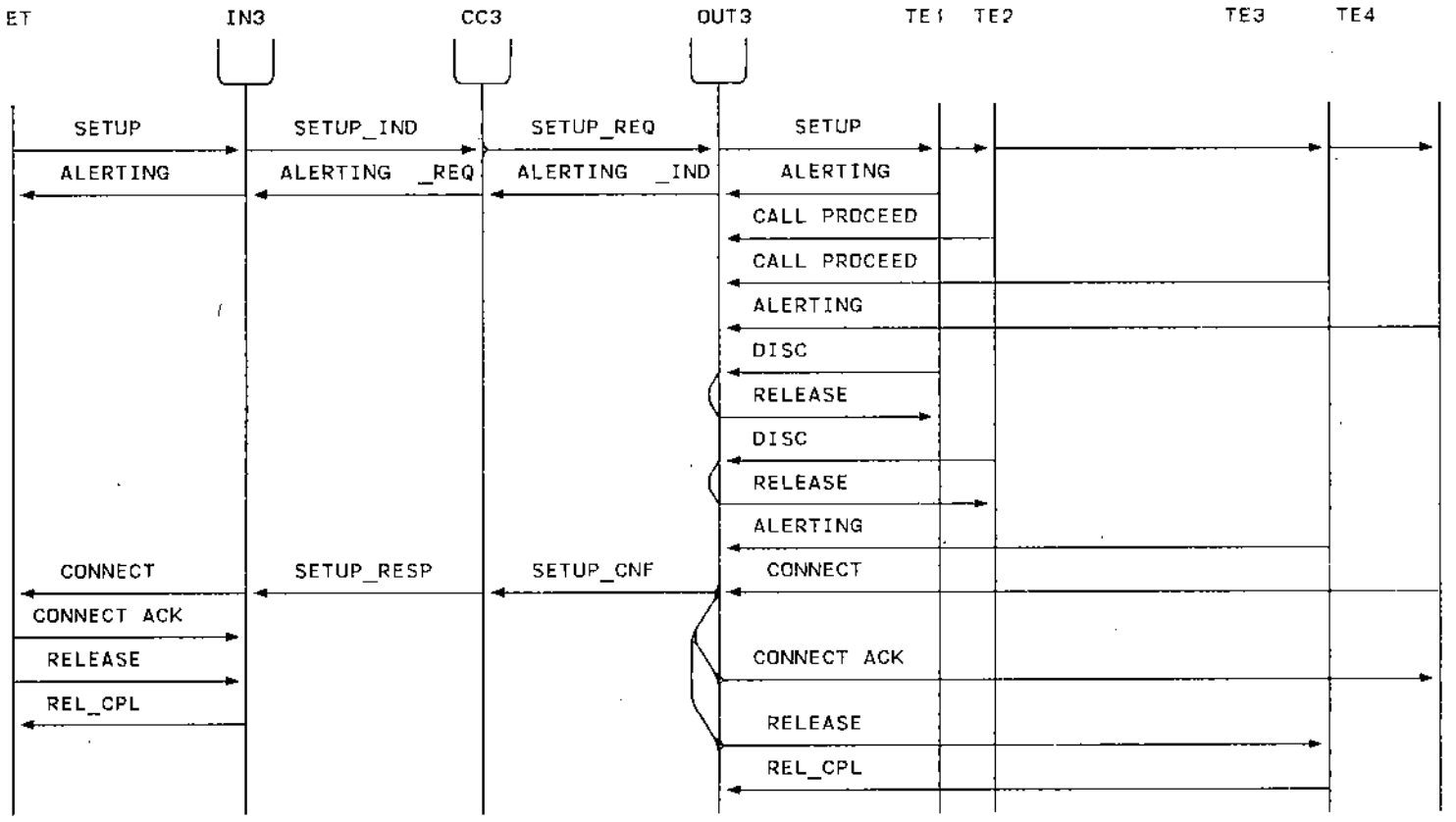


figure.18 Successful call, scenario 18
(external call, called party, mode 3)

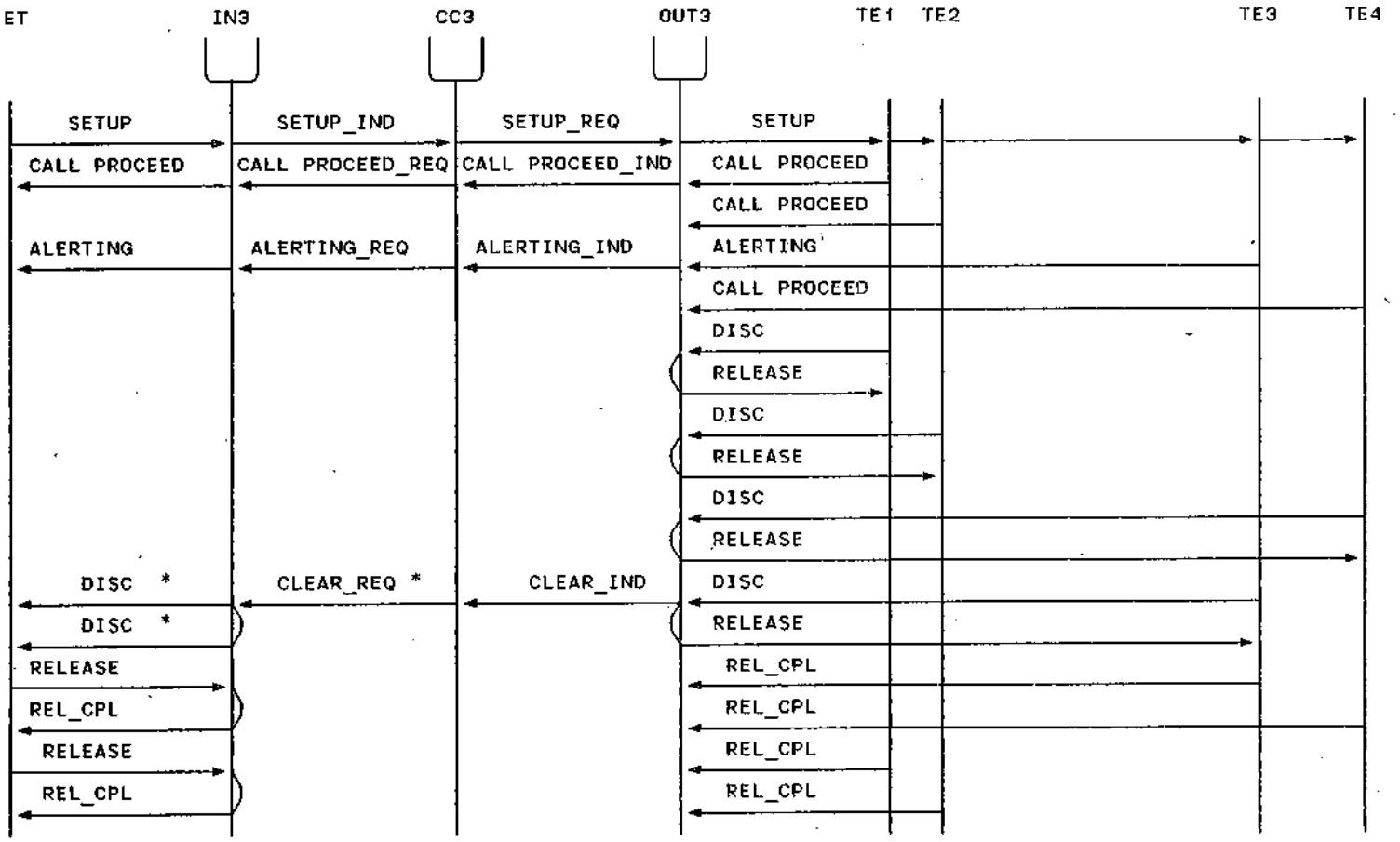


figure.19 Call failure, scenario 19
(external call, called party, mode 3)

Note: * . When IN3 receives CLEAR_REQ from CC3, DISC should be send to ET.
Since for this call, the ET is aware of the existence of TE1 and TE3(by means of the CALL PROCEED and ALERTING messages); the IN3 should send two DISC's to the ET, each with it's own information element (e.g TEI and CR).

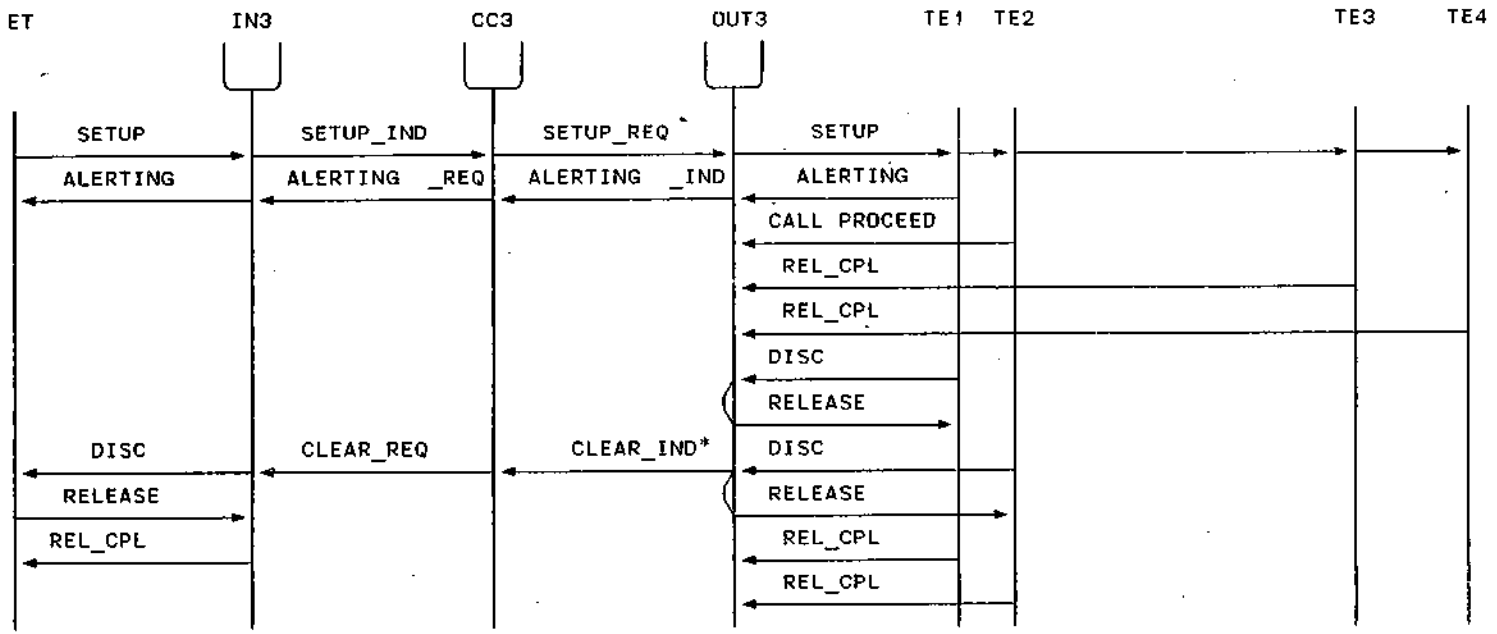


figure.20 Call failure, scenario 20
(external call, called party, mode 3)

Note: * . An CLEAR_IND is generated by OUT3, as the OUT3 process has determined that all TE's have refused to accept the call (either by DISC or REL_CPL messages).

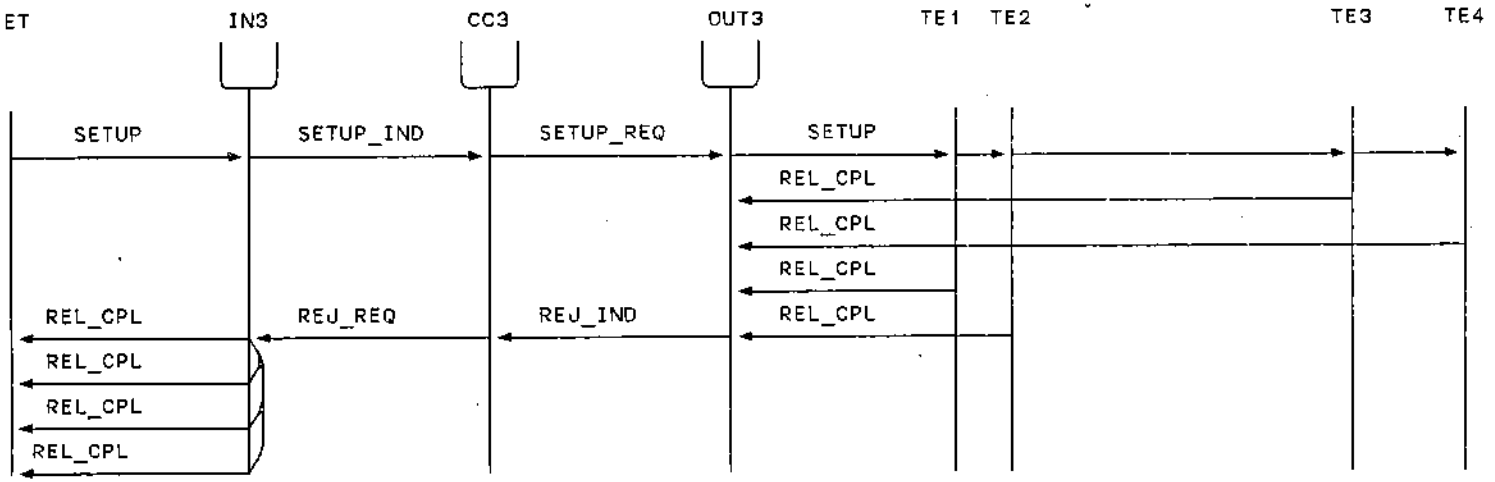


figure.21 Call failure, scenario 21
(external call, called party, mode 3)

Note: REJ_IND is generated by OUT3 as all TE's have sent REL_CPL.

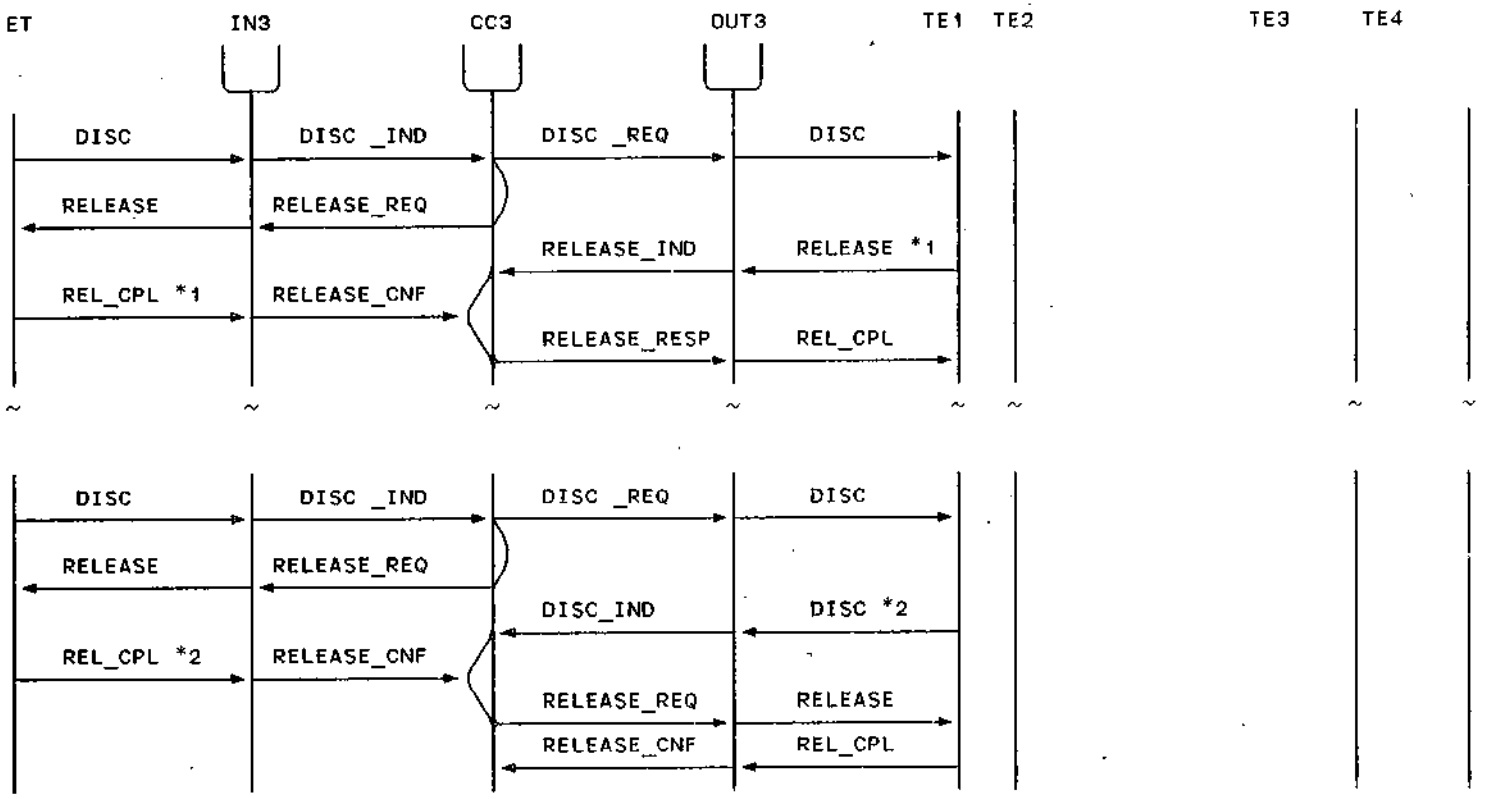


figure.22 Clearing of an active call by the ET, scenario 22
(external call, called party, mode 3)

Note: *1. The sequence of RELEASE and REL_CPL can be changed.
*2. The sequence of DISC and REL_CPL is interchangeable.

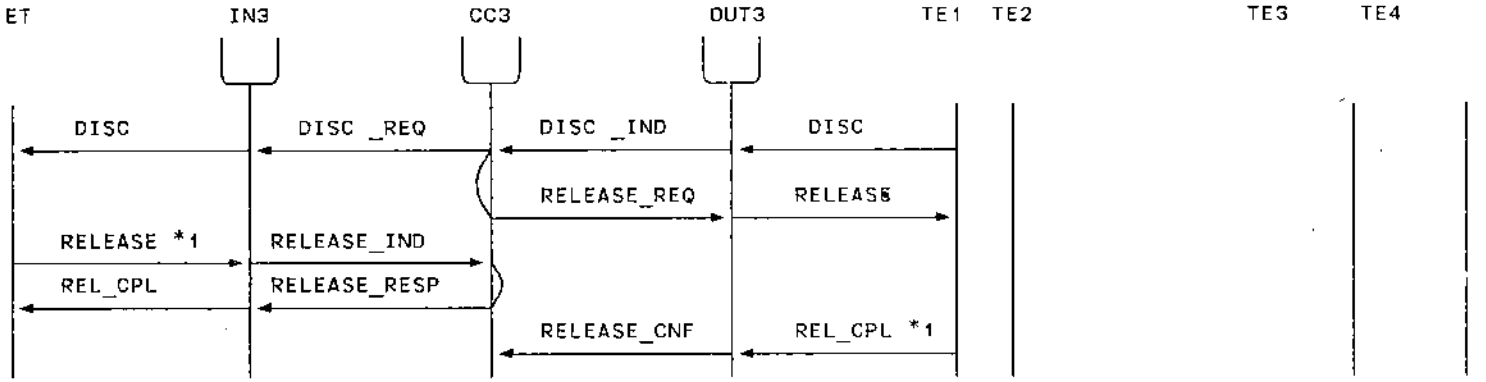
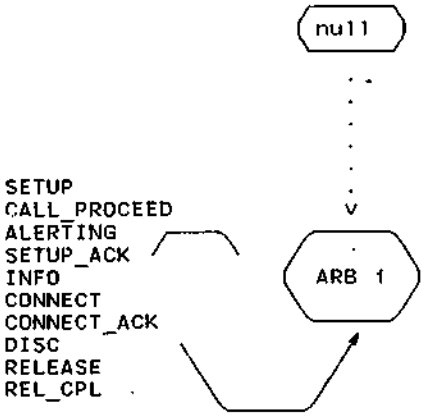


figure.23 Cearing of an active call by the TE, scenario 23
(external call, called party, mode 3)

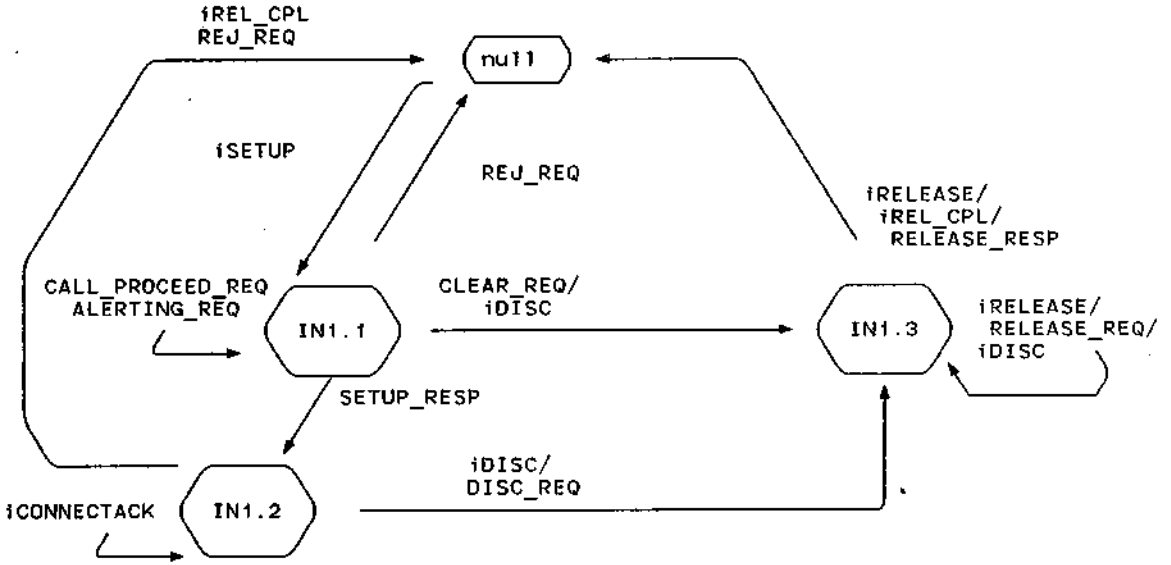
Note: *1. The sequence of RELEASE and REL_CPL is interchangeable.

APPENDIX B: THE STATE DIAGRAMS OF THE NT12 L3's PROCESSES



ARB_process

Figure 1. ARB's state diagrams



IN_process model

Figure 2. IN1's state diagrams

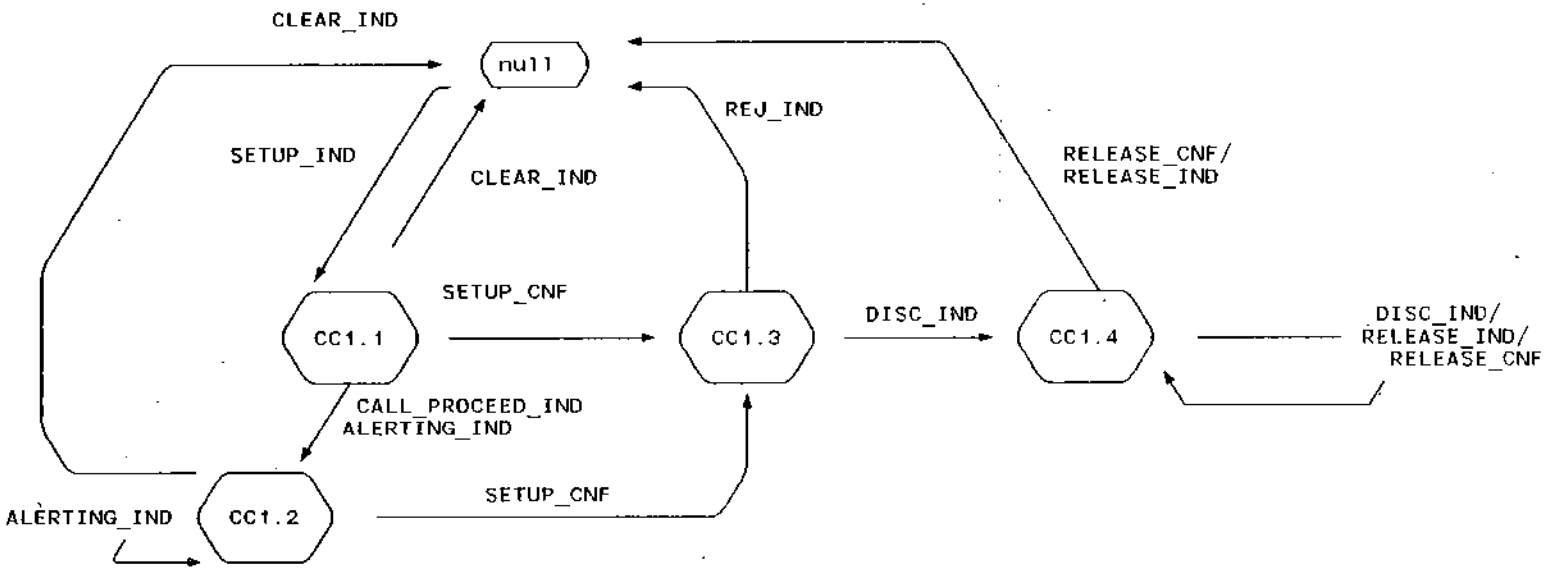
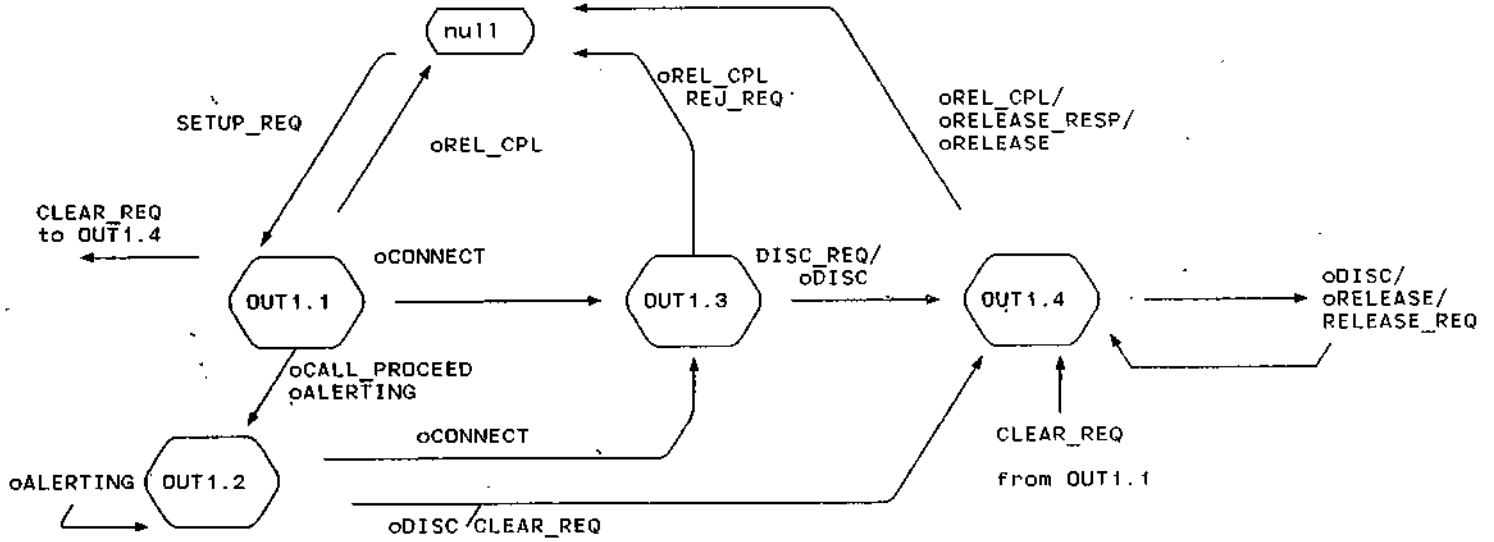
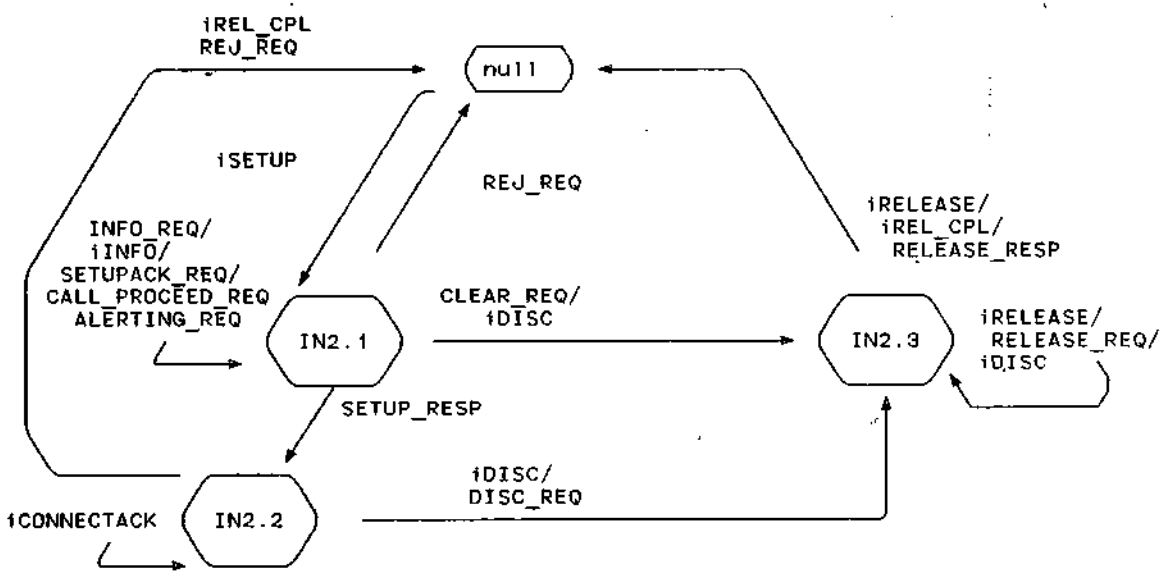


Figure 3. CC1's state diagrams



OUT_process model

Figure 4. OUT1's state diagrams



IN_process mode2

Figure 5. IN2's state diagrams

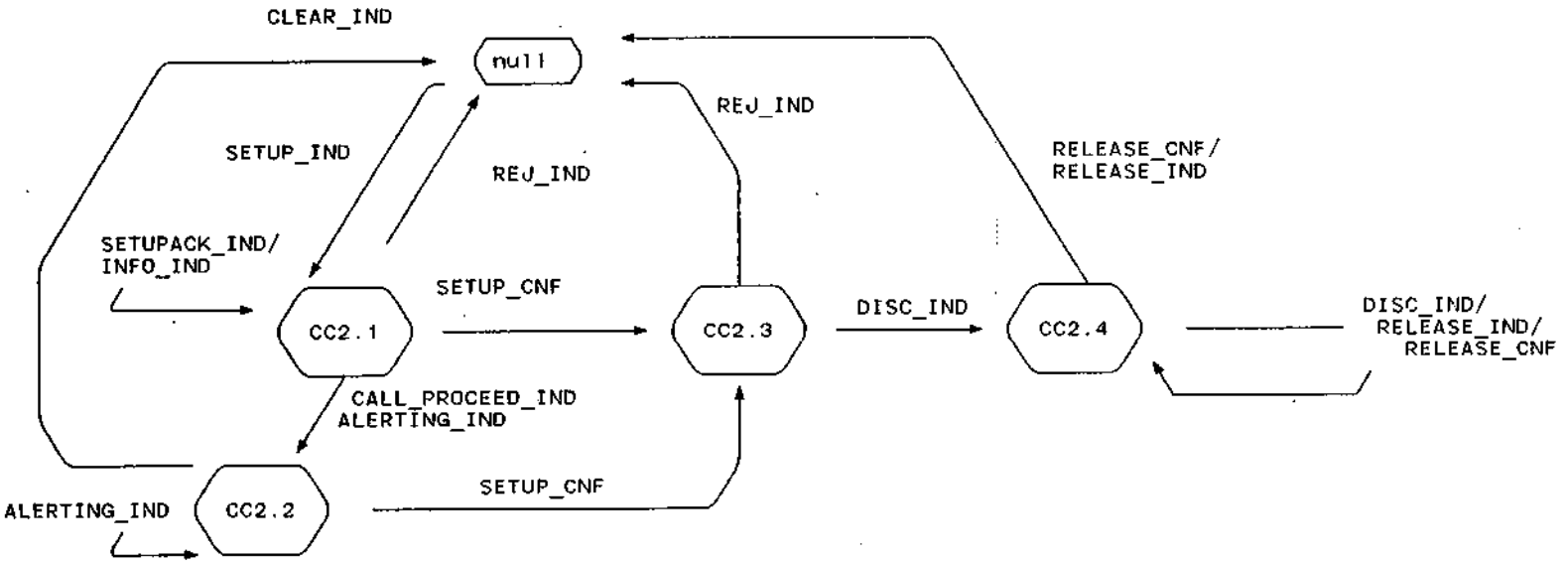
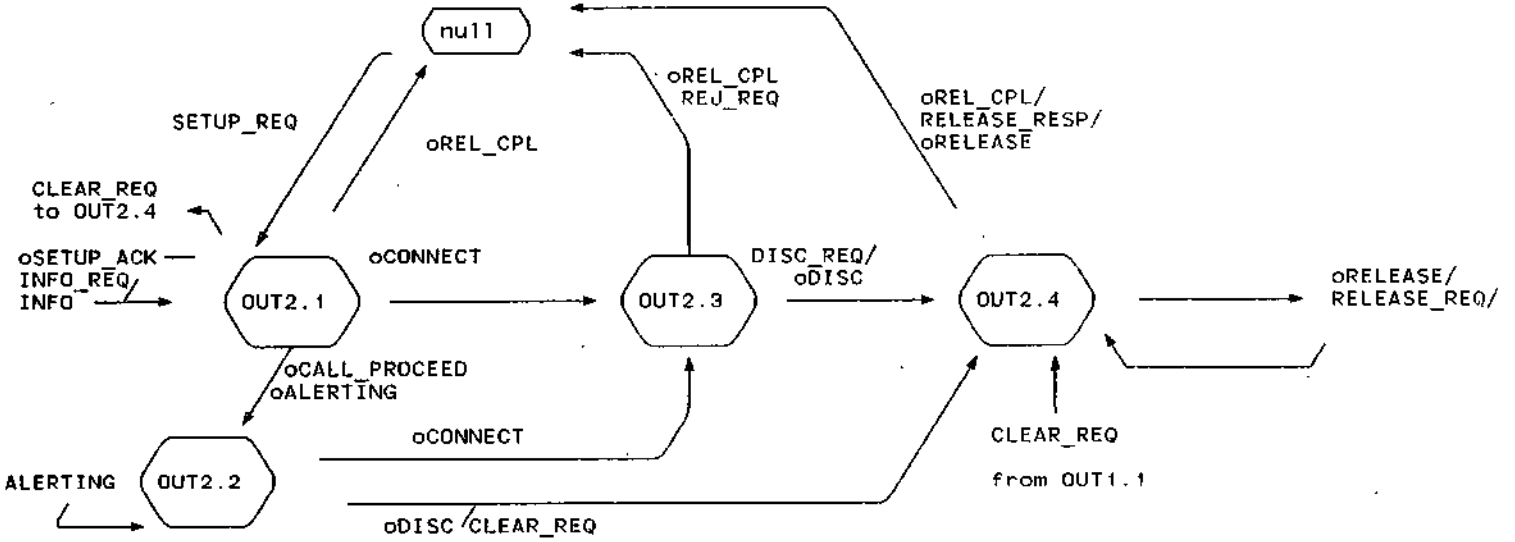
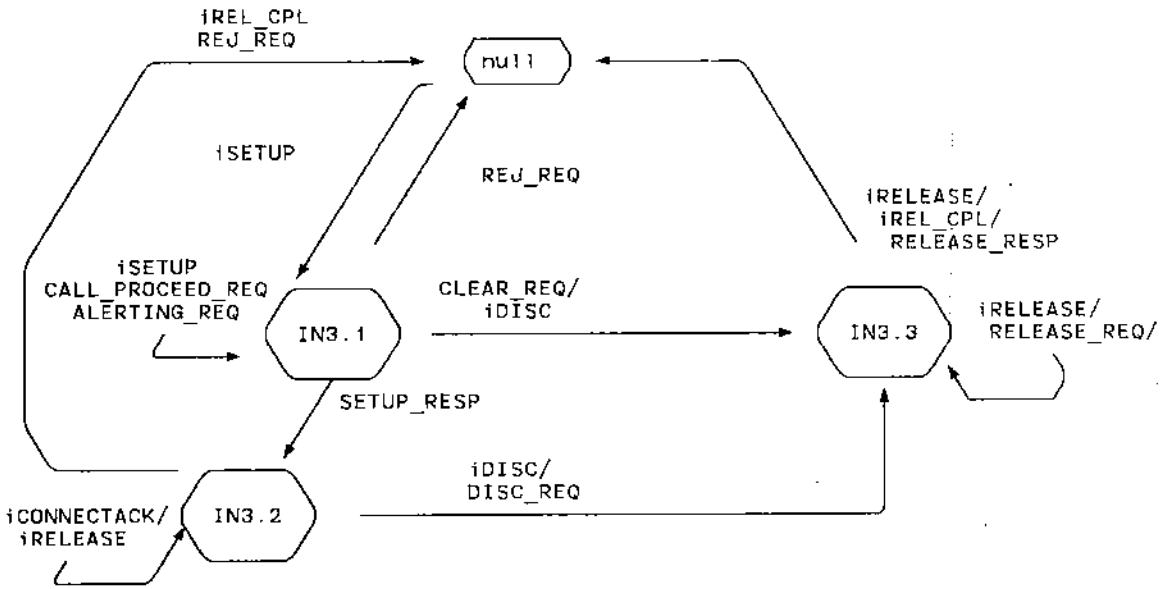


Figure 6. CC2's state diagrams



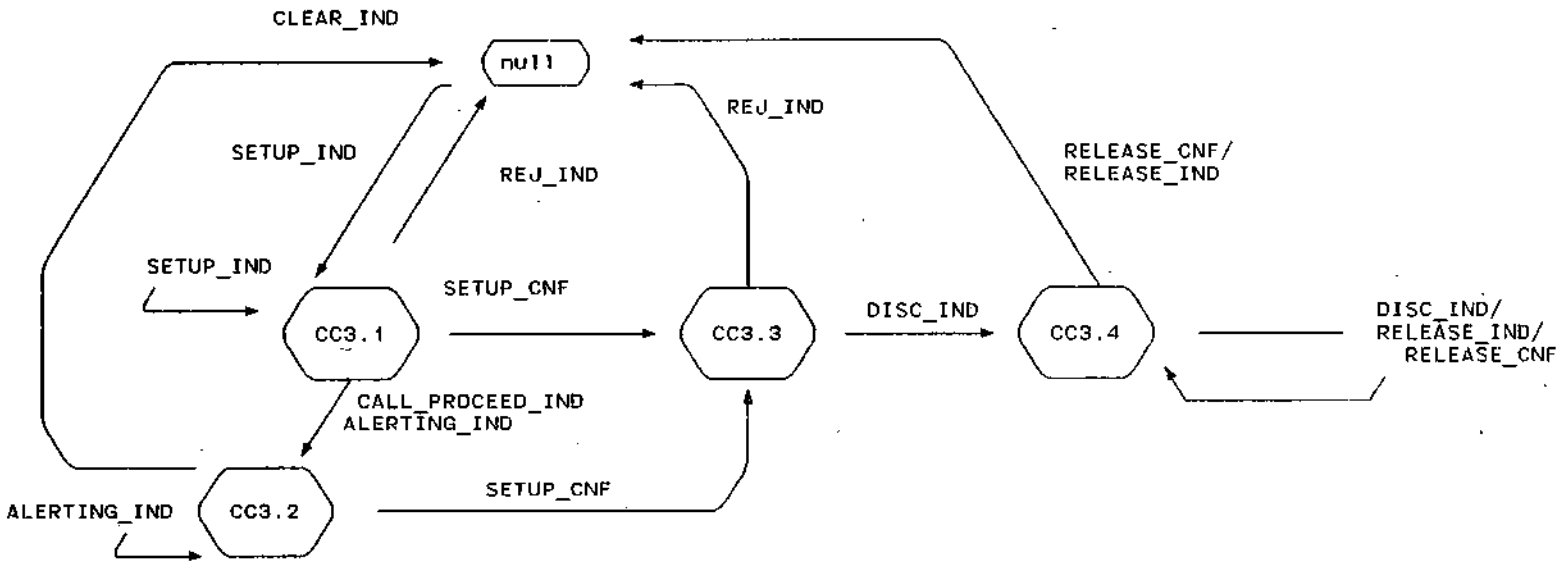
OUT_process mode2

Figure 7. OUT2's state diagrams



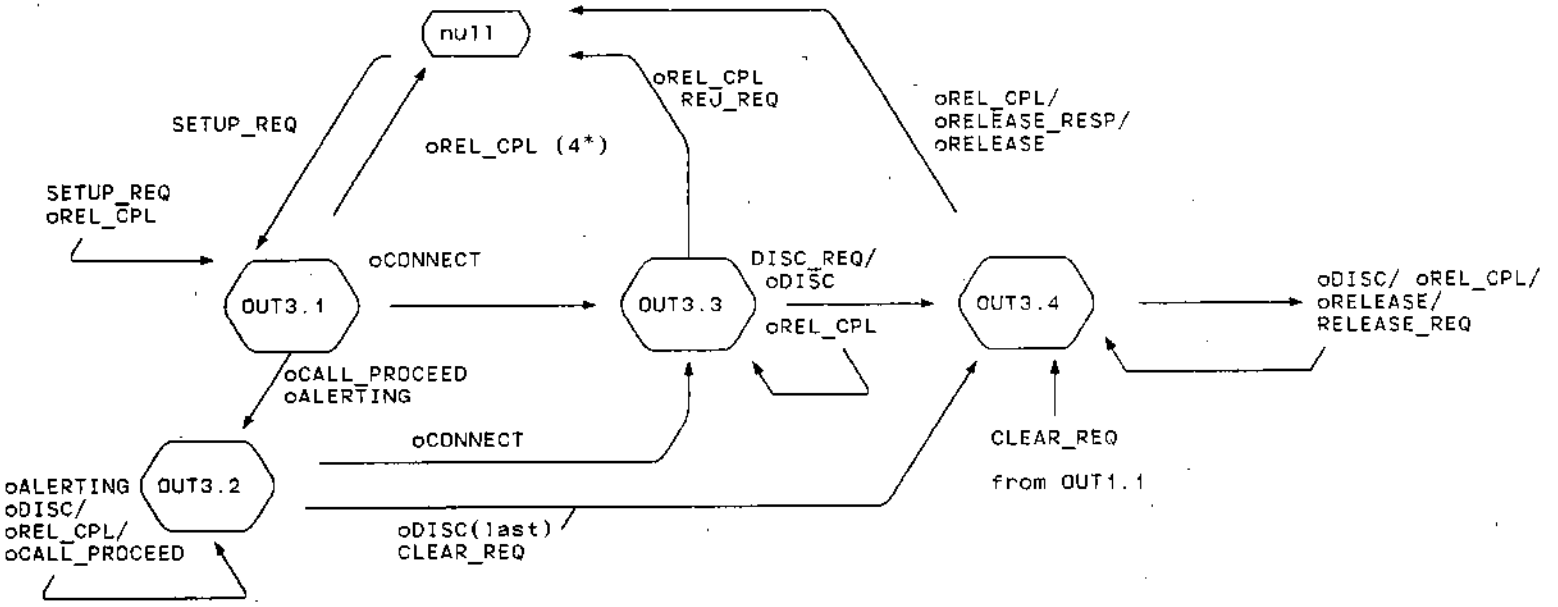
IN_process mode3

Figure 8. IN3's state diagrams



CC_process mode3

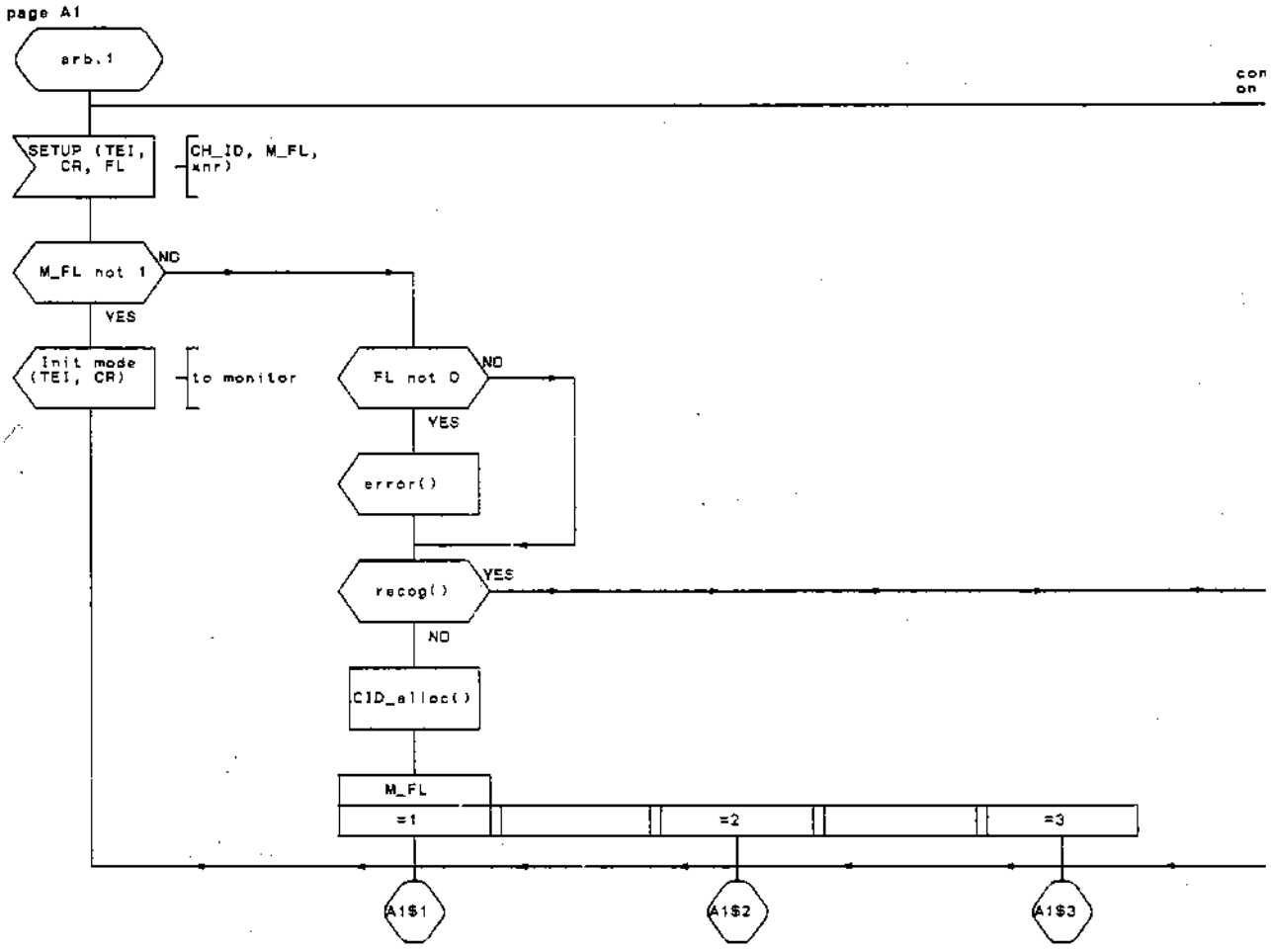
Figure 9. CC3's state diagrams



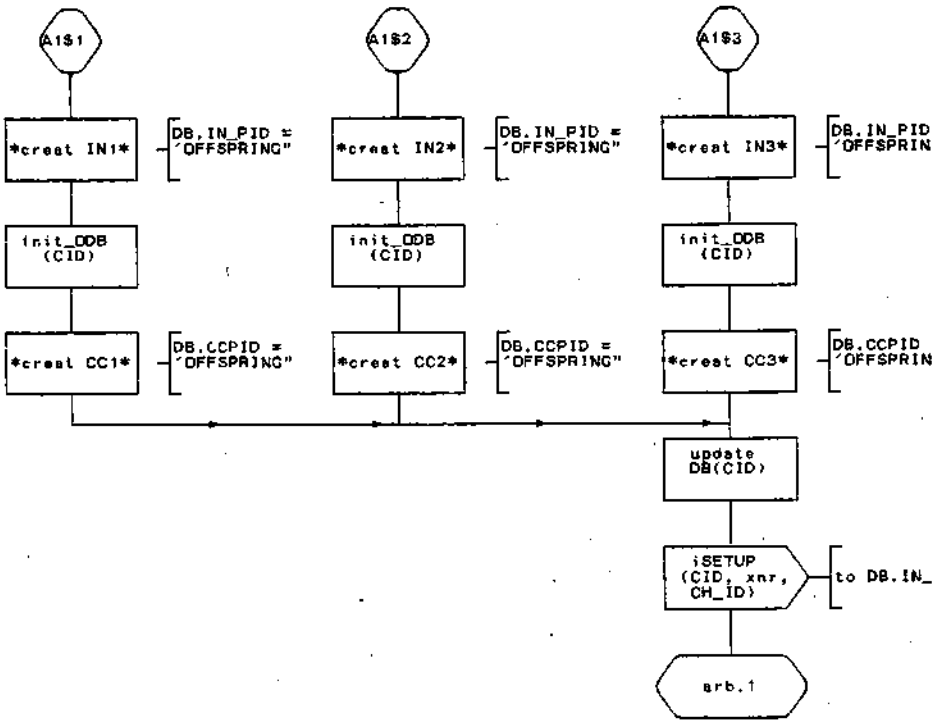
OUT_process mode3

Figure 10. OUT's state diagrams

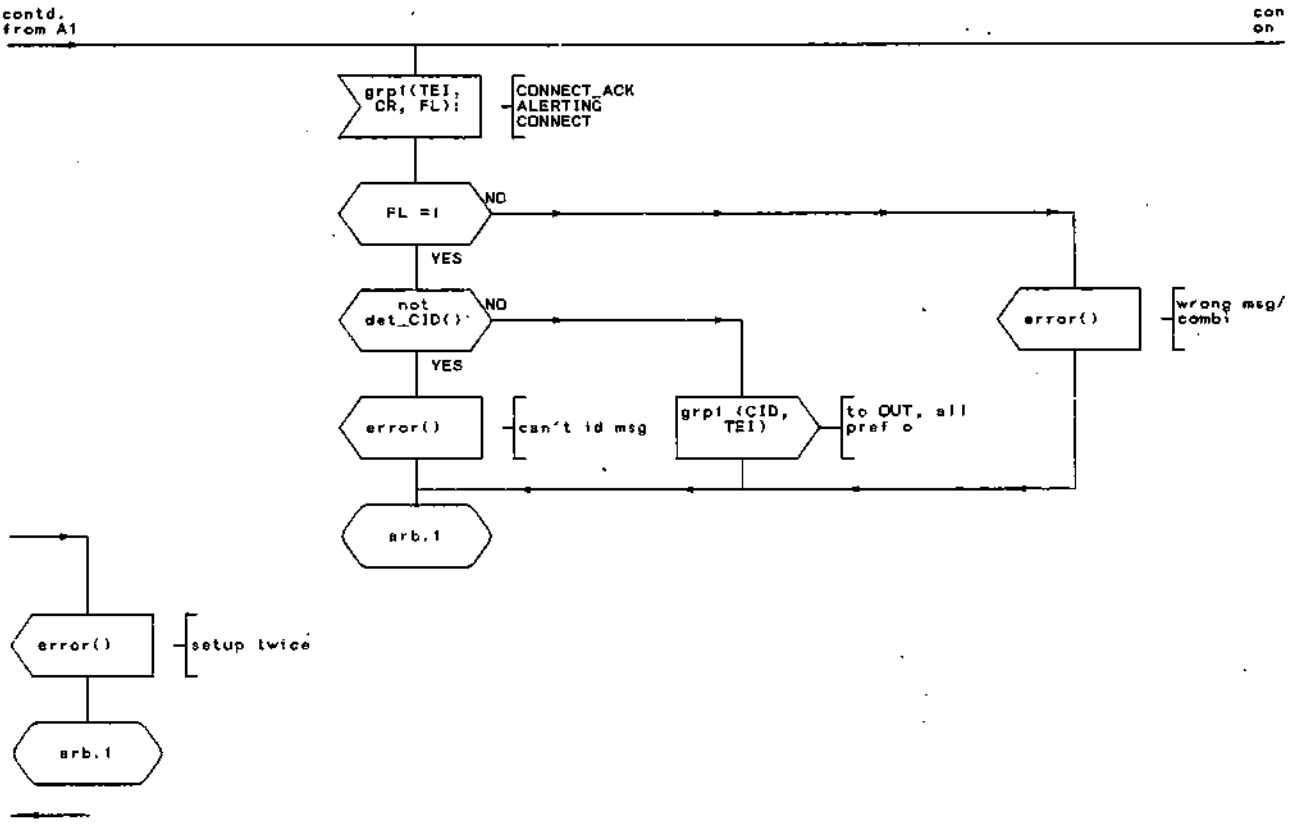
_APPENDIX_C:_THE_SDL_DIAGRAMS_OF_THE_NT12_L3's_PROCESSES_


page A1
arb.1
SETUP (TEI, CR, FL
CH_ID, M_FL, xnr)
M_FL not 1
NO
YES
Init mode (TEI, CR)
to monitor
FL not 0
NO
YES
error()
recog()
YES
NO
CID_alloc()
M_FL
=1
=2
=3
A1$1
A1$2
A1$3

page A1 contd.

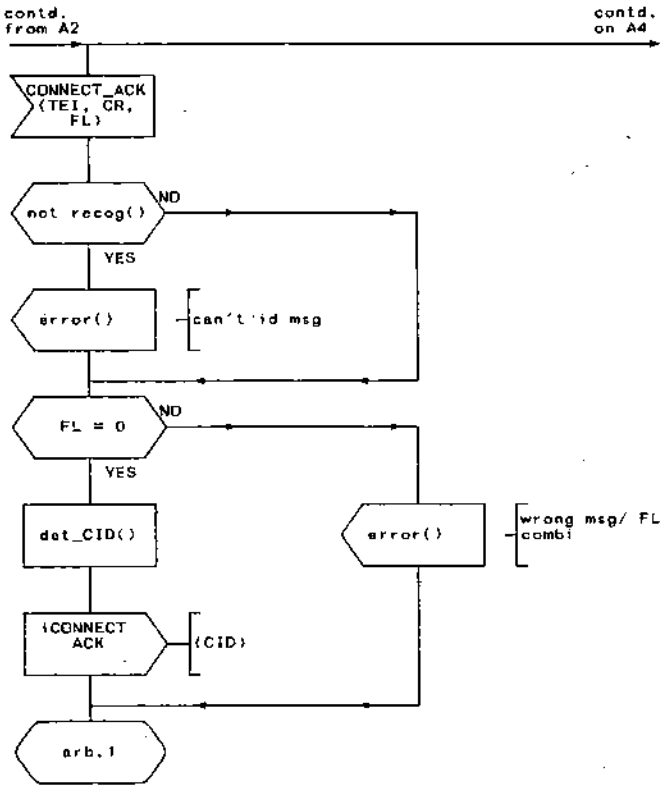


page A2

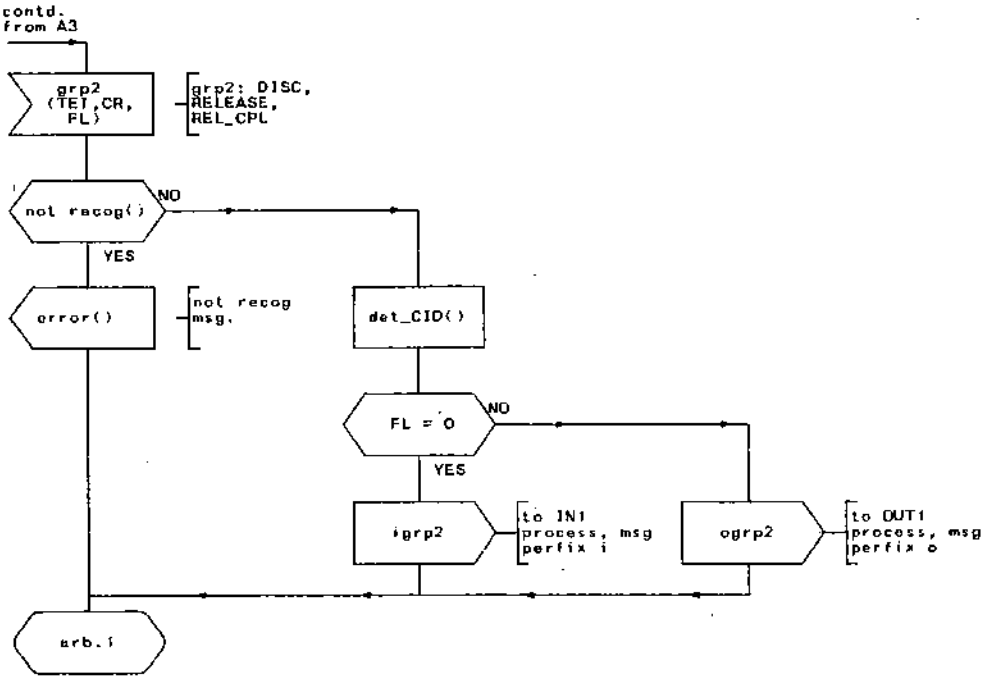

contd.
from A1
con
on
grp1(TEI,
CR, FL);
CONNECT_ACK
ALERTING
CONNECT
FL =1
NO
YES
not
det_CID()
NO
YES
error()
wrong msg/
combi
error()
can't id msg
grp1 (CID,
TEI)
to OUT, all
pref o
arb.1

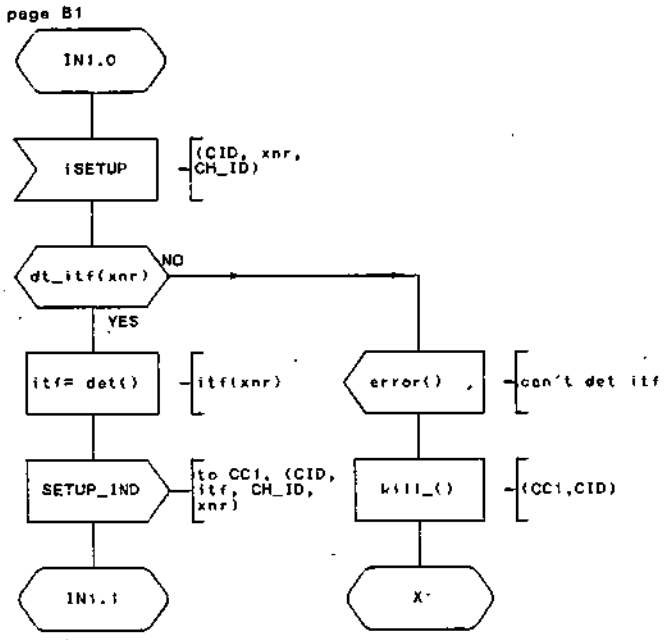
error()
setup twice
arb.1

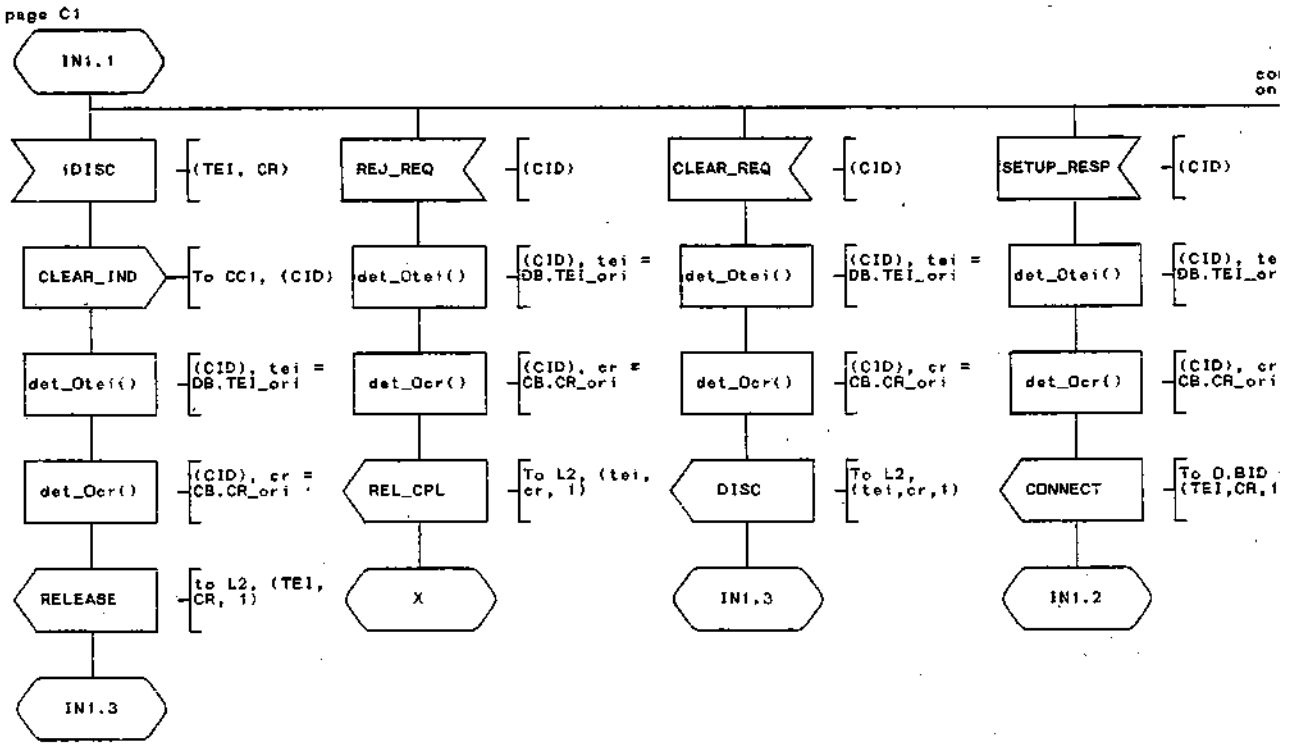
page A3


contd.
from A2
contd.
on A4
CONNECT_ACK
(TEI, CR,
FL)
not recog()
ND
YES
error()
can't id msg
FL = 0
ND
YES
det_CID()
error()
wrong msg/ FL
combi
(CONNECT
ACK
(CID)
arb.1

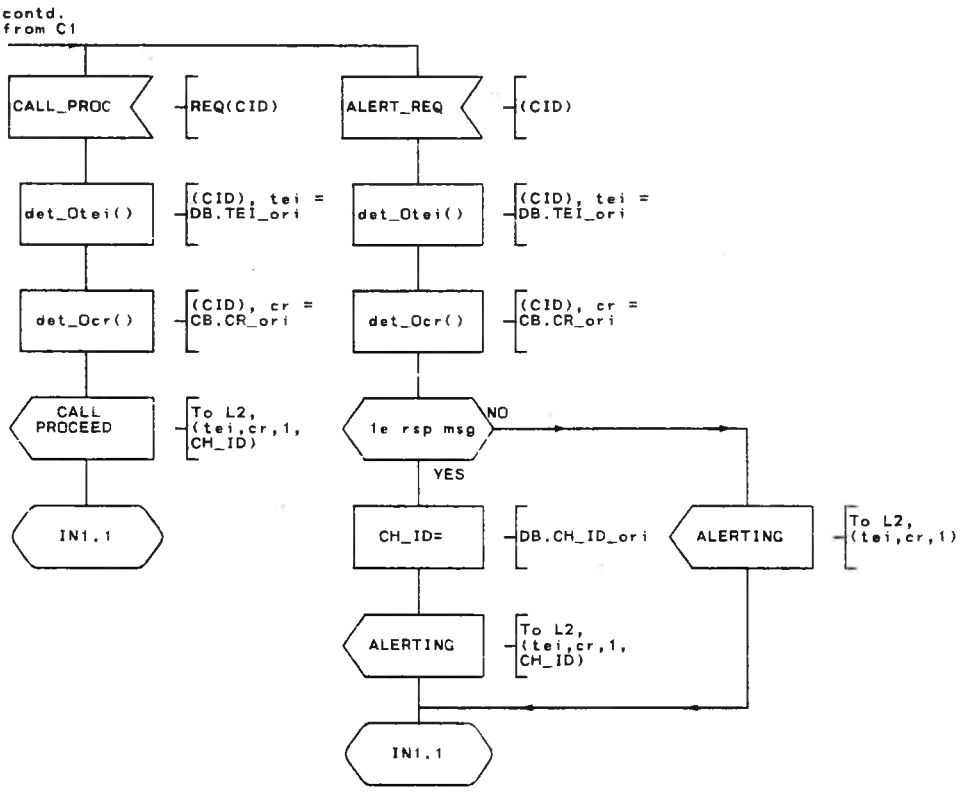
page A4

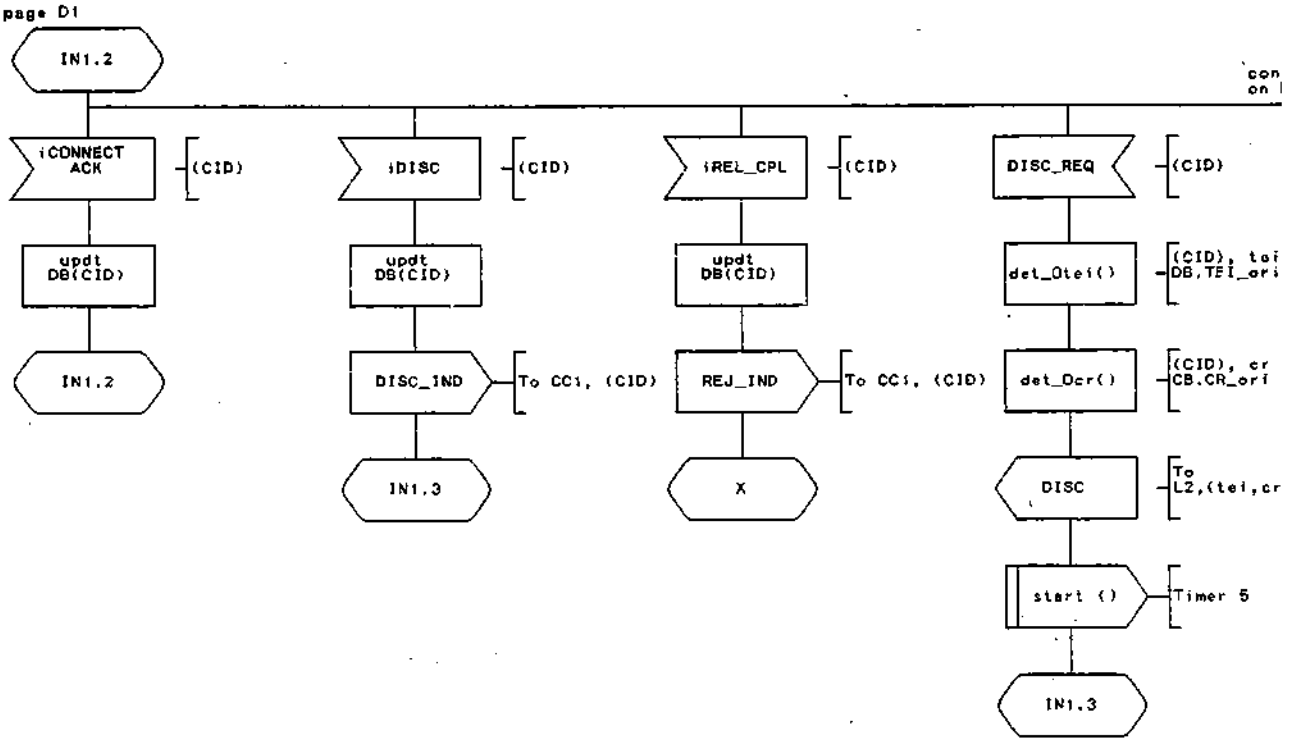

contd.
from A3
grp2
(TEI,CR,
FL)
grp2: DISC,
RELEASE,
REL_CPL
not recog()
NO
YES
error()
not recog
msg.
det_CID()
FL = 0
NO
YES
igrp2
to IN1
process, msg
perfix i
ogrp2
to OUT1
process, msg
perfix o
arb.1


page B1
IN1.0
iSETUP
(CID, xnr, CH_ID)
dt_itf(xnr)
NO
YES
itf= det()
itf(xnr)
error()
can't det itf
SETUP_IND
to CC1, (CID, itf, CH_ID, xnr)
kill_()
(CC1,CID)
IN1.1
X

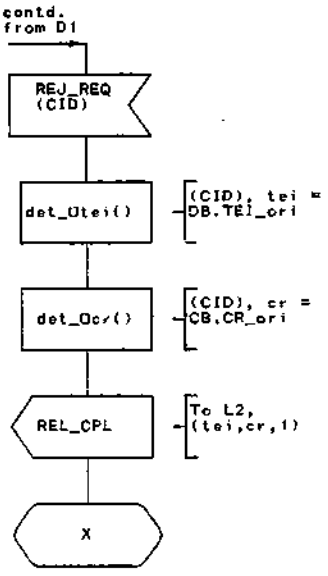

page C1
IN1.1
(DISC
(TEI, CR)
CLEAR_IND
To CC1, (CID)
det_Otei()
(CID), tei = DB.TEI_ori
det_Ocr()
(CID), cr = CB.CR_ori
RELEASE
to L2, (TEI, CR, 1)
IN1.3
REJ_REQ
(CID)
det_Otei()
(CID), tei = DB.TEI_ori
det_Ocr()
(CID), cr = CB.CR_ori
REL_CPL
To L2, (tei, cr, 1)
X
CLEAR_REQ
(CID)
det_Otei()
(CID), tei = DB.TEI_ori
det_Ocr()
(CID), cr = CB.CR_ori
DISC
To L2, (tei,cr,1)
IN1.3
SETUP_RESP
(CID)
det_Otei()
(CID), te DB.TEI_or
det_Ocr()
(CID), cr CB.CR_ori
CONNECT
To D.BID (TEI,CR,1
IN1.2

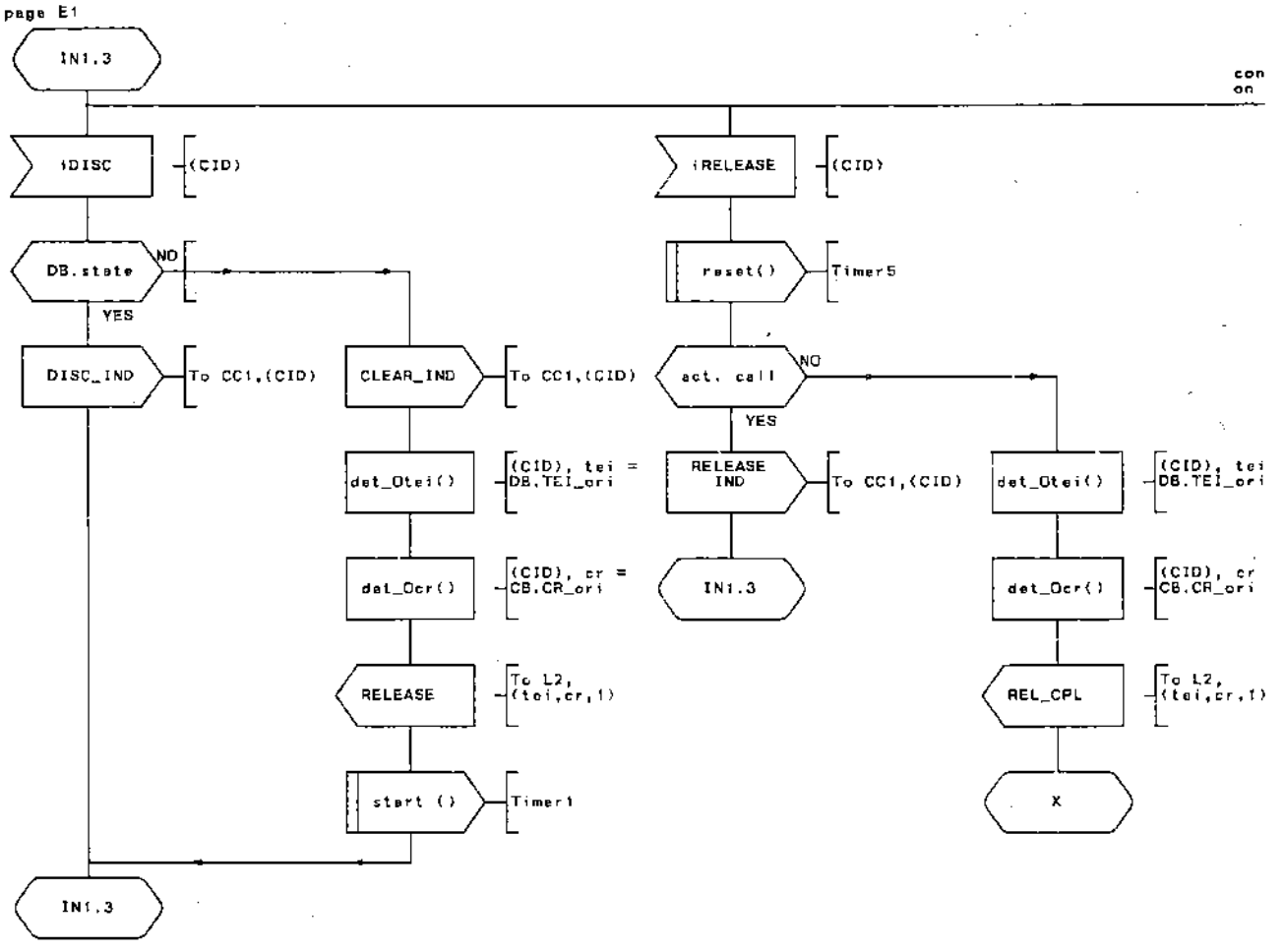
page C2


contd.
from C1
CALL_PROC
REQ(CID)
ALERT_REQ
(CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
det_Ocr()
(CID), cr =
CB.CR_ori
CALL
PROCEED
To L2,
(tei,cr,1,
CH_ID)
1e rsp msg
NO
YES
IN1.1
CH_ID=
DB.CH_ID_ori
ALERTING
To L2,
(tei,cr,1)
ALERTING
To L2,
(tei,cr,1,
CH_ID)
IN1.1

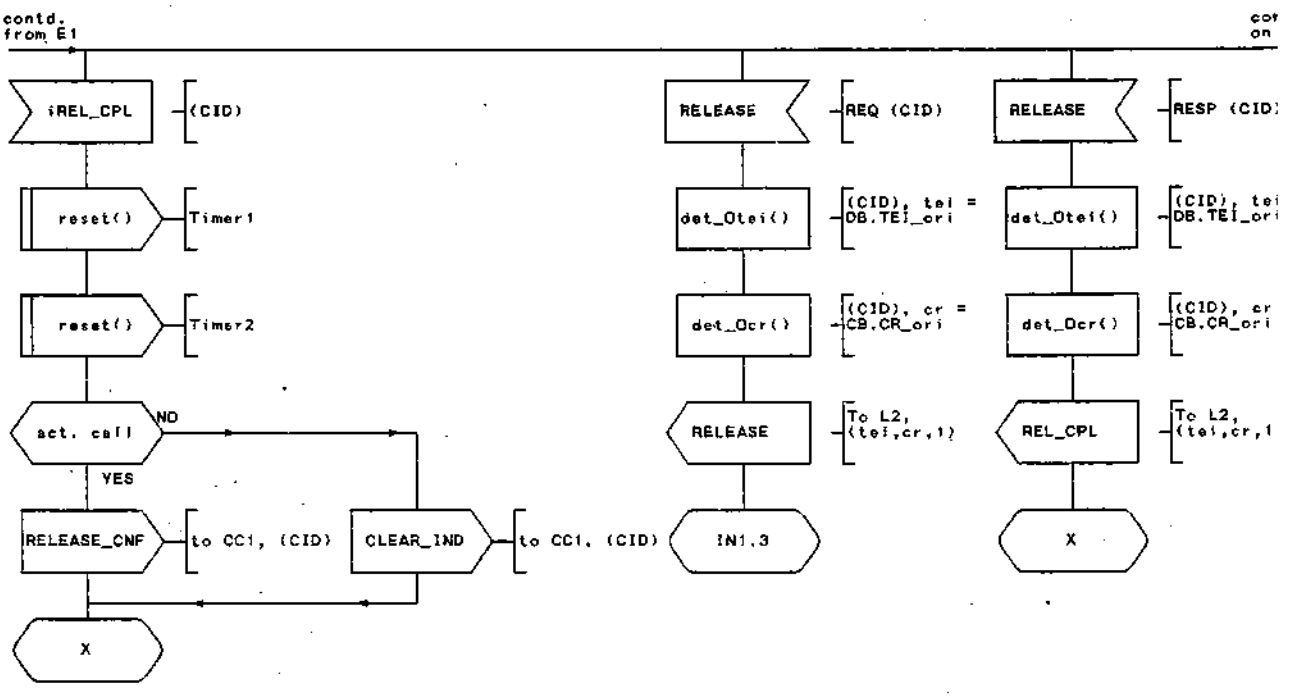

page D1
IN1.2
con
on
iCONNECT ACK
(CID)
iDISC
(CID)
iREL_CPL
(CID)
DISC_REQ
(CID)
updt DB(CID)
updt DB(CID)
updt DB(CID)
det_Otei()
(CID), tei
DB,TEI_ori
IN1.2
DISC_IND
To CC1, (CID)
REJ_IND
To CC1, (CID)
det_Ocr()
(CID), cr
CB.CR_ori
IN1.3
X
DISC
To
L2,(tei,cr
start ()
Timer 5
IN1.3

page D2

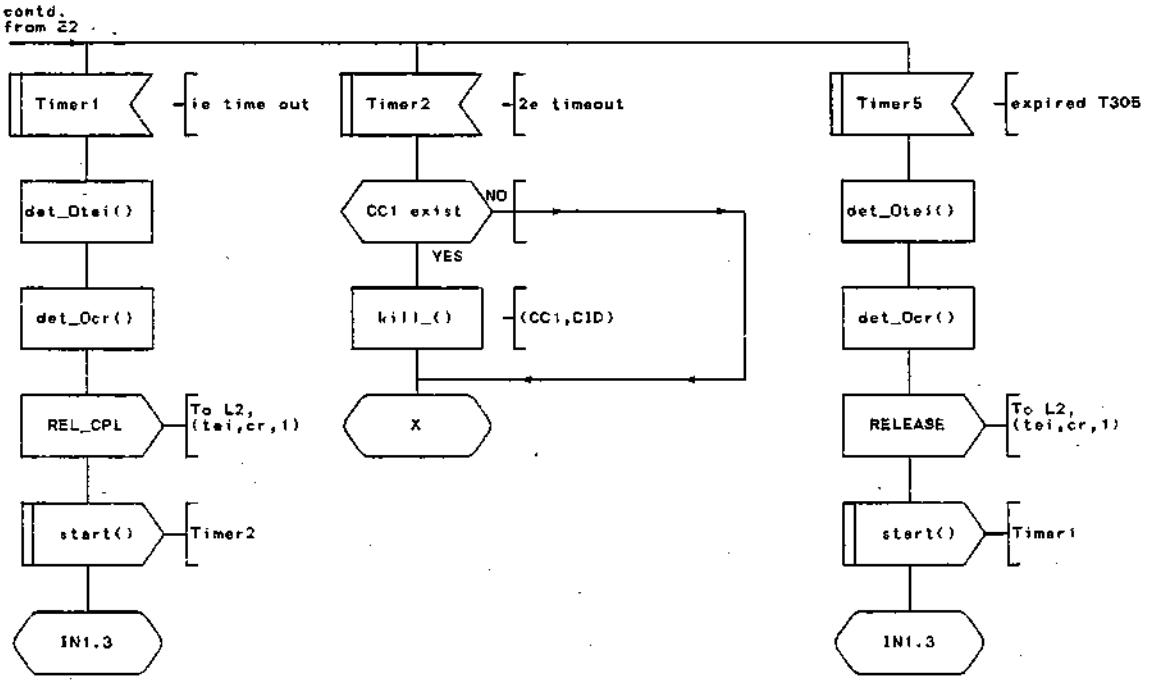

contd.
from D1
REJ_REQ
(CID)
det_Utei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
REL_CPL
To L2,
(tei,cr,1)
X

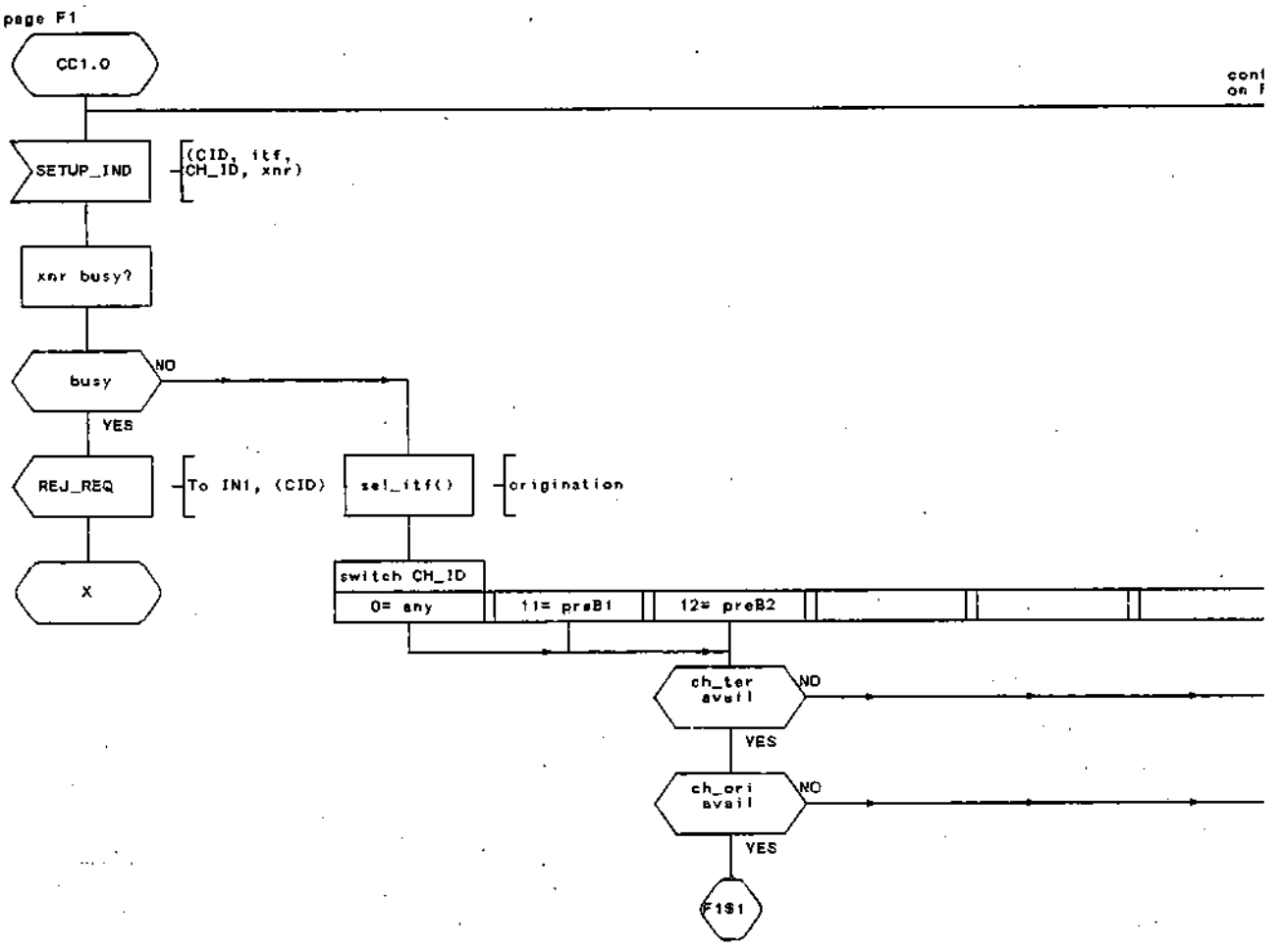

page E1
IN1,3
con
on
↓DISC
(CID)
↓RELEASE
(CID)
DB.state
NO
YES
reset()
Timer5
DISC_IND
To CC1,(CID)
CLEAR_IND
To CC1,(CID)
act. call
NO
YES
det_Otei()
(CID), tei =
DB.TEI_ori
RELEASE
IND
To CC1,(CID)
det_Otei()
(CID), tei
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
IN1,3
det_Ocr()
(CID), cr
CB.CR_ori
RELEASE
To L2,
(tei,cr,1)
REL_CPL
To L2,
(tei,cr,1)
start ()
Timer1
X
IN1,3

page E2

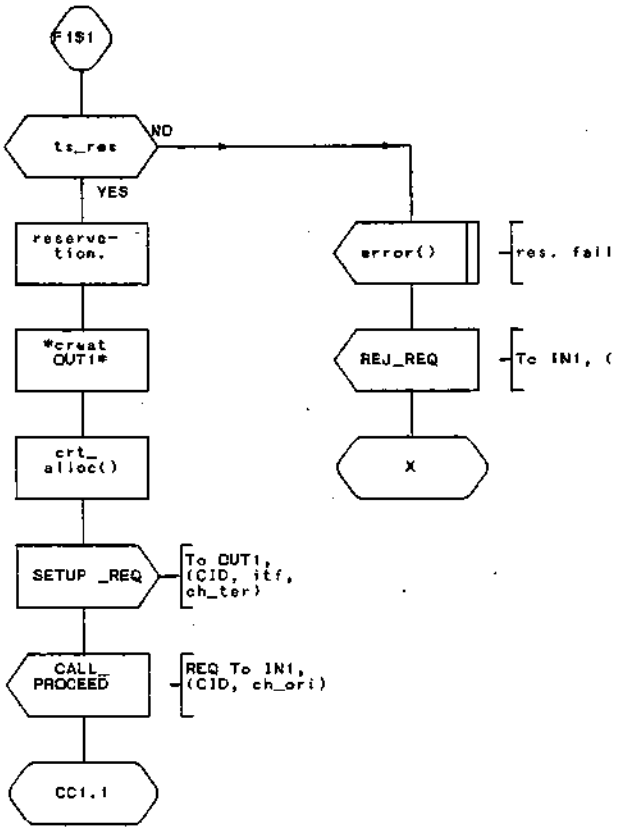

contd.
from E1
cor
on
iREL_CPL
(CID)
RELEASE
REQ (CID)
RELEASE
RESP (CID)
reset()
Timer1
det_Otei()
(CID), tei =
DB.TEI_ori
det_Otei()
(CID), tei
DB.TEI_ori
reset()
Timer2
det_Ocr()
(CID), cr =
CB.CR_ori
det_Ocr()
(CID), cr
CB.CR_ori
act. call
NO
YES
RELEASE
To L2,
(tei,cr,1)
REL_CPL
To L2,
(tei,cr,1
RELEASE_CNF
to CC1, (CID)
CLEAR_IND
to CC1, (CID)
IN1.3
X
X

page E3


contd.
from E2
Timer1
ie time out
Timer2
2e timeout
Timer5
expired T305
det_Otei()
CC1 exist
NO
YES
det_Otei()
det_Ocr()
kill_()
(CC1,CID)
det_Ocr()
REL_CPL
To L2,
(tei,cr,1)
X
RELEASE
To L2,
(tei,cr,1)
start()
Timer2
start()
Timer1
IN1.3
IN1.3


page F1
CC1.0
cont
on F
SETUP_IND
(CID, itf,
CH_ID, xnr)
xnr busy?
busy
NO
YES
REJ_REQ
To IN1, (CID)
sel_itf()
origination
X
switch CH_ID
0= any
11= preB1
12= preB2
ch_ter
avail
NO
YES
ch_ori
avail
NO
YES
F1$1

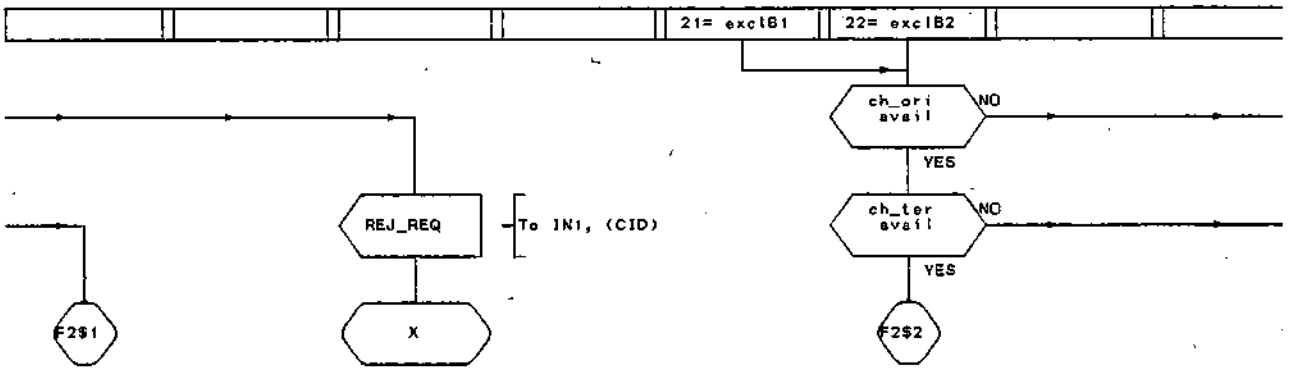
page F1 contd.



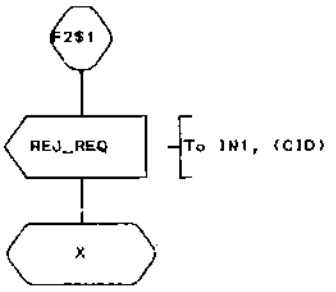
page F2

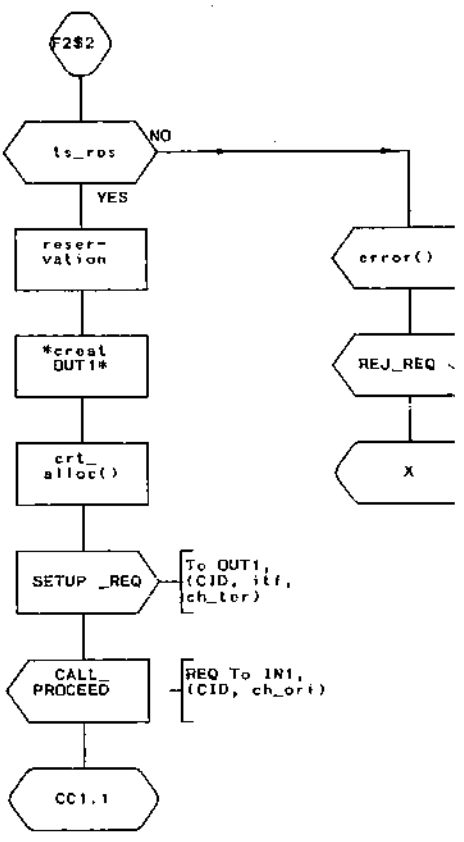
contd.
from F1

con
on

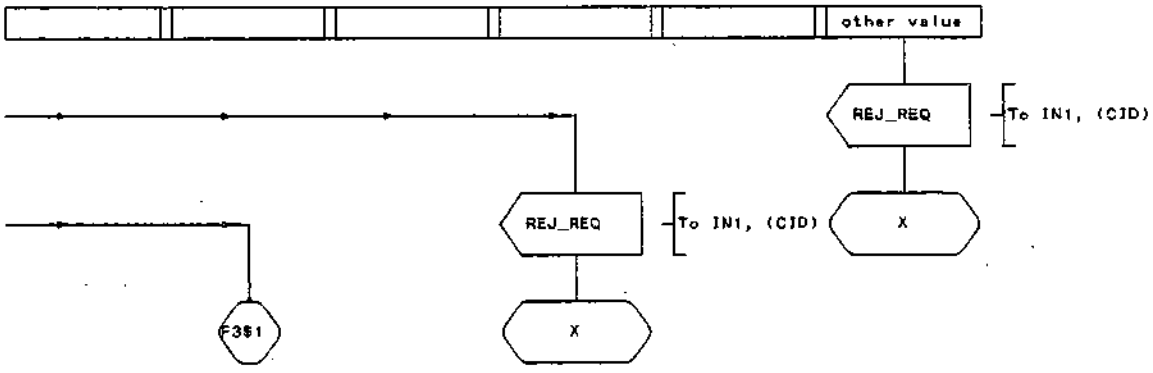


page F2 contd.

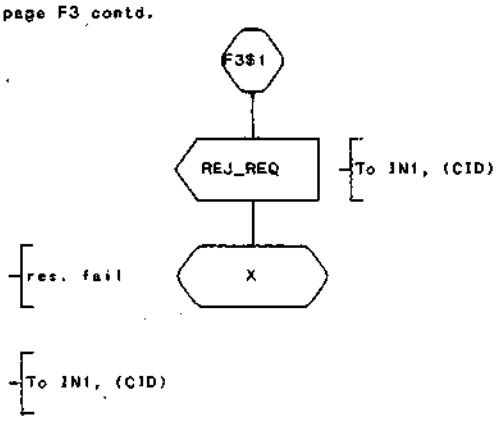


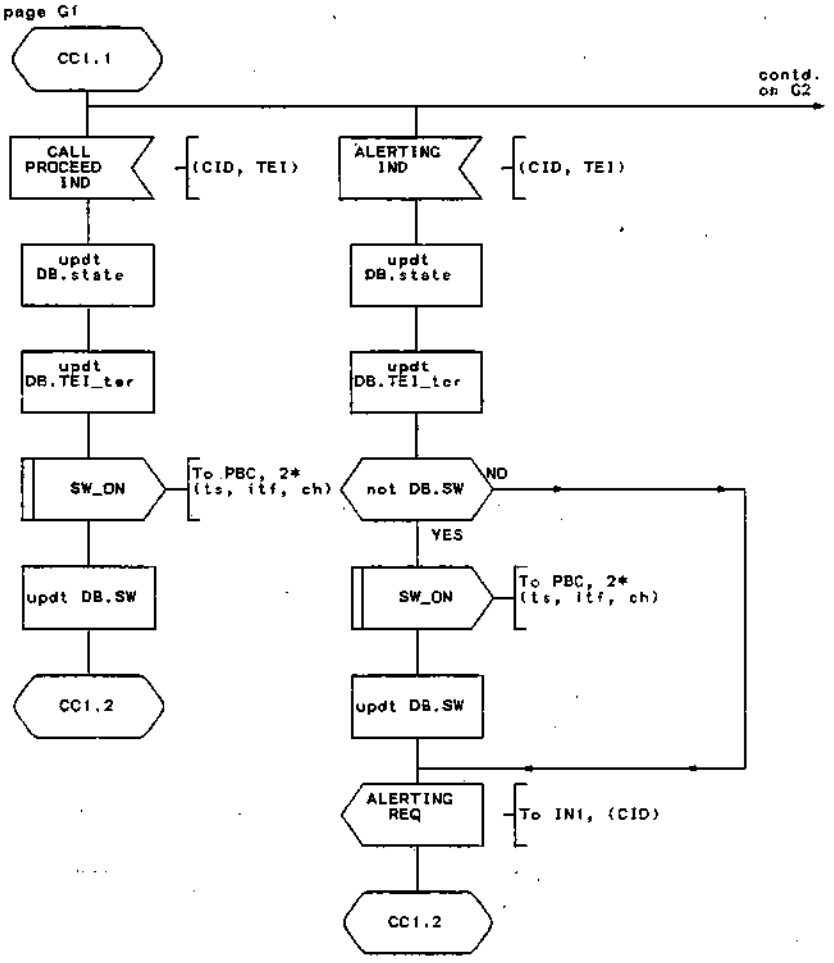


page F3

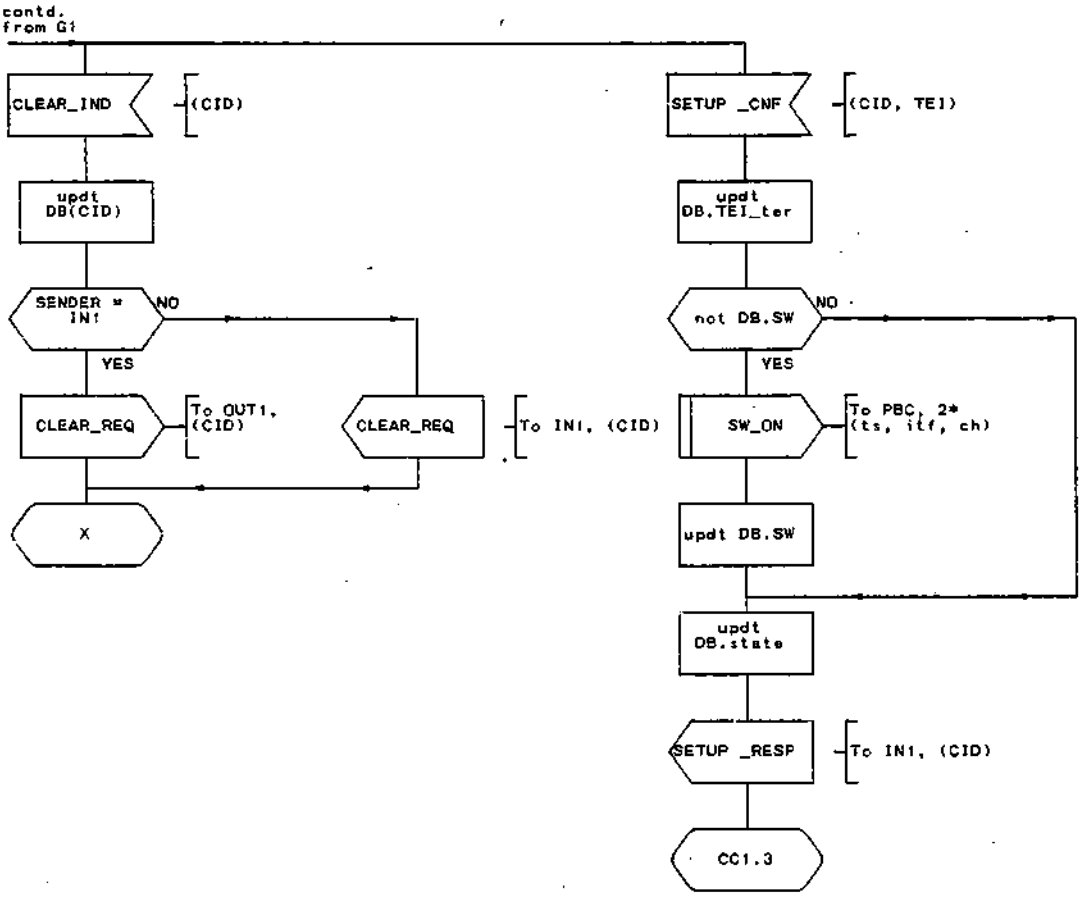


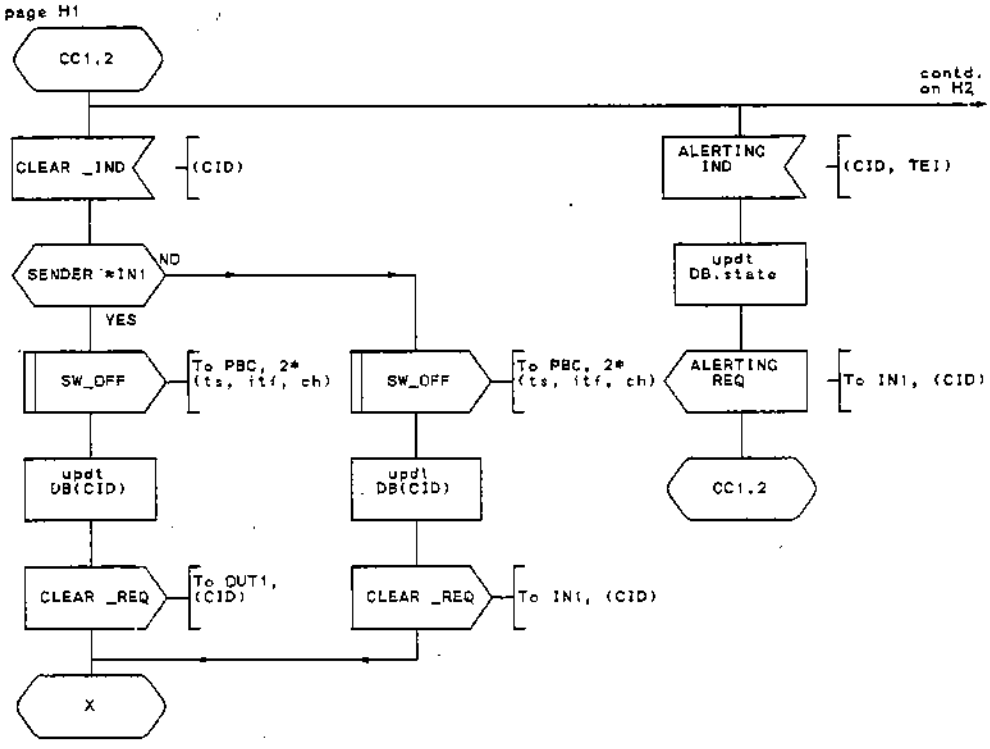
page F3 contd.



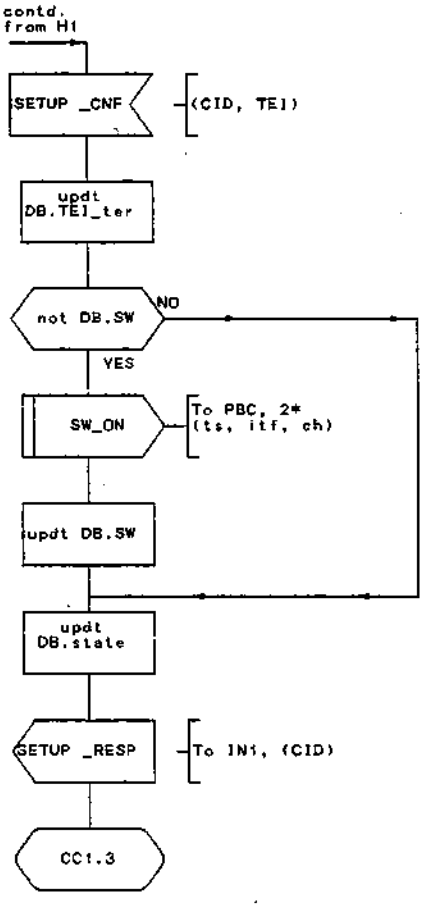

page G1
CC1.1
contd.
on G2
CALL
PROCEED
IND
(CID, TEI)
ALERTING
IND
(CID, TEI)
updt
DB.state
updt
DB.state
updt
DB.TEI_ter
updt
DB.TEI_ter
SW_ON
To PBC, 2*
(ts, itf, ch)
not DB.SW
NO
YES
updt DB.SW
SW_ON
To PBC, 2*
(ts, itf, ch)
CC1.2
updt DB.SW
ALERTING
REQ
To IN1, (CID)
CC1.2

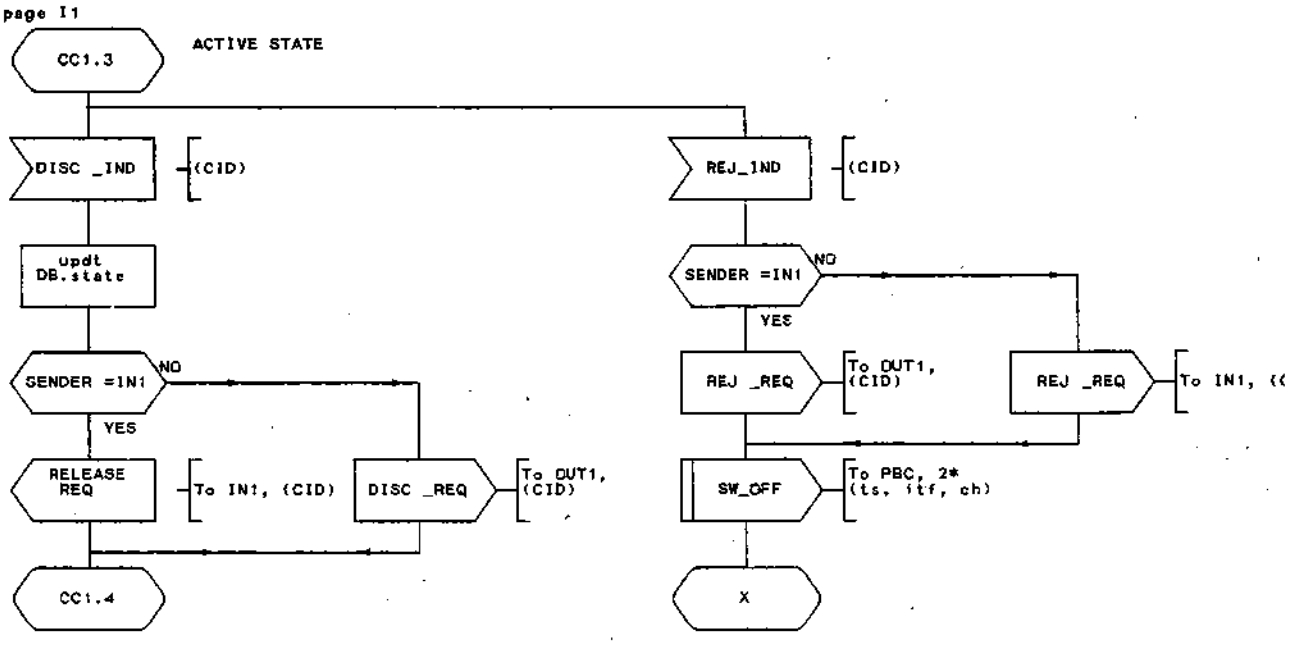
page G2

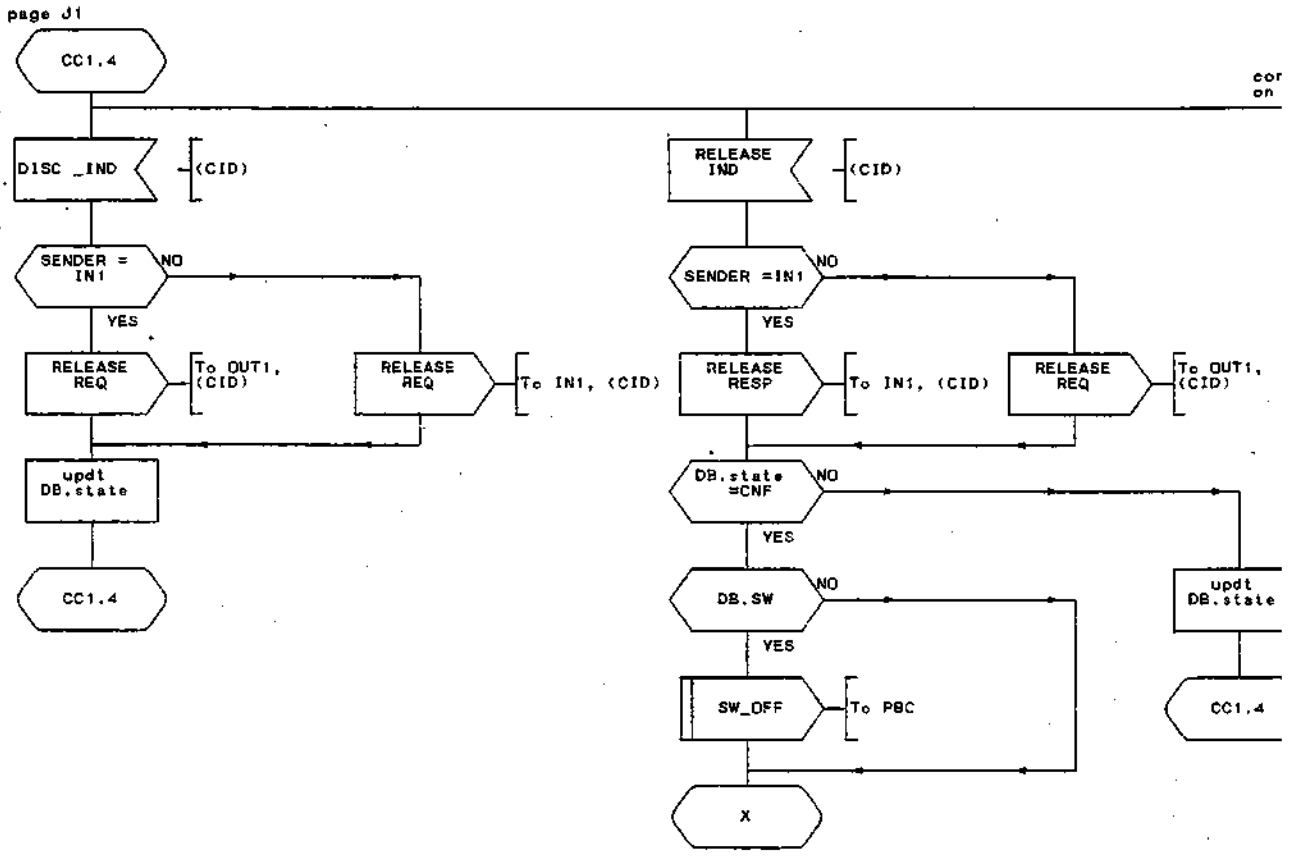

contd.
from G1
CLEAR_IND
(CID)
updt
DB(CID)
SENDER =
IN1
NO
YES
CLEAR_REQ
To OUT1,
(CID)
CLEAR_REQ
To IN1, (CID)
X
SETUP _CNF
(CID, TEI)
updt
DB.TEI_ter
not DB.SW
NO
YES
SW_ON
To PBC, 2*
(ts, itf, ch)
updt DB.SW
updt
DB.state
SETUP _RESP
To IN1, (CID)
CC1.3


page H1
CC1.2
contd.
on H2
CLEAR _IND
(CID)
ALERTING
IND
(CID, TEI)
SENDER *IN1
ND
YES
updt
DB.state
SW_OFF
To PBC, 2*
(ts, itf, ch)
SW_OFF
To PBC, 2*
(ts, itf, ch)
ALERTING
REQ
To IN1, (CID)
updt
DB(CID)
updt
DB(CID)
CC1.2
CLEAR _REQ
To OUT1,
(CID)
CLEAR _REQ
To IN1, (CID)
X

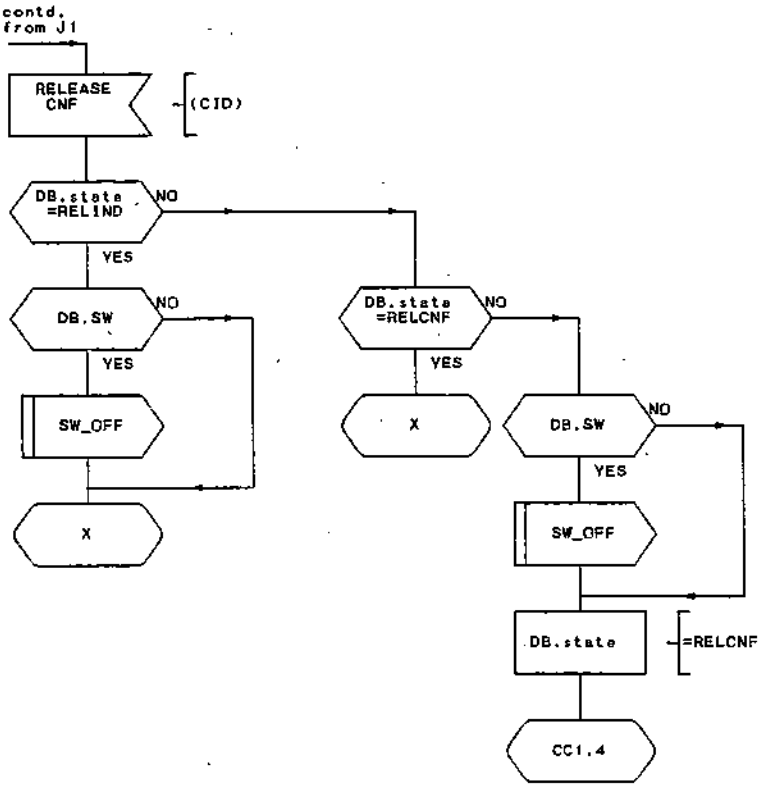
page H2

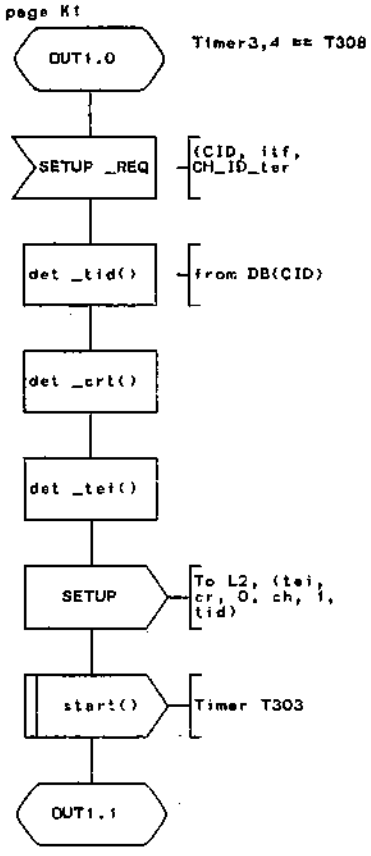

contd.
from H1
SETUP _CNF
(CID, TEI)
updt
DB.TEI_ter
not DB.SW
NO
YES
SW_ON
To PBC, 2*
(ts, itf, ch)
updt DB.SW
updt
DB.state
SETUP _RESP
To IN1, (CID)
CC1.3

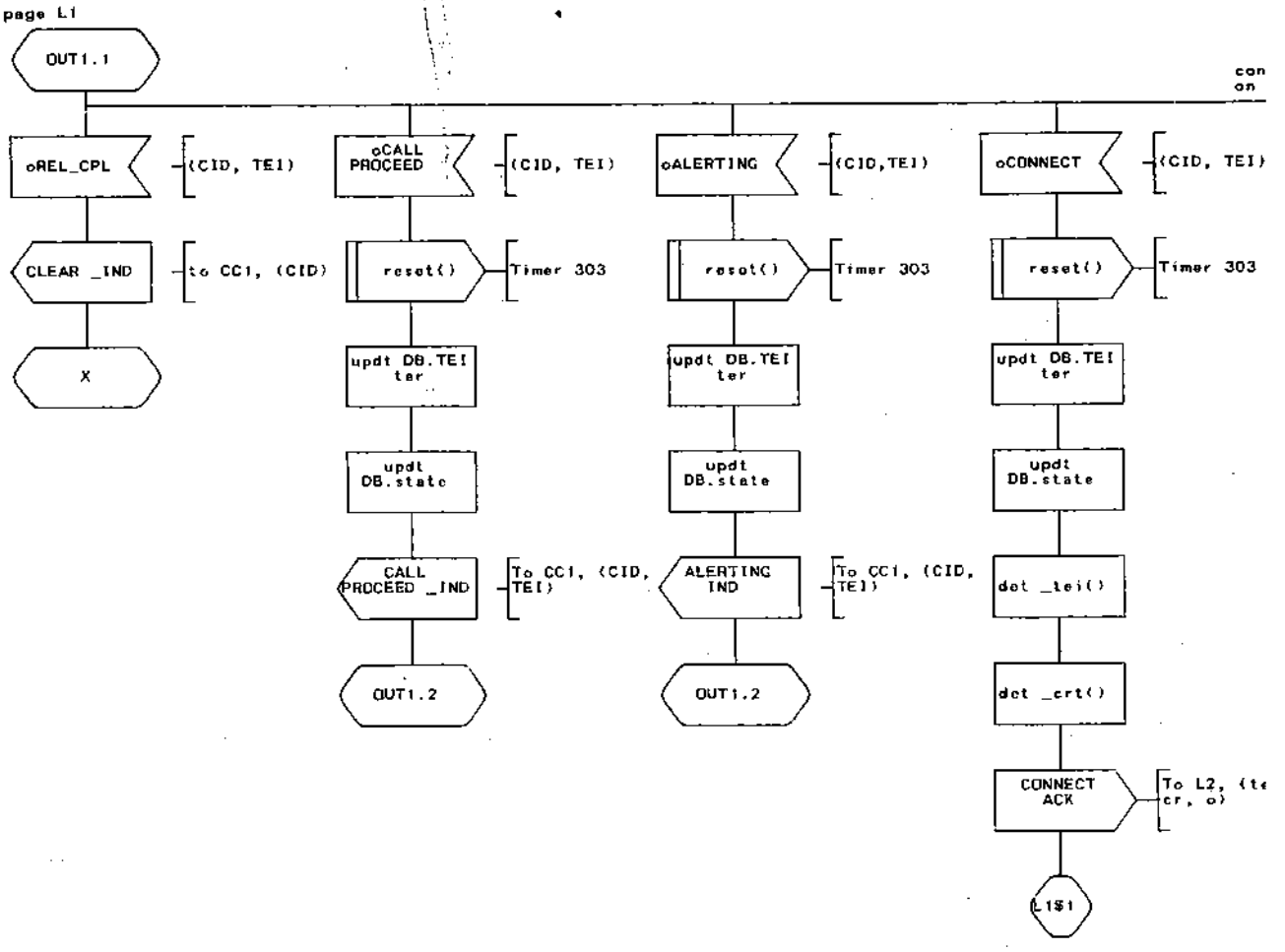

page I1
CC1.3
ACTIVE STATE
DISC _IND
(CID)
updt
DB.state
SENDER =IN1
NO
YES
RELEASE
REQ
To IN1, (CID)
DISC _REQ
To OUT1,
(CID)
CC1.4
REJ_IND
(CID)
SENDER =IN1
NO
YES
REJ _REQ
To OUT1,
(CID)
REJ _REQ
To IN1, ((
SW_OFF
To PBC, 2*
(ts, itf, ch)
X


page J1
CC1.4
cor
on
DISC _IND
(CID)
SENDER = IN1
NO
YES
RELEASE REQ
To OUT1, (CID)
RELEASE REQ
To IN1, (CID)
updt DB.state
CC1.4
RELEASE IND
(CID)
SENDER =IN1
NO
YES
RELEASE RESP
To IN1, (CID)
RELEASE REQ
To OUT1, (CID)
DB.state =CNF
NO
YES
DB.SW
NO
YES
SW_OFF
To PBC
X
updt DB.state
CC1.4

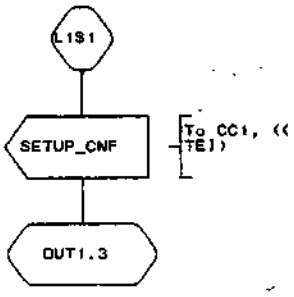
page J2


contd.
from J1
RELEASE CNF
(CID)
DB.state =REL1ND
NO
YES
DB.SW
NO
YES
SW_OFF
X
DB.state =RELCNF
NO
YES
X
DB.SW
NO
YES
SW_OFF
DB.state
=RELCNF
CC1.4

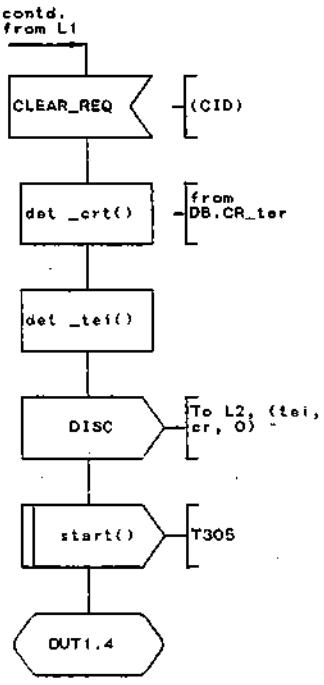

page K1
OUT1.0
Timer3,4 == T308
SETUP _REQ
(CID, itf,
CH_ID_ter
det _tid()
from DB(CID)
det _crt()
det _tei()
SETUP
To L2, (tei,
cr, 0, ch, 1,
tid)
start()
Timer T303
OUT1.1


page L1
OUT1.1
con
on
oREL_CPL
(CID, TEI)
oCALL PROCEED
(CID, TEI)
oALERTING
(CID,TEI)
oCONNECT
(CID, TEI)
CLEAR _IND
to CC1, (CID)
reset()
Timer 303
reset()
Timer 303
reset()
Timer 303
X
updt DB.TEI ter
updt DB.TEI ter
updt DB.TEI ter
updt DB.state
updt DB.state
updt DB.state
CALL PROCEED _IND
To CC1, (CID, TEI)
ALERTING IND
To CC1, (CID, TEI)
det _tei()
OUT1.2
OUT1.2
det _crt()
CONNECT ACK
To L2, (te
cr, o)
L1$1

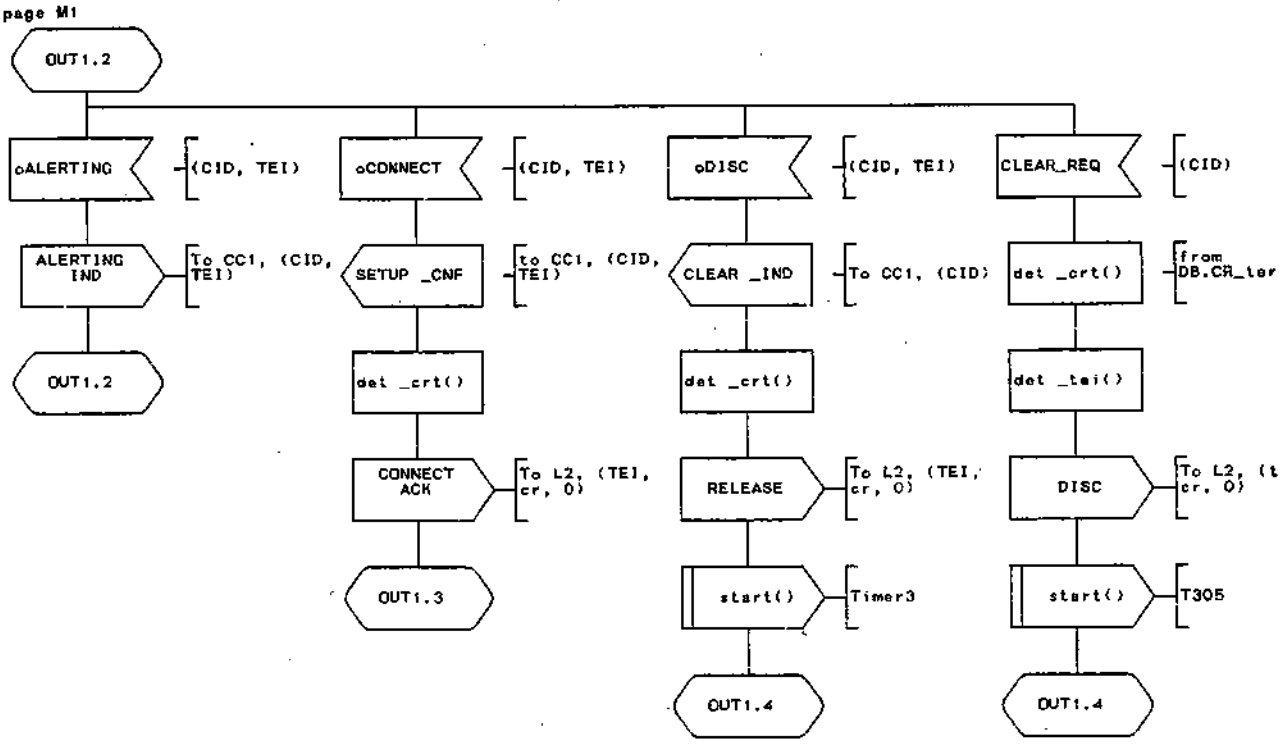
page L1 contd.

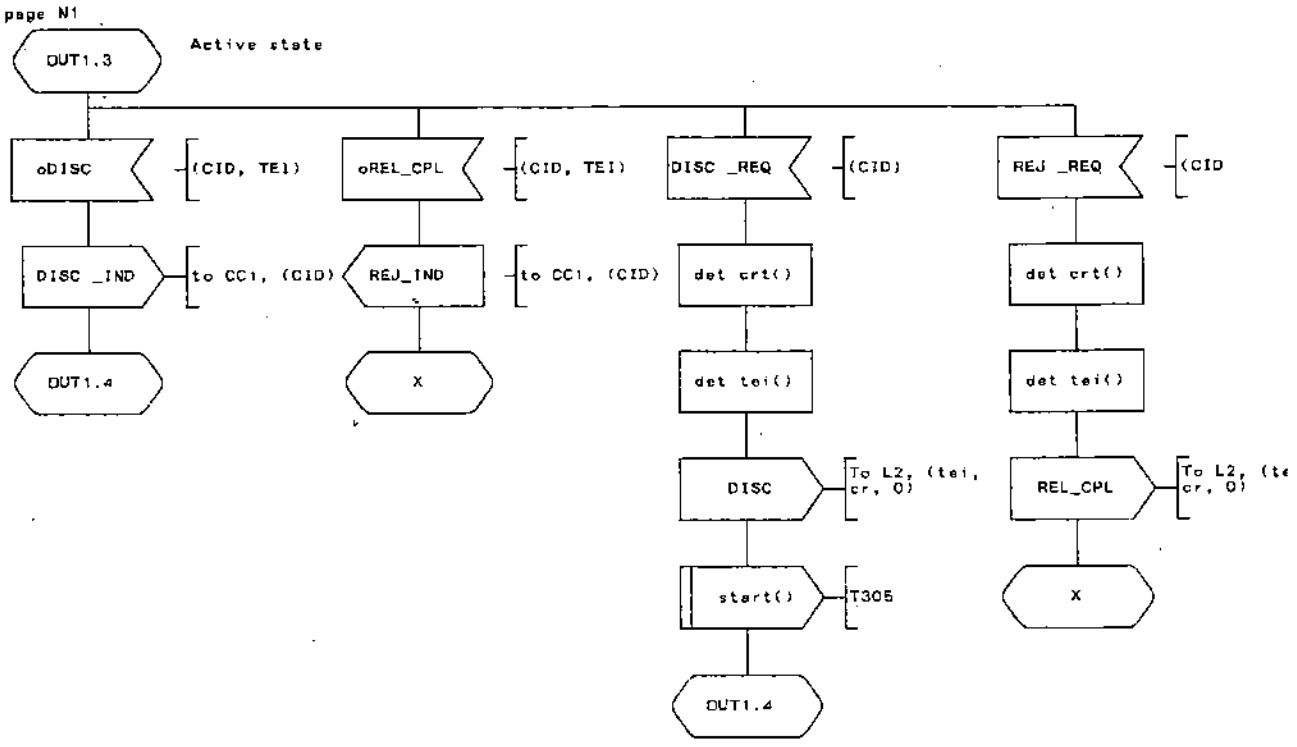


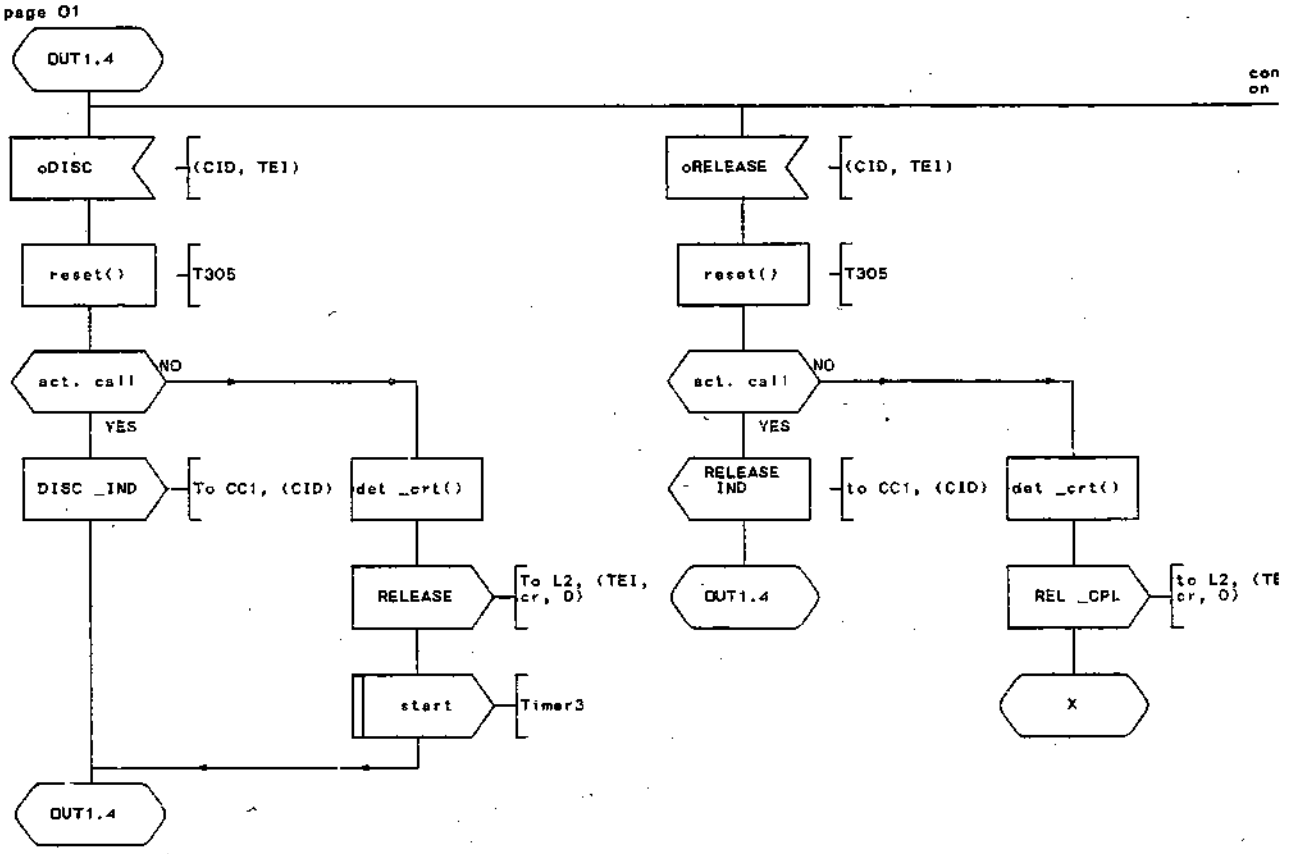
page L2


contd.
from L1
CLEAR_REQ
(CID)
det _crt()
from
DB.CR_ter
det _tei()
DISC
To L2, (tei,
cr, 0)
start()
T305
OUT1.4

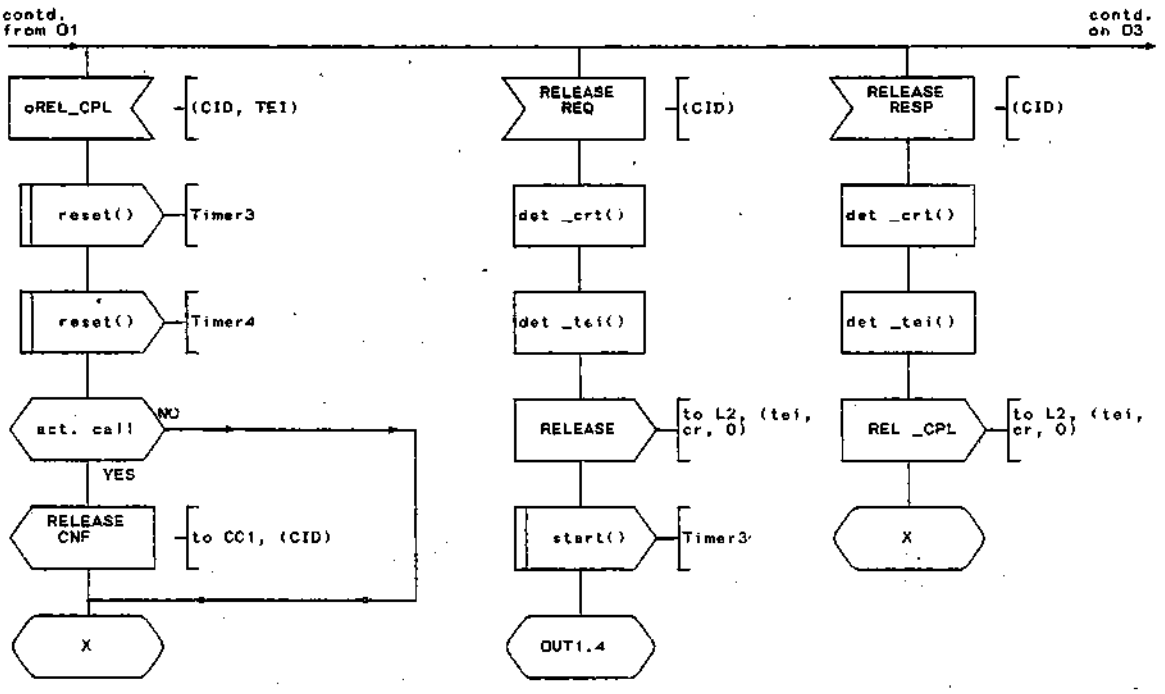
page M1


OUT1.2
oALERTING
(CID, TEI)
oCONNECT
(CID, TEI)
oDISC
(CID, TEI)
CLEAR_REQ
(CID)
ALERTING IND
To CC1, (CID, TEI)
SETUP _CNF
to CC1, (CID, TEI)
CLEAR _IND
To CC1, (CID)
det _crt()
from DB.CR_ter
OUT1.2
det _crt()
det _crt()
det _tei()
CONNECT ACK
To L2, (TEI, cr, 0)
RELEASE
To L2, (TEI, cr, 0)
DISC
To L2, (t cr, 0)
OUT1.3
start()
Timer3
start()
T305
OUT1.4
OUT1.4

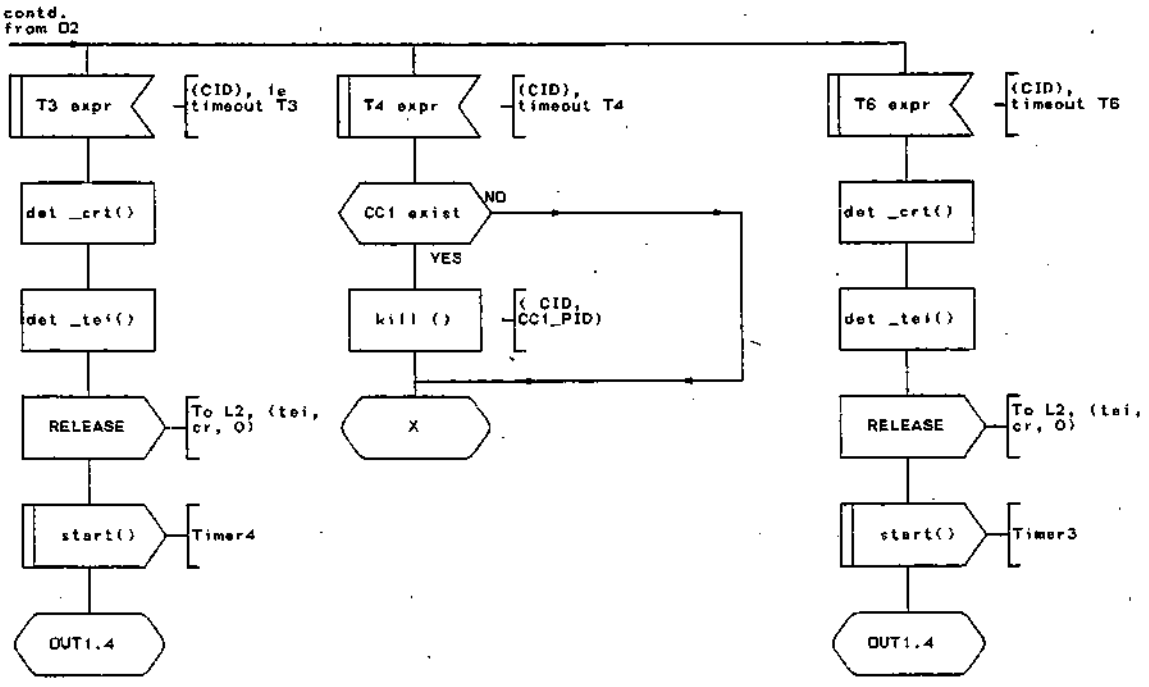

page N1
OUT1.3
Active state
oDISC
(CID, TEI)
oREL_CPL
(CID, TEI)
DISC _REQ
(CID)
REJ _REQ
(CID
DISC _IND
to CC1, (CID)
REJ_IND
to CC1, (CID)
det crt()
det crt()
OUT1.4
X
det tei()
det tei()
DISC
To L2, (tei, cr, 0)
REL_CPL
To L2, (te
cr, 0)
start()
T305
X
OUT1.4

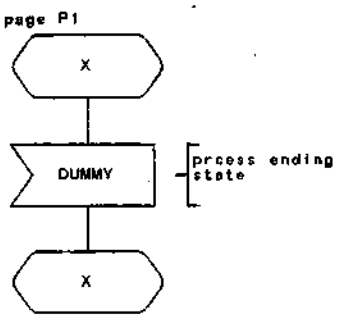

page 01
OUT1.4
con
on
oDISC
(CID, TEI)
reset()
T305
act. call
NO
YES
DISC _IND
To CC1, (CID)
det _crt()
RELEASE
To L2, (TEI,
cr, 0)
start
Timer3
OUT1.4
oRELEASE
(CID, TEI)
reset()
T305
act. call
NO
YES
RELEASE
IND
to CC1, (CID)
det _crt()
OUT1.4
REL _CPL
to L2, (TE
cr, 0)
X

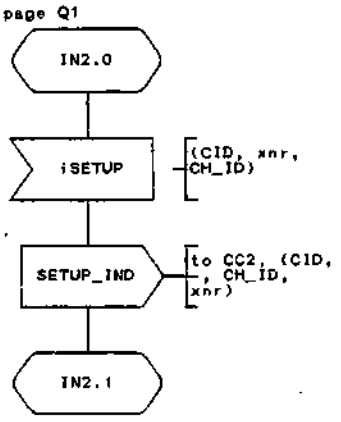
page 02

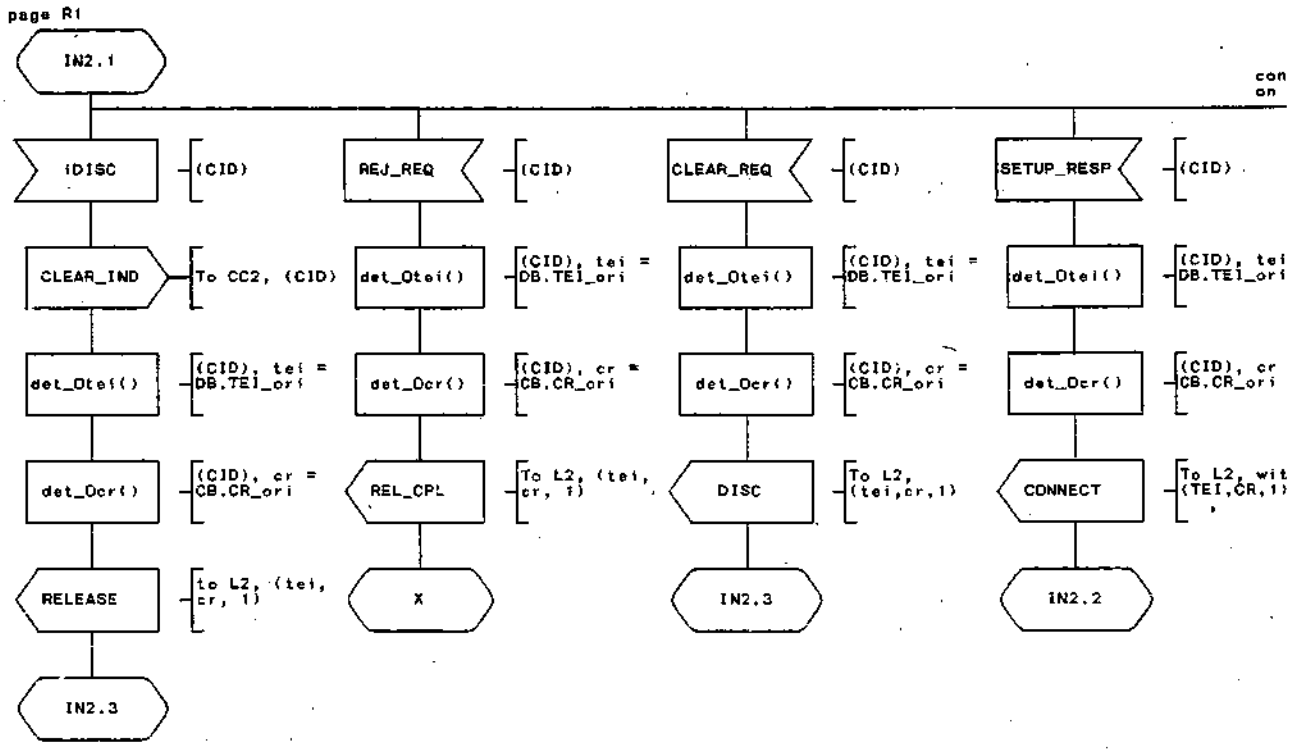

contd.
from 01
contd.
on 03
oREL_CPL
(CID, TEI)
reset()
Timer3
reset()
Timer4
act. call
NO
YES
RELEASE
CNF
to CC1, (CID)
X
RELEASE
REQ
(CID)
det _crt()
det _tei()
RELEASE
to L2, (tei,
cr, 0)
start()
Timer3
OUT1.4
RELEASE
RESP
(CID)
det _crt()
det _tei()
REL _CPL
to L2, (tei,
cr, 0)
X

page 03

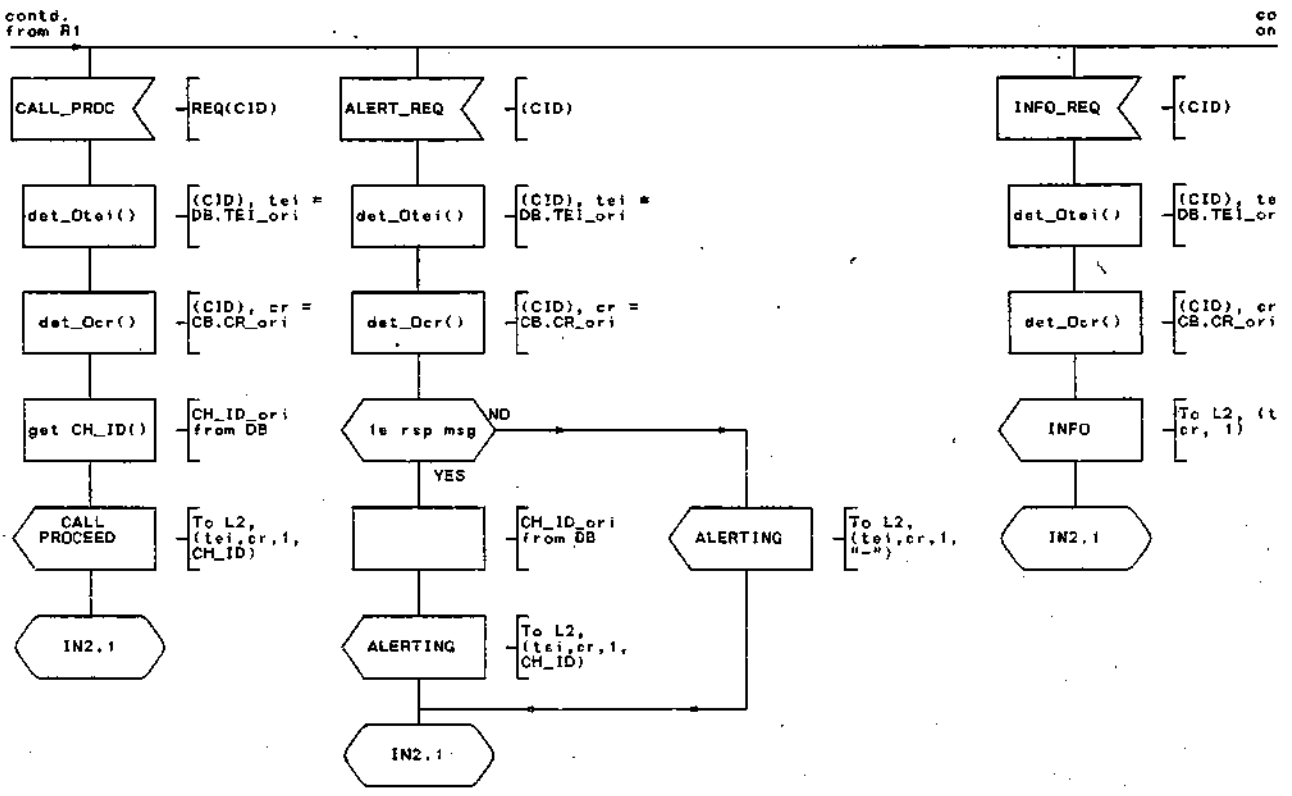

contd.
from 02
T3 expr
(CID), ie
timeout T3
T4 expr
(CID),
timeout T4
T6 expr
(CID),
timeout T6
det _crt()
CC1 exist
NO
YES
det _crt()
det _tei()
kill ()
(CID,
CC1_PID)
det _tei()
RELEASE
To L2, (tei,
cr, 0)
X
RELEASE
To L2, (tei,
cr, 0)
start()
Timer4
start()
Timer3
OUT1.4
OUT1.4


page P1
X
DUMMY
process ending state
X

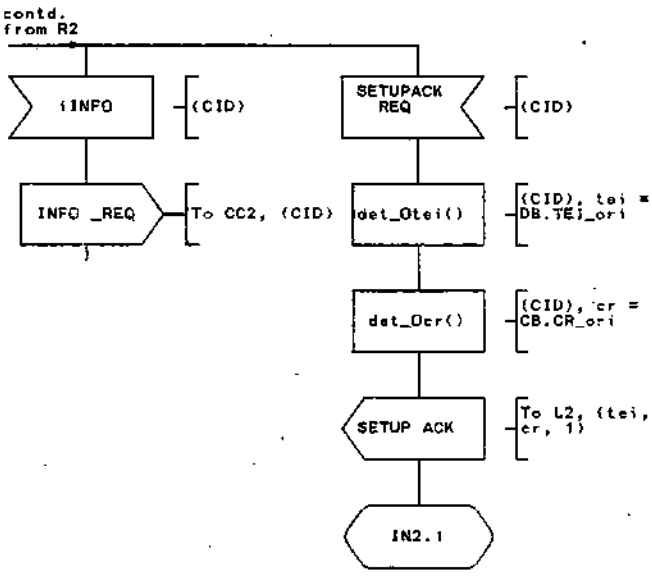

page Q1
IN2.0
iSETUP
(CID, xnr, CH_ID)
SETUP_IND
to CC2, (CID, , CH_ID, xnr)
IN2.1

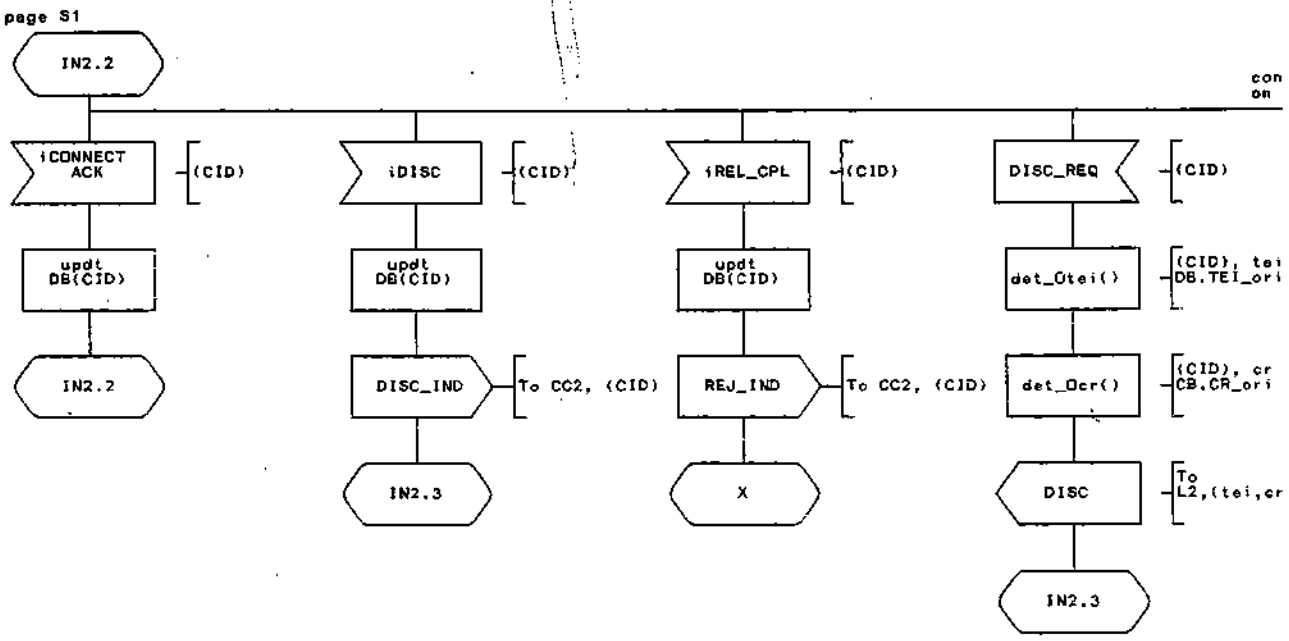

page R1
IN2.1
con
on
!DISC
(CID)
CLEAR_IND
To CC2, (CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
RELEASE
to L2, (tei,
cr, 1)
IN2.3
REJ_REQ
(CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
REL_CPL
To L2, (tei,
cr, 1)
X
CLEAR_REQ
(CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
DISC
To L2,
(tei,cr,1)
IN2.3
SETUP_RESP
(CID)
det_Otei()
(CID), tei
DB.TEI_ori
det_Ocr()
(CID), cr
CB.CR_ori
CONNECT
To L2, wit
(TEI,CR,1)
IN2.2

page R2

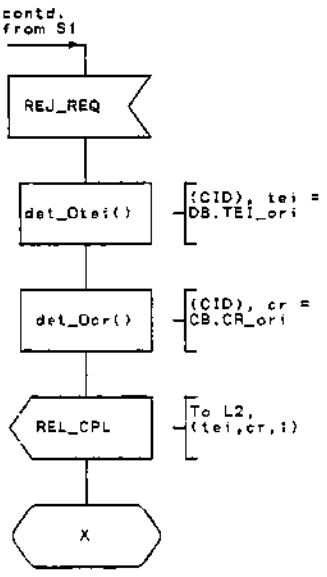

contd.
from A1
CALL_PROC
REQ(CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
get CH_ID()
CH_ID_ori
from DB
CALL
PROCEED
To L2,
(tei,cr,1,
CH_ID)
IN2.1
ALERT_REQ
(CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
is rsp msg
NO
YES
CH_ID_ori
from DB
ALERTING
To L2,
(tei,cr,1,
CH_ID)
IN2.1
ALERTING
To L2,
(tei,cr,1,
H_H)
INFO_REQ
(CID)
det_Otei()
det_Ocr()
INFO
IN2.1

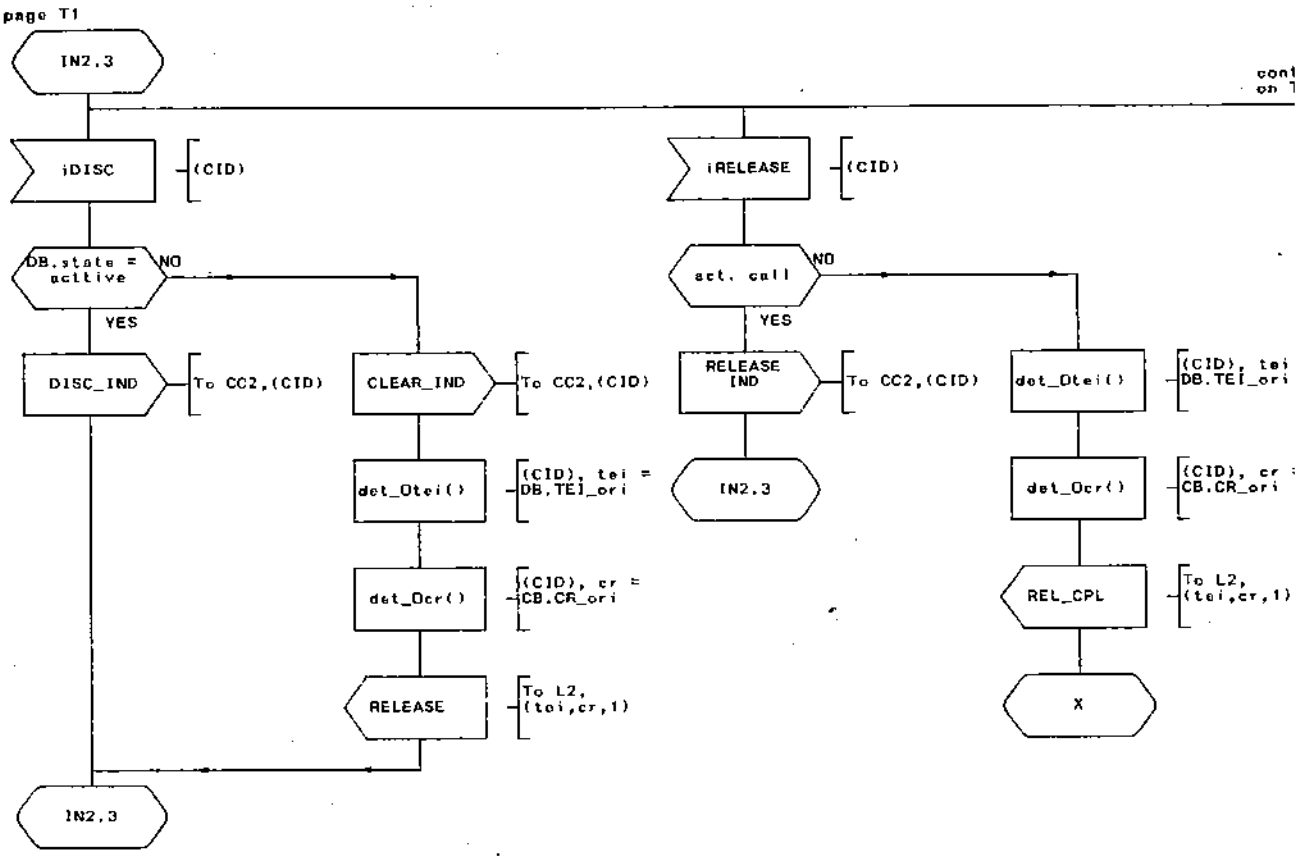
page R3


contd.
from R2
(INFO
(CID)
SETUPACK
REQ
(CID)
INFO _REQ
To CC2, (CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
SETUP ACK
To L2, (tei,
cr, 1)
IN2.1

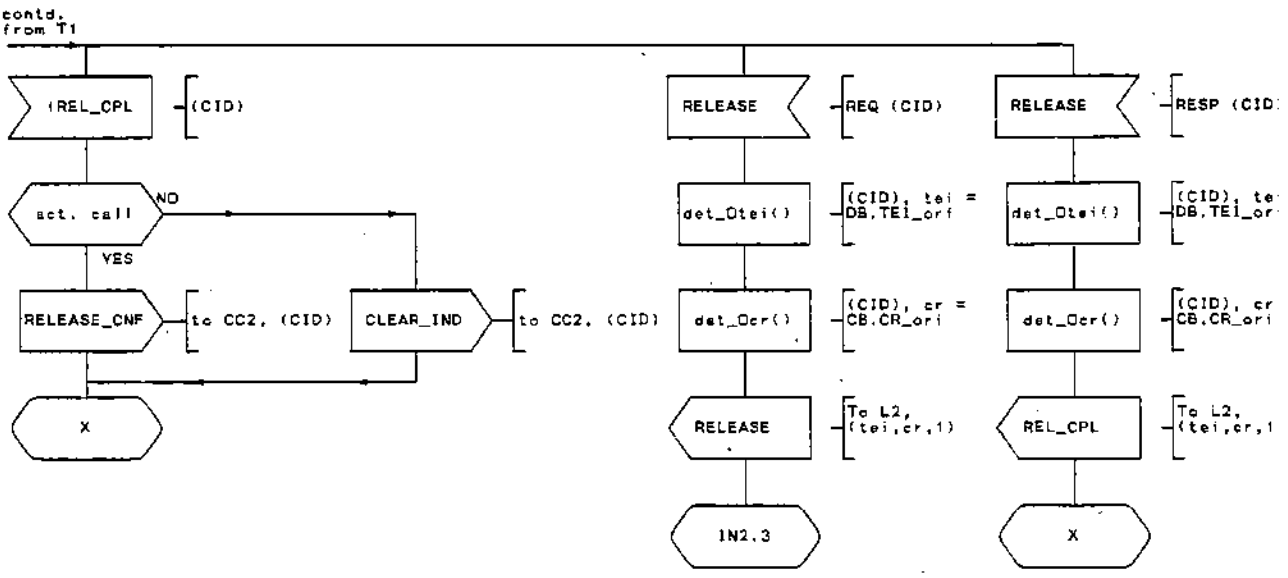

page S1
IN2.2
con
on
↓CONNECT ACK
(CID)
updt DB(CID)
IN2.2
↓DISC
(CID)
updt DB(CID)
DISC_IND
To CC2, (CID)
IN2.3
↓REL_CPL
(CID)
updt DB(CID)
REJ_IND
To CC2, (CID)
X
DISC_REQ
(CID)
det_Otei()
(CID), tei
DB.TEI_ori
det_Ocr()
(CID), cr
CB.CR_ori
DISC
To L2,(tei,cr
IN2.3

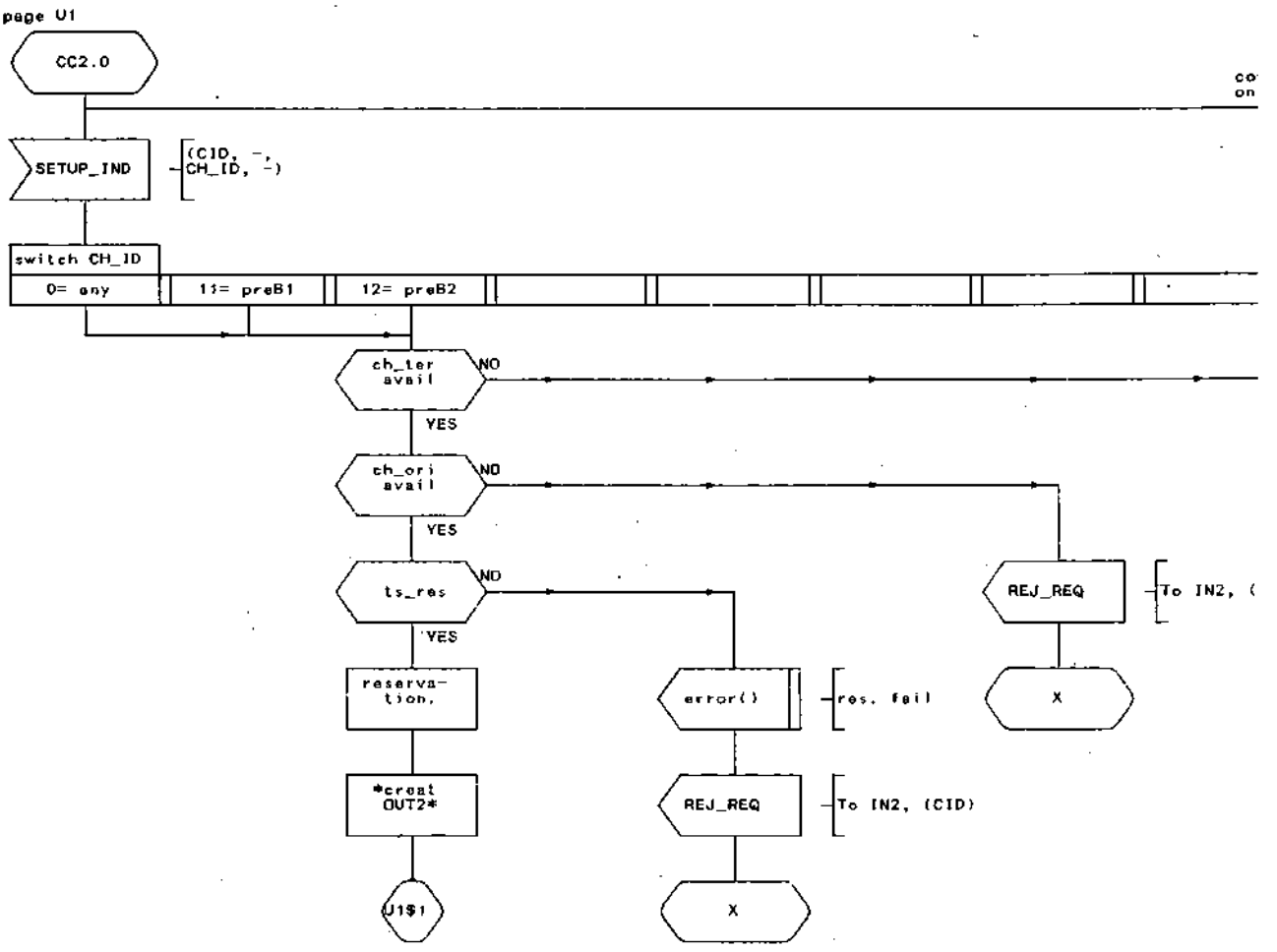
page 52


contd.
from S1
REJ_REQ
det_Otei()
{CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
REL_CPL
To L2,
(tei,cr,1)
X

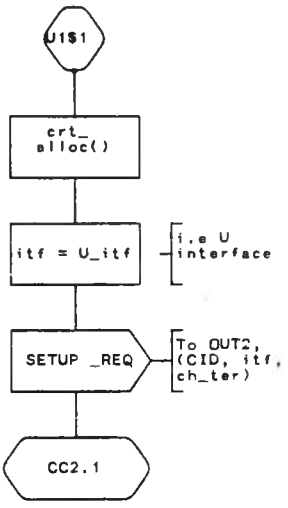

page T1
IN2,3
cont
on T
iDISC
(CID)
DB,state = active
NO
YES
DISC_IND
To CC2,(CID)
CLEAR_IND
To CC2,(CID)
det_Otei()
(CID), tei = DB.TEI_ori
det_Ocr()
(CID), cr = CB.CR_ori
RELEASE
To L2, (tei,cr,1)
IN2,3
iRELEASE
(CID)
act. call
NO
YES
RELEASE IND
To CC2,(CID)
IN2,3
det_Otei()
(CID), tei
DB.TEI_ori
det_Ocr()
(CID), cr
CB.CR_ori
REL_CPL
To L2, (tei,cr,1)
X

page T2


contd.
from T1
(REL_CPL
(CID)
act. call
NO
YES
RELEASE_CNF
to CC2, (CID)
CLEAR_IND
to CC2, (CID)
X
RELEASE
REQ (CID)
det_Otei()
(CID), tei =
DB,TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
RELEASE
To L2,
(tei,cr,1)
IN2.3
RELEASE
RESP (CID
det_Otei()
(CID), te
DB,TEI_or
det_Ocr()
(CID), cr
CB.CR_ori
REL_CPL
To L2,
(tei,cr,1
X


page U1
CC2.0
SETUP_IND
(CID, -,
CH_ID, -)
switch CH_ID
0= any
11= preB1
12= preB2
ch_ter
avail
NO
YES
ch_ori
avail
NO
YES
ts_res
NO
YES
REJ_REQ
To IN2, (
reserva-
tion,
error()
res. fail
X
*creat
OUT2*
REJ_REQ
To IN2, (CID)
U1$1
X

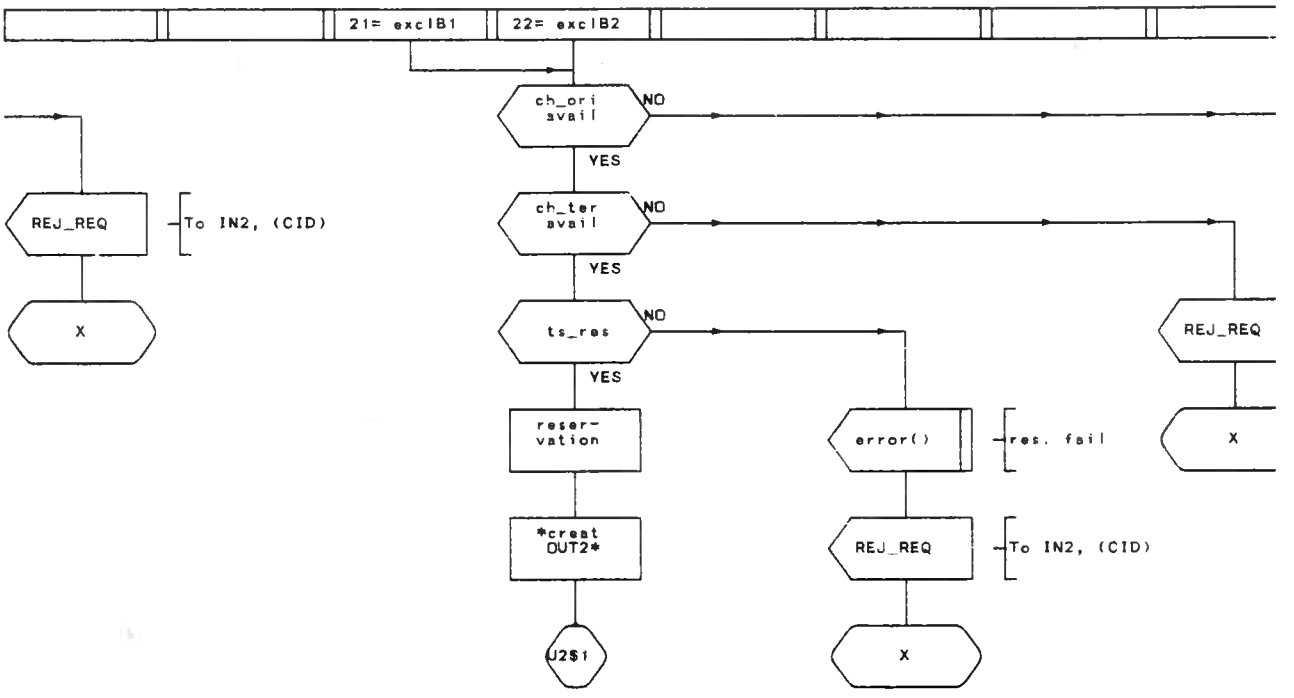
page U1 contd.



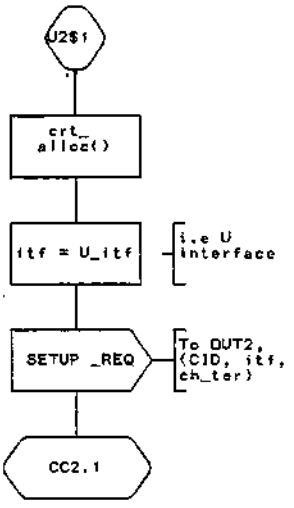
page U2

contd.
from U1

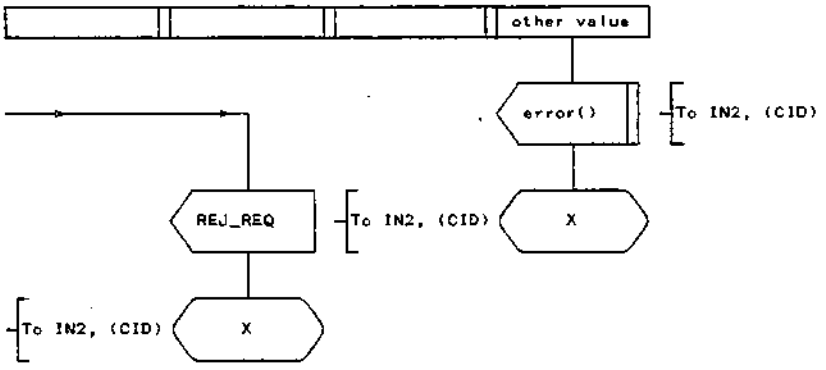
cor
on

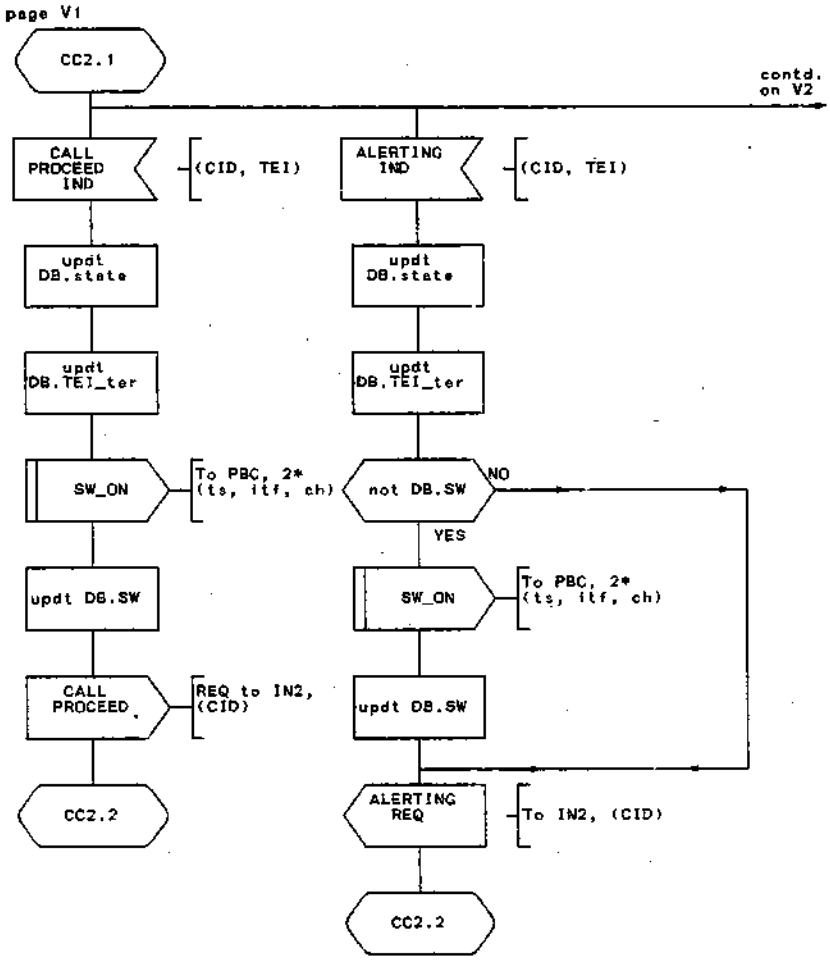


page U2 contd.

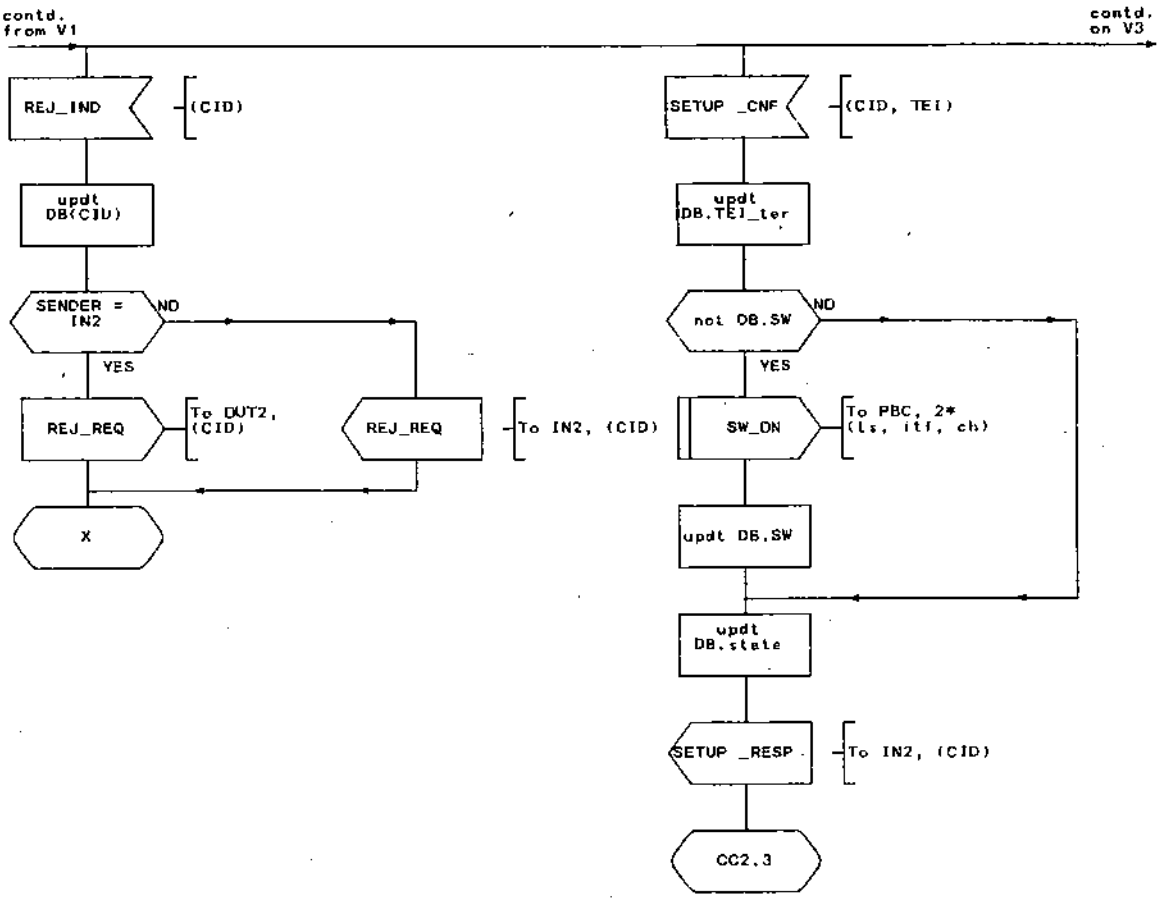


page U3

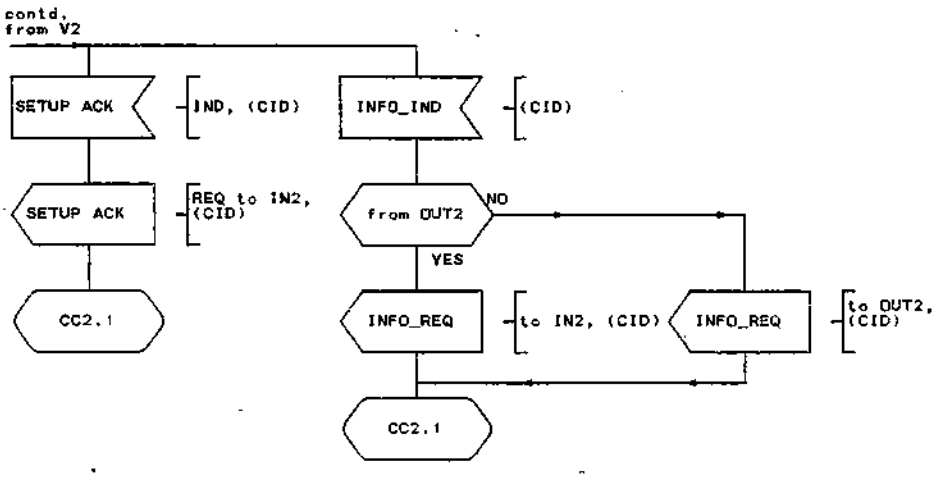


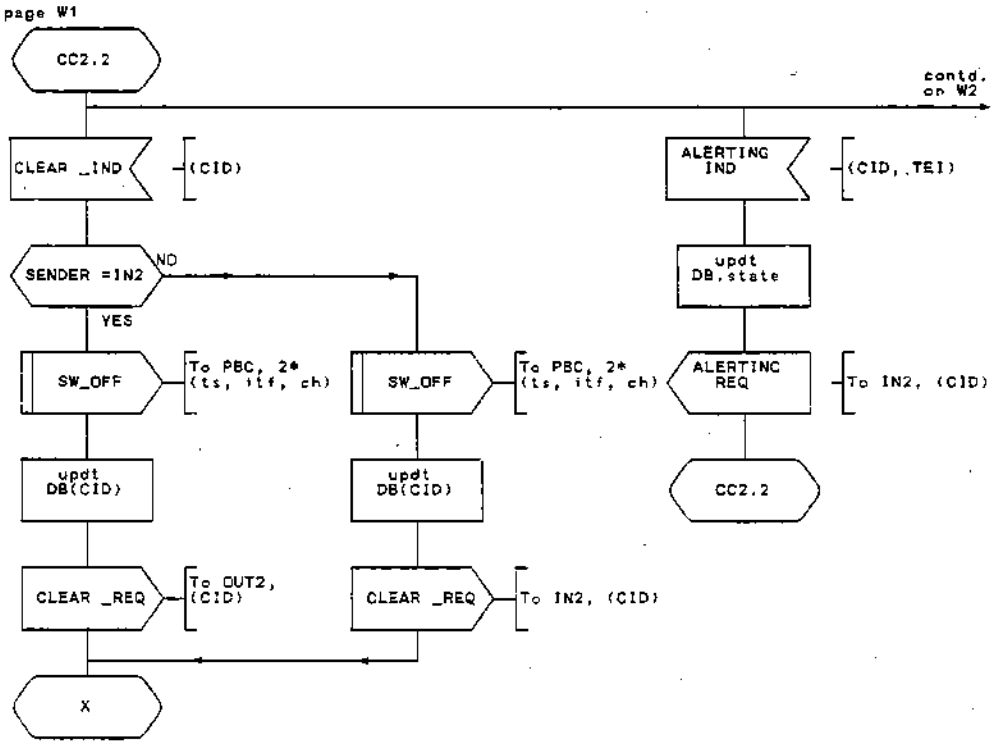

page V1
CC2.1
contd. on V2
CALL PROCEED IND
(CID, TEI)
ALERTING IND
(CID, TEI)
updt DB.state
updt DB.state
updt DB.TEI_ter
updt DB.TEI_ter
SW_ON
To PBC, 2* (ts, itf, ch)
not DB.SW
NO
YES
updt DB.SW
SW_ON
To PBC, 2* (ts, itf, ch)
CALL PROCEED
REQ to IN2, (CID)
updt DB.SW
CC2.2
ALERTING REQ
To IN2, (CID)
CC2.2

page V2

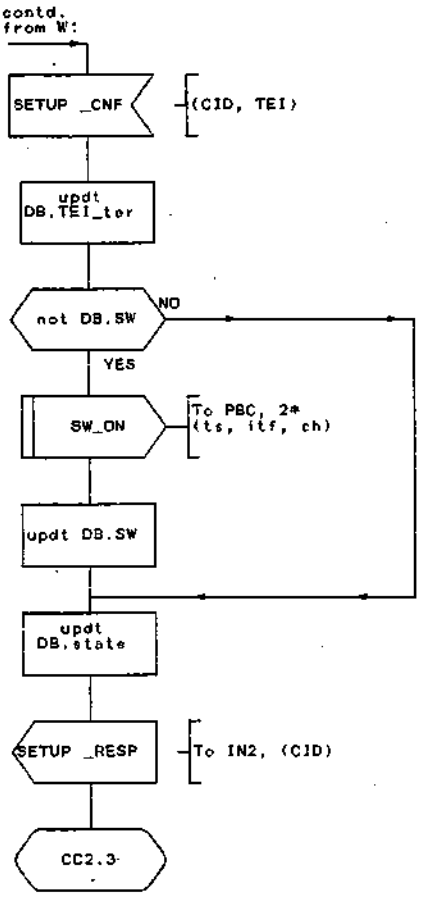

contd.
from V1
contd.
on V3
REJ_IND
(CID)
updt
DB(CID)
SENDER =
IN2
NO
YES
REJ_REQ
To OUT2,
(CID)
REJ_REQ
To IN2, (CID)
X
SETUP _CNF
(CID, TEI)
updt
DB.TEI_ter
not DB.SW
NO
YES
SW_DN
To PBC, 2*
(ls, itf, ch)
updt DB.SW
updt
DB.state
SETUP _RESP
To IN2, (CID)
CC2.3

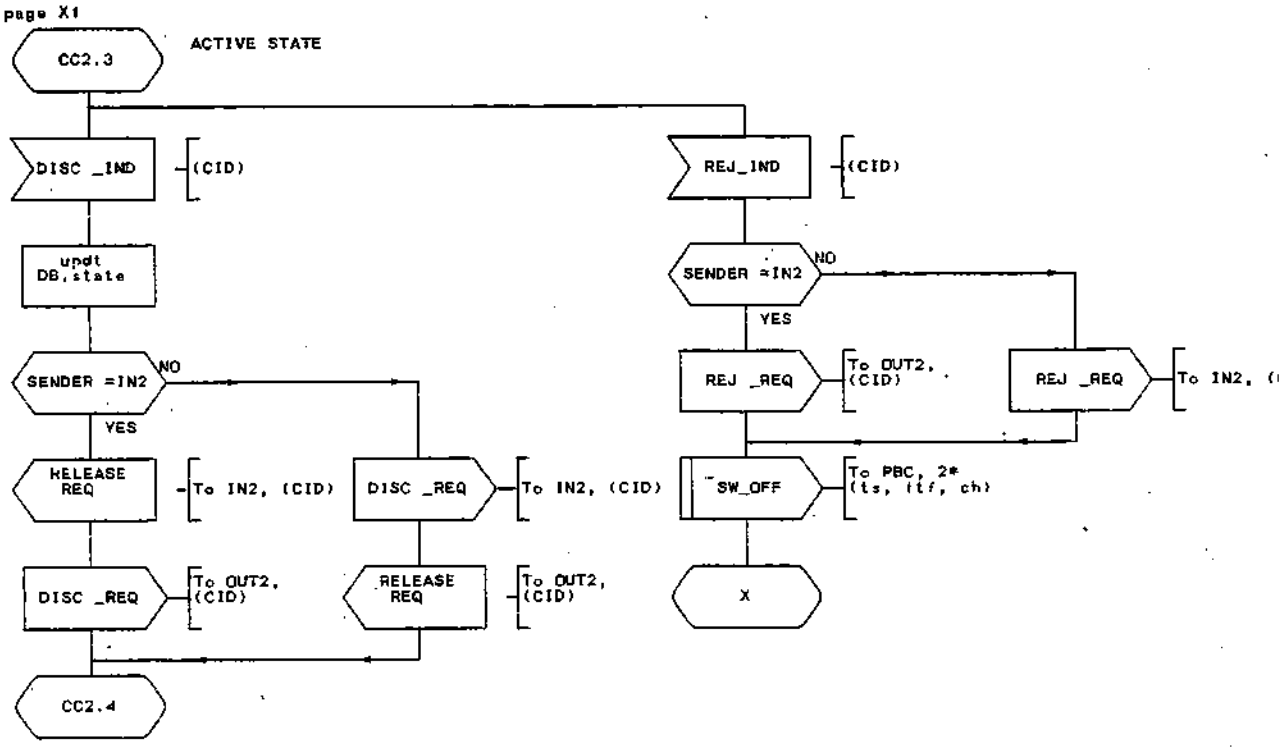
page V3

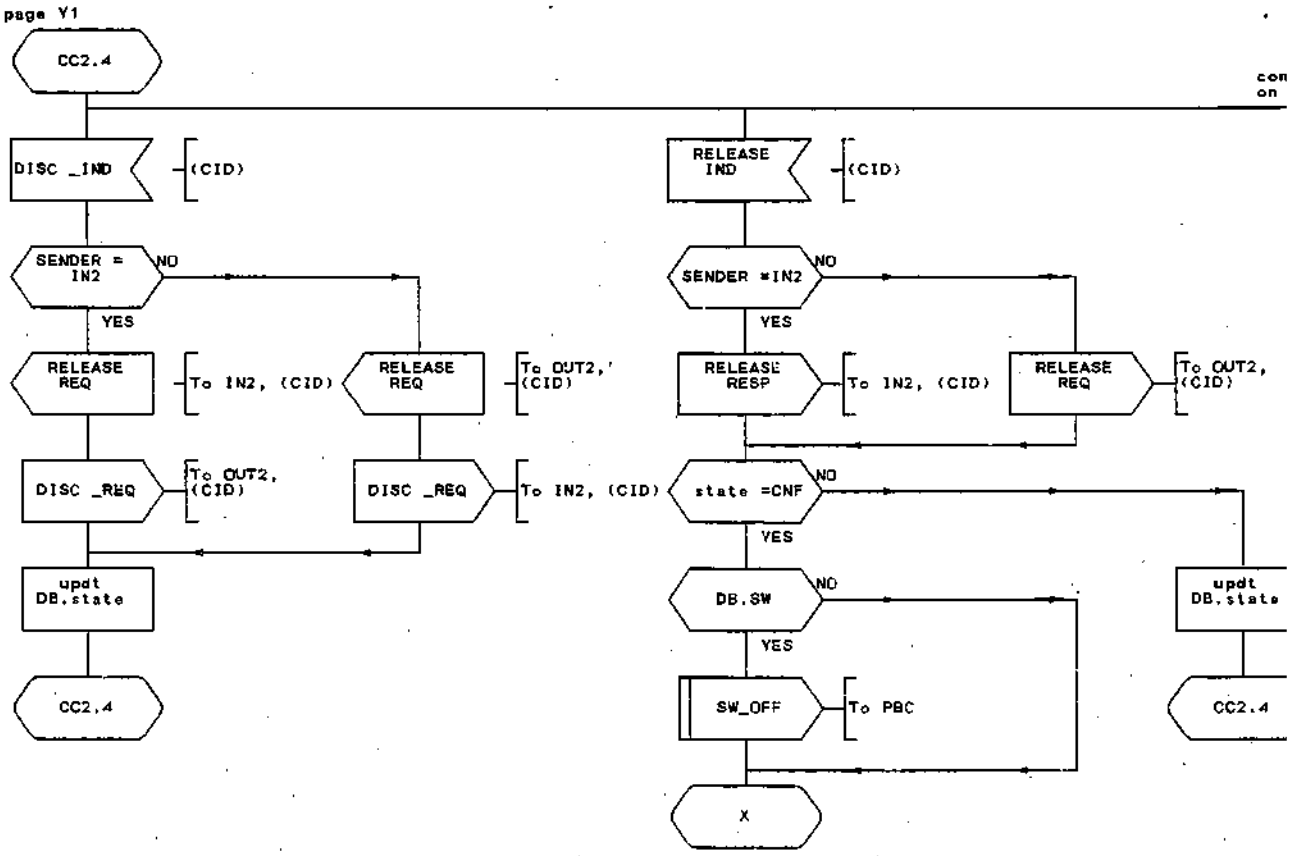

contd,
from V2
SETUP ACK
IND, (CID)
INFO_IND
(CID)
SETUP ACK
REQ to IN2,
(CID)
from OUT2
NO
YES
CC2.1
INFO_REQ
to IN2, (CID)
INFO_REQ
to OUT2,
(CID)
CC2.1


page W1
CC2.2
contd.
on W2
CLEAR _IND
(CID)
ALERTING
IND
(CID, TEI)
SENDER =IN2
NO
YES
updt
DB.state
SW_OFF
To PBC, 2*
(ts, itf, ch)
SW_OFF
To PBC, 2*
(ts, itf, ch)
ALERTINC
REQ
To IN2, (CID)
updt
DB(CID)
updt
DB(CID)
CC2.2
CLEAR _REQ
To OUT2,
(CID)
CLEAR _REQ
To IN2, (CID)
X

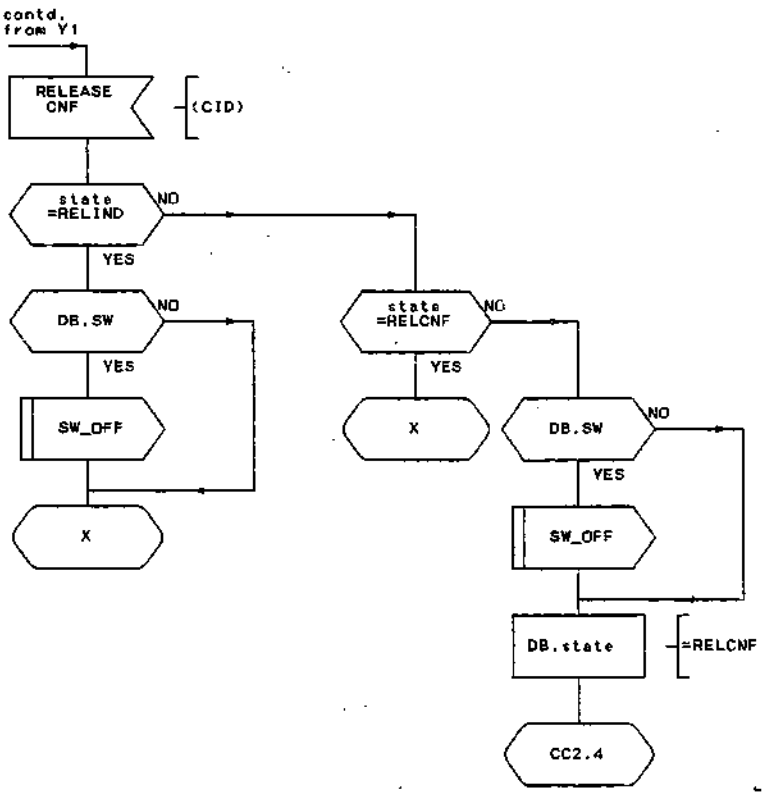
page W2

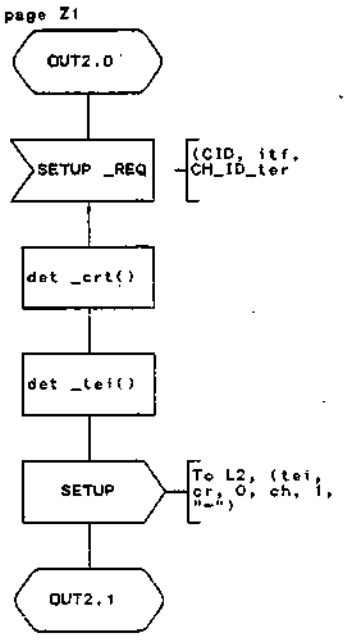

contd.
from W:
SETUP _CNF
(CID, TEI)
updt
DB.TEI_ter
not DB.SW
NO
YES
SW_ON
To PBC, 2*
(ts, itf, ch)
updt DB.SW
updt
DB.state
SETUP _RESP
To IN2, (CID)
CC2.3

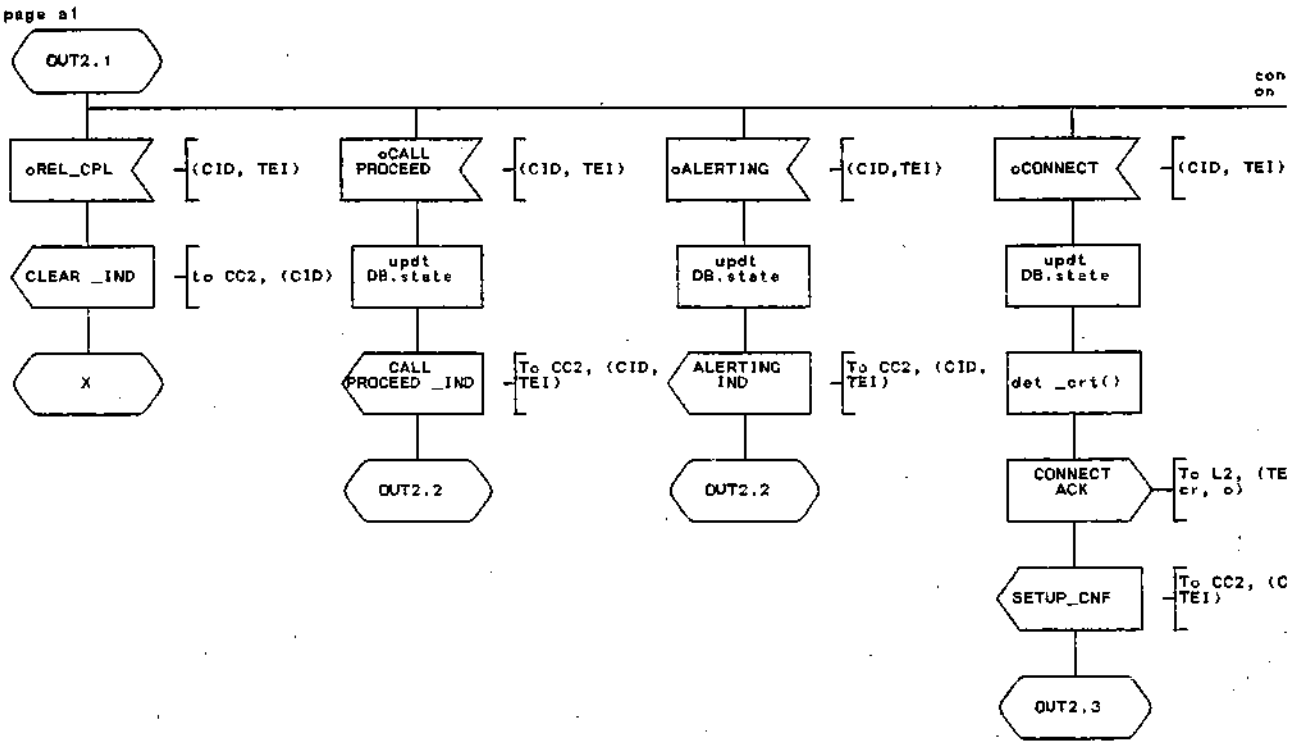

page X1
CC2.3
ACTIVE STATE
DISC _IND
(CID)
updt
DB, state
SENDER =IN2
NO
YES
RELEASE
REQ
To IN2, (CID)
DISC _REQ
To OUT2,
(CID)
DISC _REQ
To IN2, (CID)
RELEASE
REQ
To OUT2,
(CID)
CC2.4
REJ_IND
(CID)
SENDER ≠IN2
NO
YES
REJ _REQ
To OUT2,
(CID)
REJ _REQ
To IN2, (
SW_OFF
To PBC, 2*
(ts, (tf, ch)
X


page Y1
CC2.4
con
on
DISC _IND
(CID)
SENDER = IN2
NO
YES
RELEASE REQ
To IN2, (CID)
RELEASE REQ
To OUT2,' (CID)
DISC _REQ
To OUT2, (CID)
DISC _REQ
To IN2, (CID)
updt DB.state
CC2.4
RELEASE IND
(CID)
SENDER *IN2
NO
YES
RELEASE RESP
To IN2, (CID)
RELEASE REQ
To OUT2, (CID)
state =CNF
NO
YES
DB.SW
NO
YES
SW_OFF
To PBC
X
updt DB.state
CC2.4

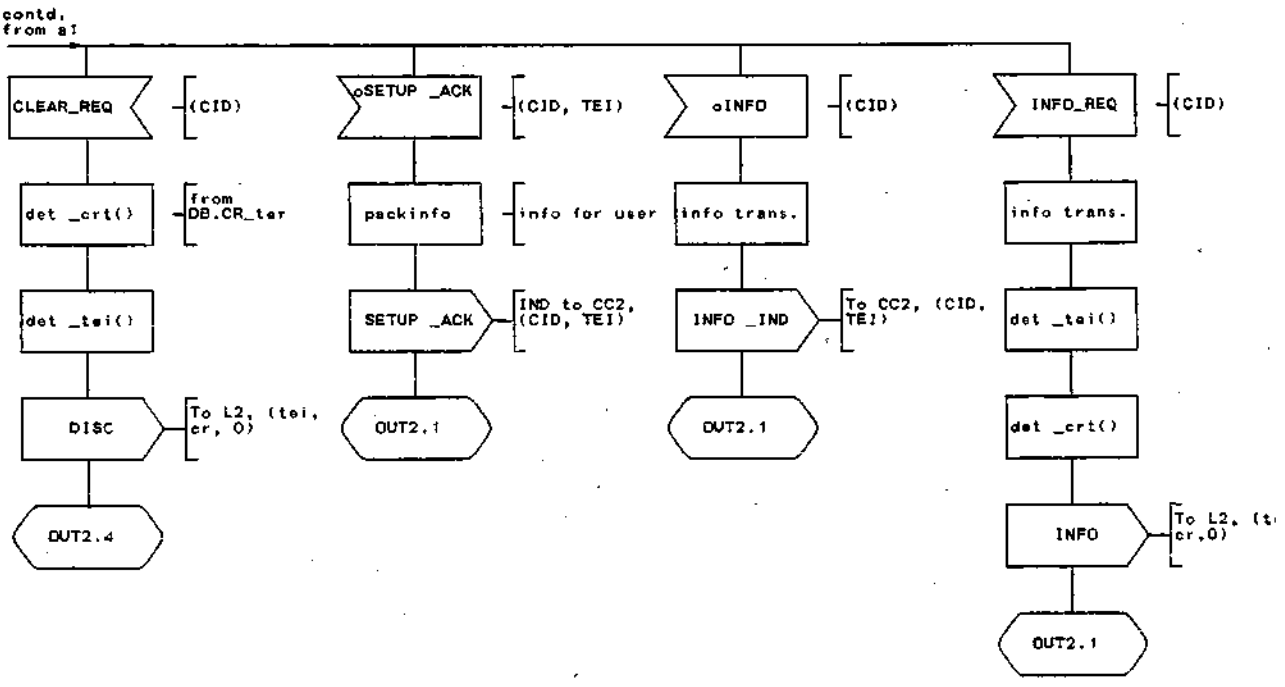
page Y2


contd.
from Y1
RELEASE CNF
(CID)
state =RELIND
NO
YES
DB.SW
NO
YES
SW_OFF
X
state =RELCNF
NO
YES
X
DB.SW
NO
YES
SW_OFF
DB.state
=RELCNF
CC2.4

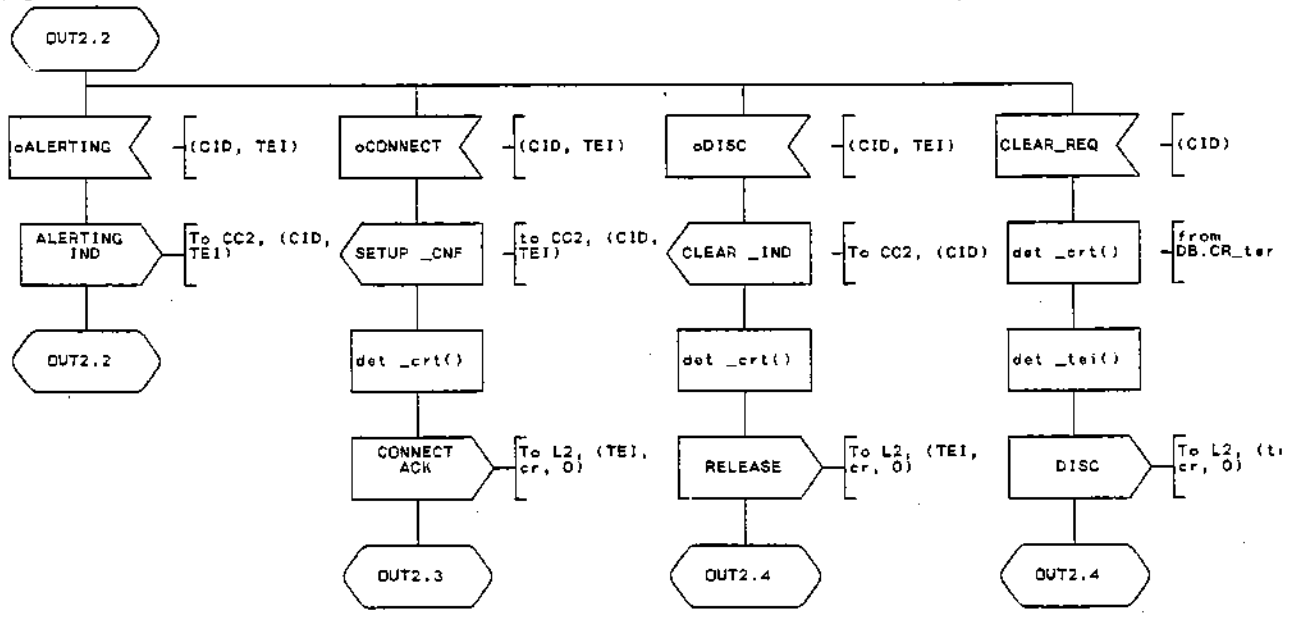

page Z1
OUT2.0
SETUP _REQ
(CID, itf,
CH_ID_ter
det _crt()
det _tei()
SETUP
To L2, (tei,
cr, 0, ch, 1,
"-")
OUT2.1

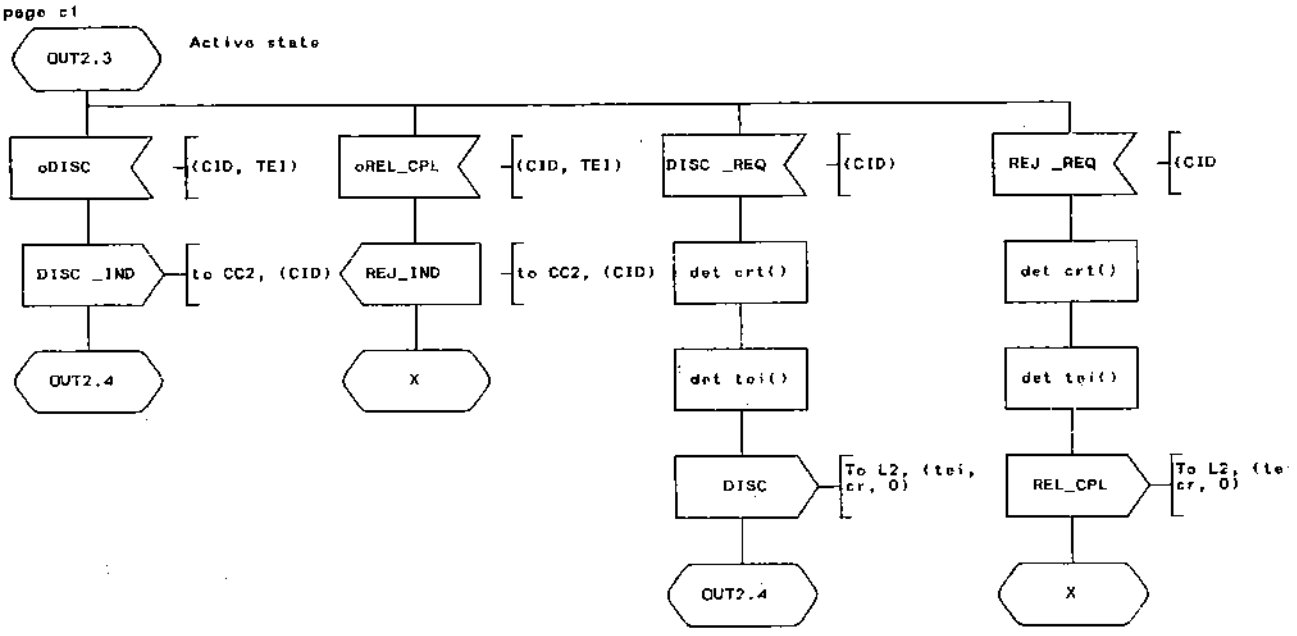

page a1
OUT2.1
con
on
oREL_CPL
(CID, TEI)
oCALL PROCEED
(CID, TEI)
oALERTING
(CID,TEI)
oCONNECT
(CID, TEI)
CLEAR _IND
to CC2, (CID)
updt DB.state
updt DB.state
updt DB.state
X
CALL PROCEED _IND
To CC2, (CID, TEI)
ALERTING IND
To CC2, (CID, TEI)
det _crt()
OUT2.2
OUT2.2
CONNECT ACK
To L2, (TE
er, o)
SETUP_CNF
To CC2, (C
TEI)
OUT2.3

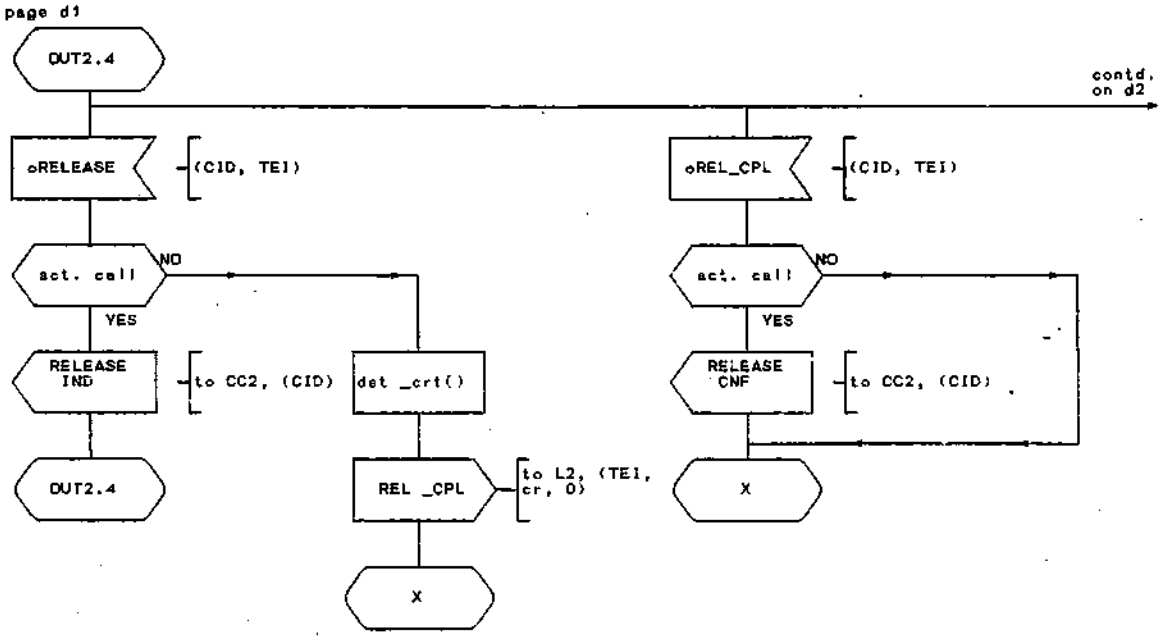
page a2


contd.
from a1
CLEAR_REQ
(CID)
oSETUP _ACK
(CID, TEI)
oINFO
(CID)
INFO_REQ
(CID)
det _crt()
from DB.CR_tar
packinfo
info for user
info trans.
info trans.
det _tei()
SETUP _ACK
IND to CC2, (CID, TEI)
INFO _IND
To CC2, (CID, TEI)
det _tei()
DISC
To L2, (tei, cr, 0)
OUT2.1
OUT2.1
det _crt()
OUT2.4
INFO
To L2, (t
cr,0)
OUT2.1

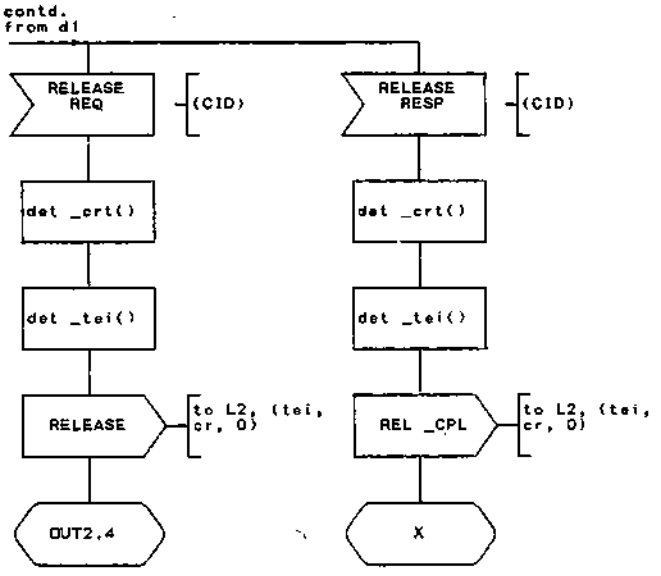
page b1

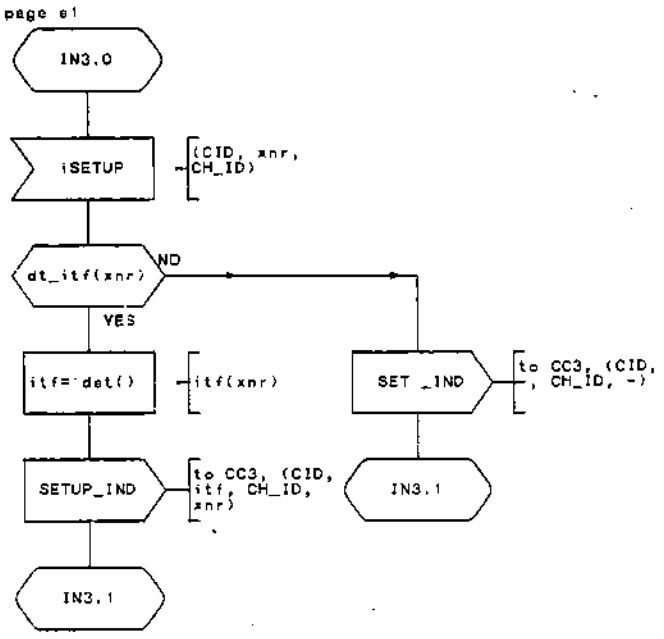

OUT2.2
oALERTING
(CID, TEI)
oCONNECT
(CID, TEI)
oDISC
(CID, TEI)
CLEAR_REQ
(CID)
ALERTING IND
To CC2, (CID, TEI)
SETUP _CNF
to CC2, (CID, TEI)
CLEAR _IND
To CC2, (CID)
det _crt()
from DB.CR_ter
OUT2.2
det _crt()
det _crt()
det _tei()
CONNECT ACK
To L2, (TEI, cr, 0)
RELEASE
To L2, (TEI, cr, 0)
DISC
To L2, (t
cr, 0)
OUT2.3
OUT2.4
OUT2.4

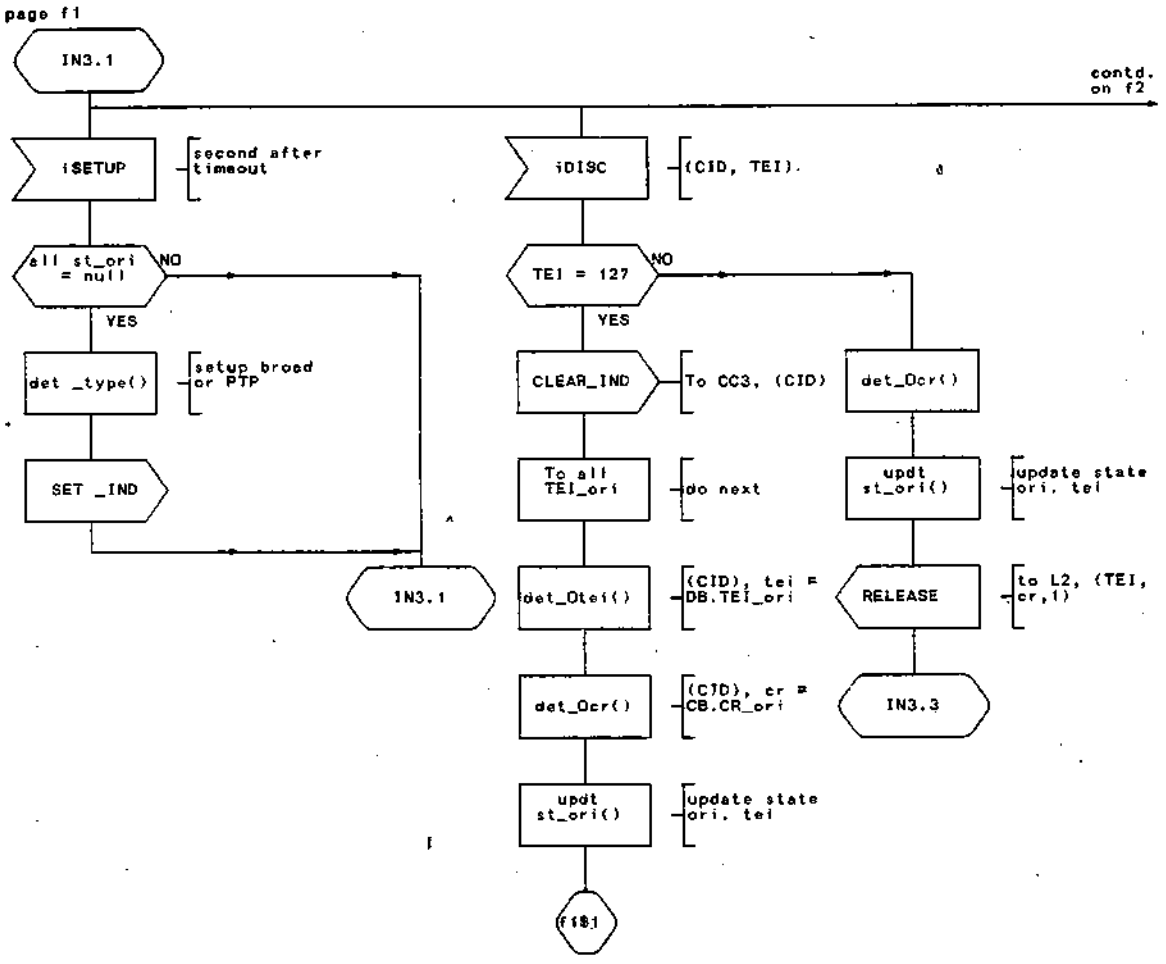

page c1
OUT2.3
Active state
oDISC
(CID, TEI)
oREL_CPL
(CID, TEI)
DISC _REQ
(CID)
REJ _REQ
(CID
DISC _IND
to CC2, (CID)
REJ_IND
to CC2, (CID)
det crt()
det crt()
OUT2.4
X
det tei()
det tei()
DISC
To L2, (tei, cr, 0)
REL_CPL
To L2, (te cr, 0)
OUT2.4
X


page d1
OUT2.4
contd.
on d2
oRELEASE
(CID, TEI)
oREL_CPL
(CID, TEI)
act. call
NO
YES
act. call
NO
YES
RELEASE
IND
to CC2, (CID)
det _crt()
RELEASE
CNF
to CC2, (CID)
OUT2.4
REL _CPL
to L2, (TEI,
cr, 0)
X
X

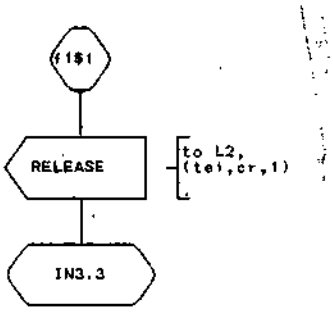
page d2


contd.
from d1
RELEASE REQ
(CID)
RELEASE RESP
(CID)
det _crt()
det _crt()
det _tei()
det _tei()
RELEASE
to L2, (tei, cr, 0)
REL _CPL
to L2, (tei, cr, 0)
OUT2.4
X

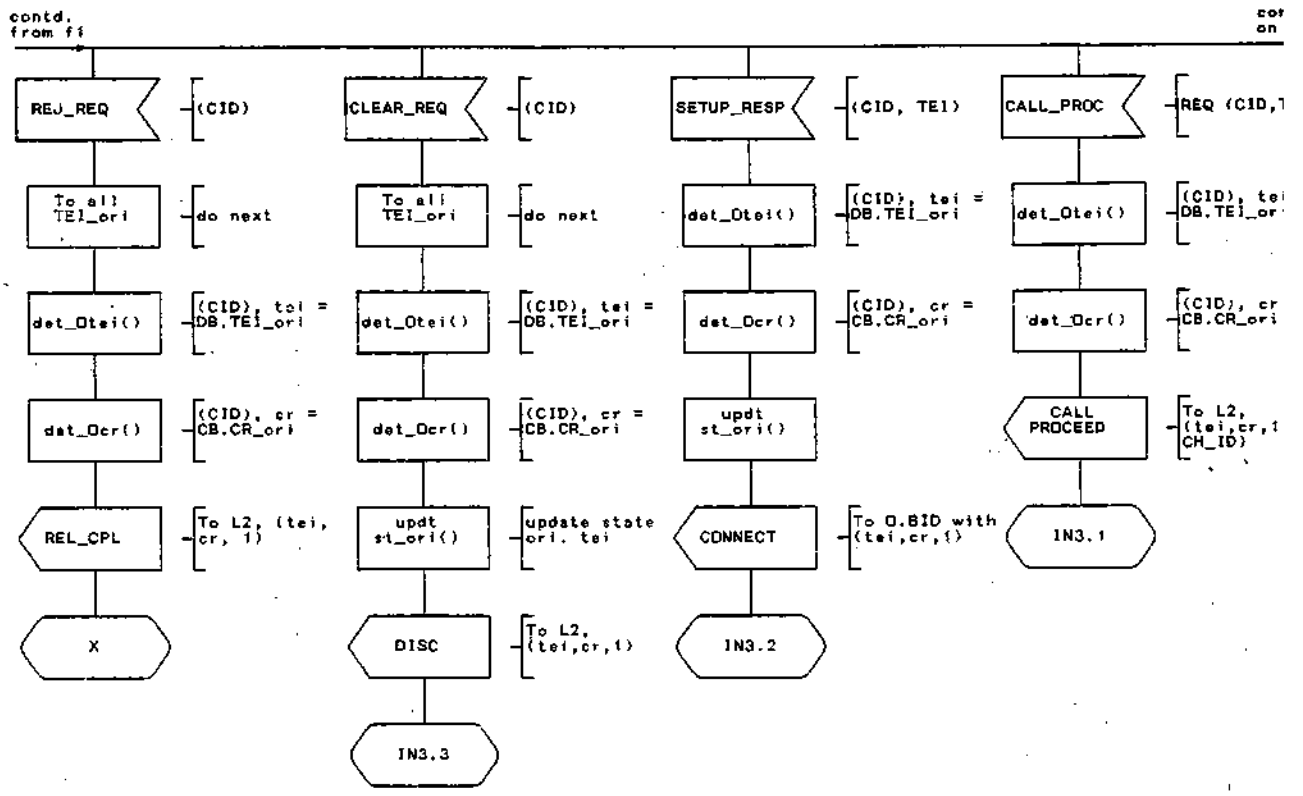

page e1
IN3.0
iSETUP
(CID, xnr, CH_ID)
dt_itf(xnr)
ND
YES
itf=dat()
itf(xnr)
SET _IND
to CC3, (CID, , CH_ID, -)
SETUP_IND
to CC3, (CID, itf, CH_ID, xnr)
IN3.1
IN3.1


page f1
IN3.1
contd. on f2
iSETUP
second after timeout
iDISC
(CID, TEI).
all st_ori = null
NO
YES
TEI = 127
NO
YES
det _type()
setup broad or PTP
CLEAR_IND
To CC3, (CID)
det_Ocr()
SET _IND
To all TEI_ori
do next
updt st_ori()
update state ori, tei
IN3.1
det_Otei()
(CID), tei = DB.TEI_ori
RELEASE
to L2, (TEI, cr,1)
det_Ocr()
(CID), cr = CB.CR_ori
IN3.3
updt st_ori()
update state ori, tei
f1$1

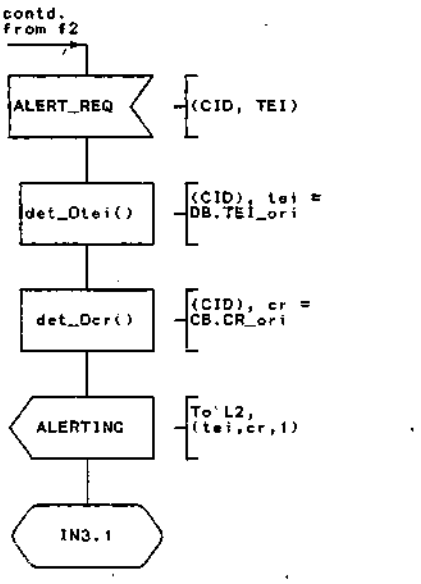
page f1 contd.

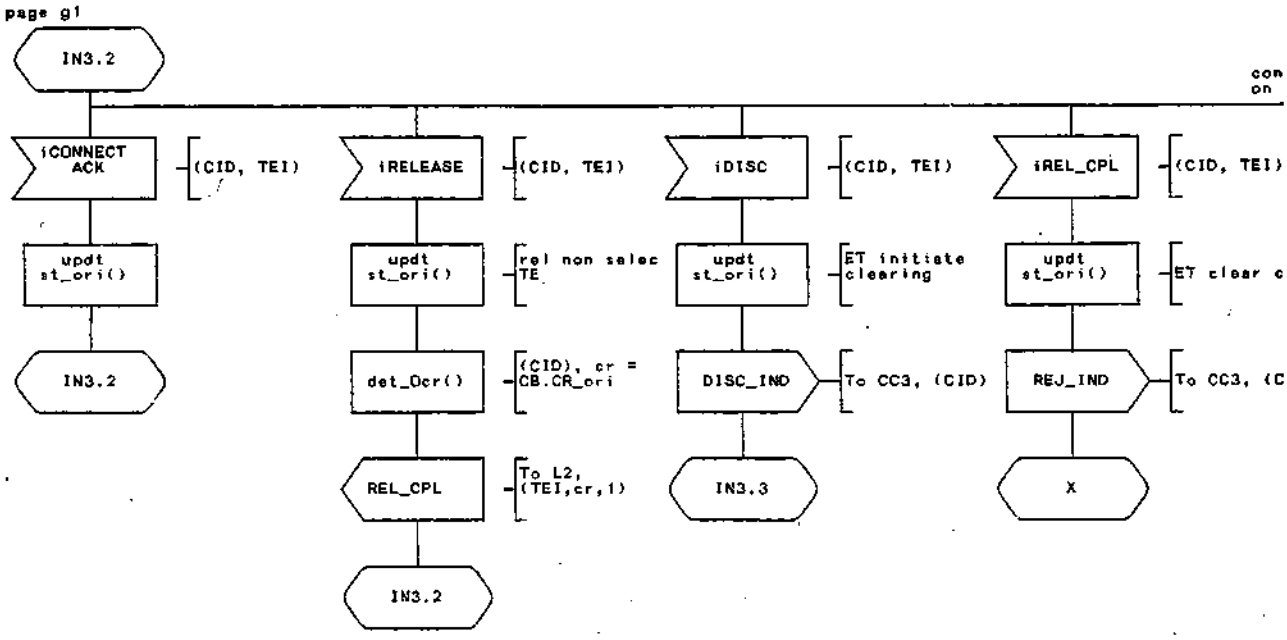


page f2

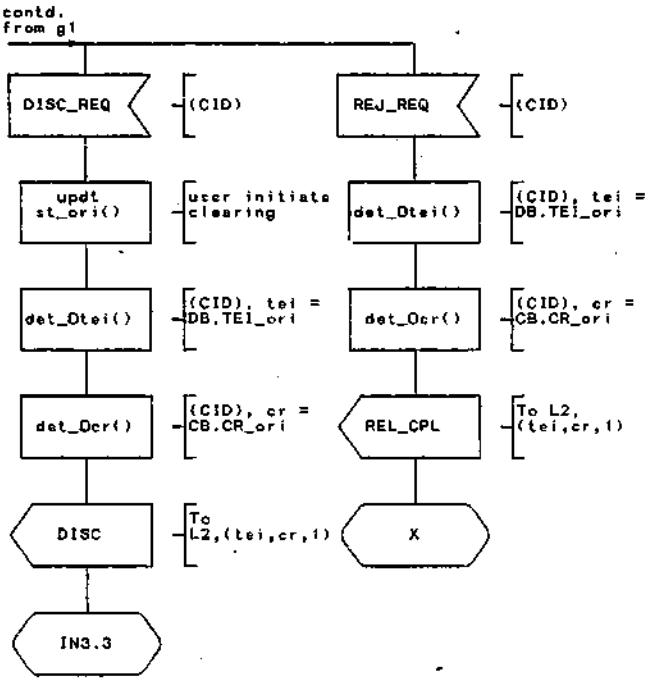

contd.
from f1
REJ_REQ
(CID)
To all
TEI_ori
do next
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
REL_CPL
To L2, (tei,
cr, 1)
X
CLEAR_REQ
(CID)
To all
TEI_ori
do next
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
updt
st_ori()
update state
ori. tei
DISC
To L2,
(tei,cr,1)
IN3.3
SETUP_RESP
(CID, TEI)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
updt
st_ori()
CONNECT
To D.BID with
(tei,cr,1)
IN3.2
CALL_PROC
det_Otei()
det_Ocr()
CALL
PROCEED
To L2,
((tei,cr,1
CH_ID)
IN3.1

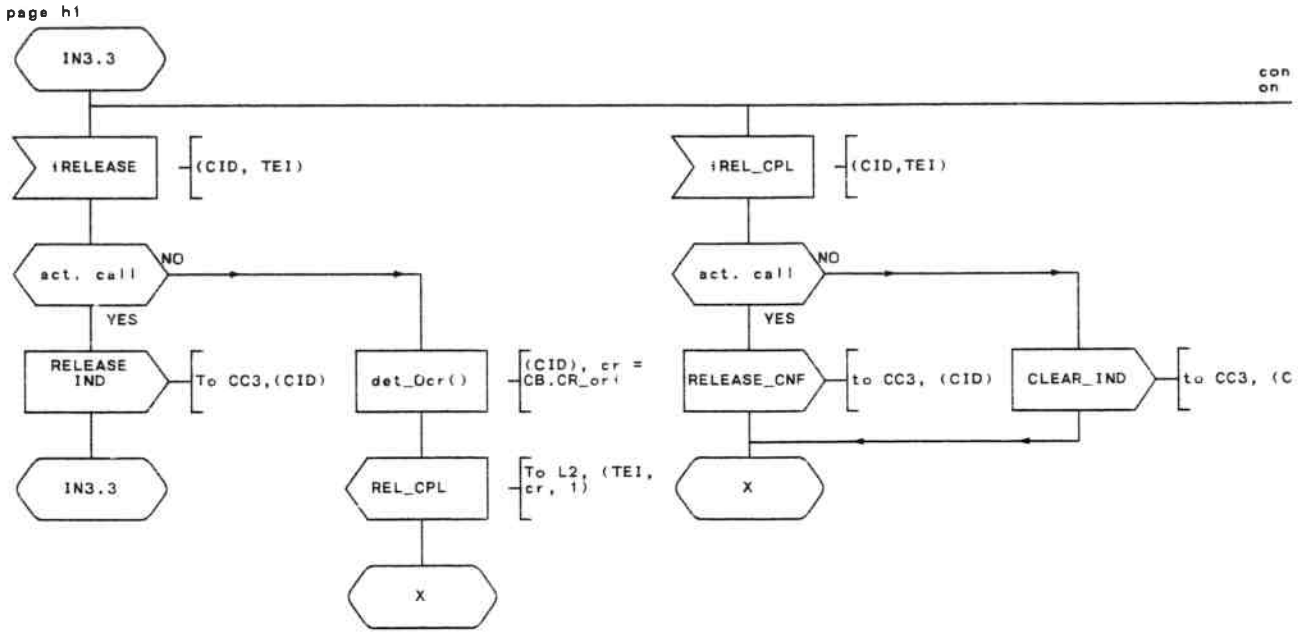
page f3


contd.
from f2
ALERT_REQ
(CID, TEI)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
ALERTING
To`L2,
(tei,cr,1)
IN3.1

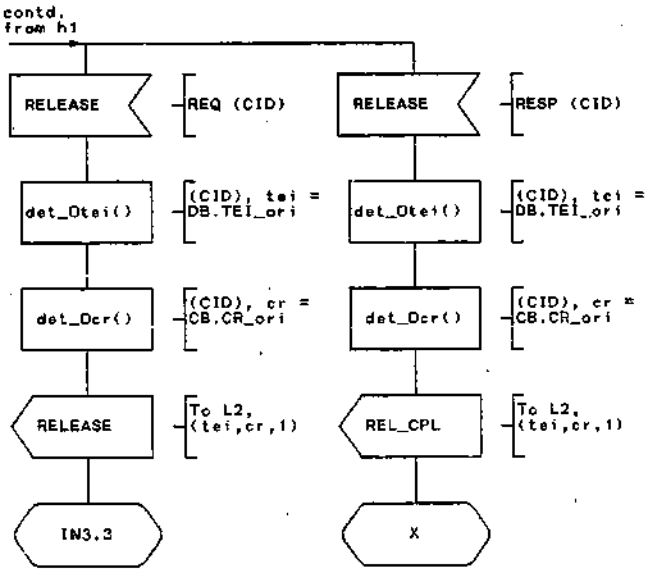

page g1
IN3.2
con
on
iCONNECT ACK
(CID, TEI)
iRELEASE
(CID, TEI)
iDISC
(CID, TEI)
iREL_CPL
(CID, TEI)
updt st_ori()
updt st_ori()
rel non selec TE
updt st_ori()
ET initiate clearing
updt st_ori()
ET clear c
IN3.2
det_Ocr()
(CID), cr = CB.CR_ori
DISC_IND
To CC3, (CID)
REJ_IND
To CC3, (C
REL_CPL
To L2, (TEI,cr,1)
IN3.3
X
IN3.2

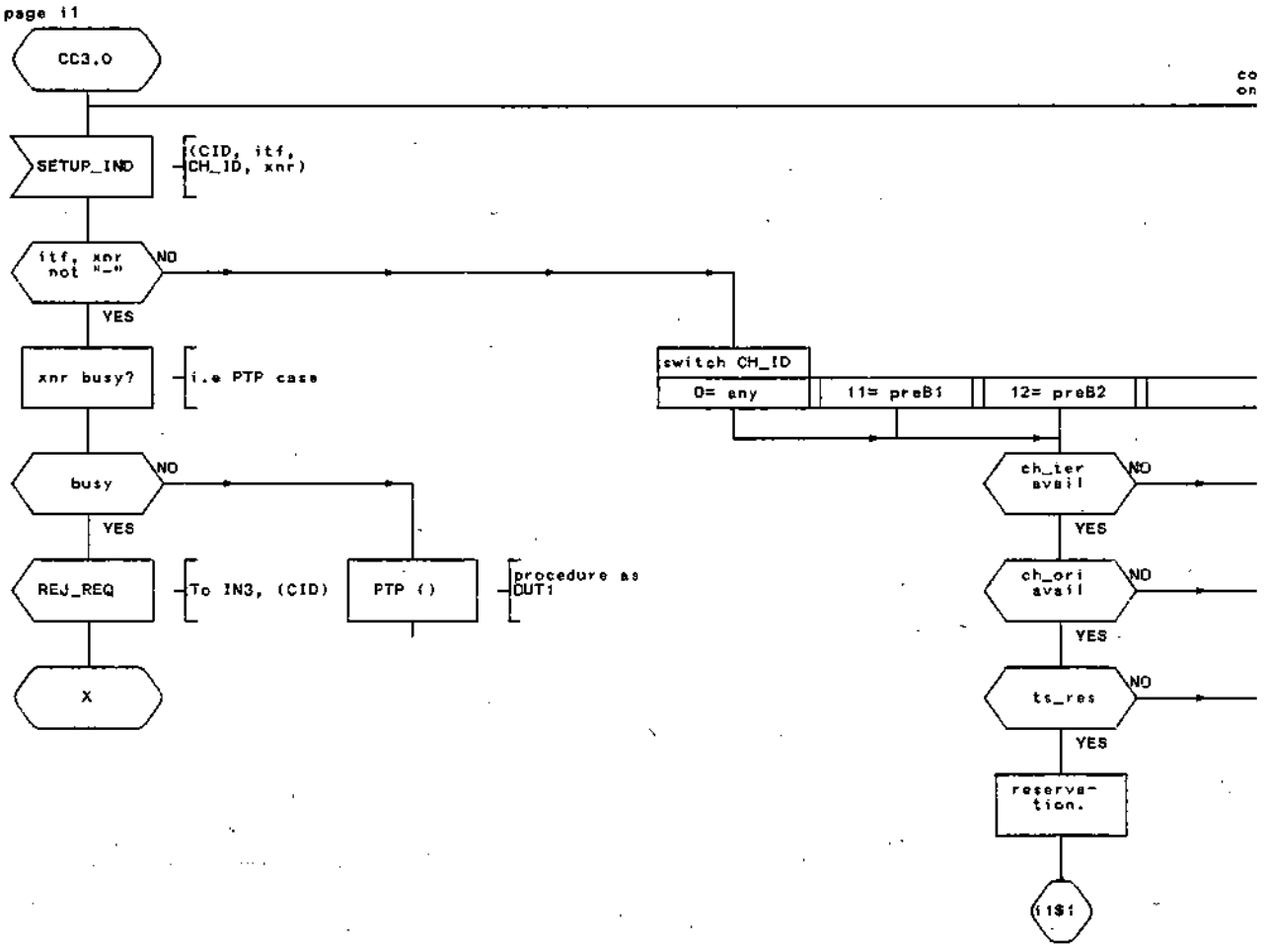
page g2


contd.
from g1
DISC_REQ
(CID)
REJ_REQ
(CID)
updt st_ori()
user initiate clearing
det_Otei()
(CID), tei = DB.TEI_ori
det_Otei()
(CID), tei = DB.TEI_ori
det_Ocr()
(CID), cr = CB.CR_ori
det_Ocr()
(CID), cr = CB.CR_ori
REL_CPL
To L2, (tei,cr,1)
DISC
To L2,(tei,cr,1)
X
IN3.3

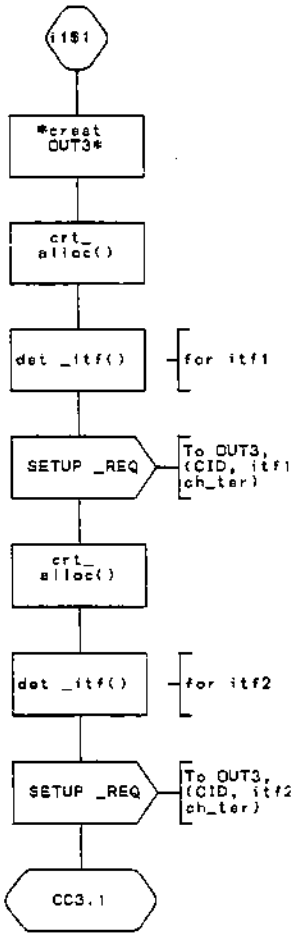

page h1
IN3.3
con
on
iRELEASE
(CID, TEI)
act. call
NO
YES
RELEASE IND
To CC3,(CID)
IN3.3
det_Ocr()
(CID), cr =
CB.CR_or
REL_CPL
To L2, (TEI,
cr, 1)
X
iREL_CPL
(CID,TEI)
act. call
NO
YES
RELEASE_CNF
to CC3, (CID)
CLEAR_IND
to CC3, (C
X

page h2

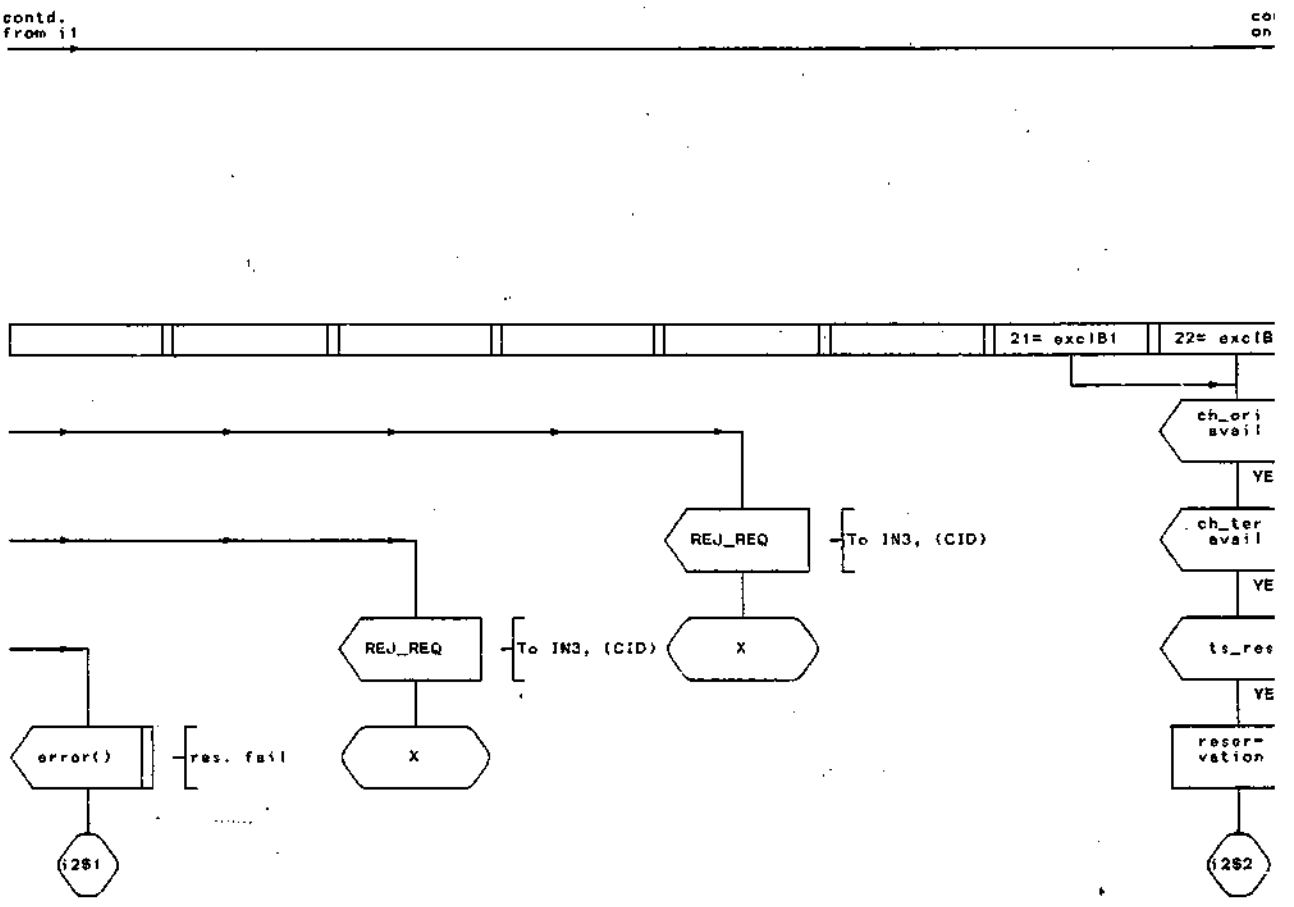

contd.
from h1
RELEASE
REQ (CID)
RELEASE
RESP (CID)
det_Otei()
(CID), tei =
DB.TEI_ori
det_Otei()
(CID), tei =
DB.TEI_ori
det_Ocr()
(CID), cr =
CB.CR_ori
det_Ocr()
(CID), cr =
CB.CR_ori
RELEASE
To L2,
(tei,cr,1)
REL_CPL
To L2,
(tei,cr,1)
IN3.3
X

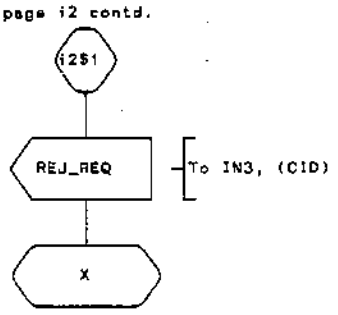

page i1
CC3.0
SETUP_IND
(CID, itf, CH_ID, xnr)
itf, xnr not "-"
NO
YES
xnr busy?
i.e PTP case
busy
NO
YES
REJ_REQ
To IN3, (CID)
PTP ()
procedure as OUT1
X
switch CH_ID
0= any
11= preB1
12= preB2
ch_ter avail
NO
YES
ch_ori avail
NO
YES
ts_res
NO
YES
reserva- tion.
i1$1

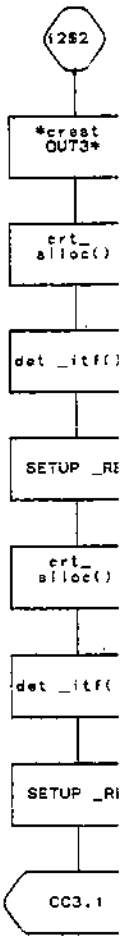
page i1 contd.



page i2


contd.
from i1
21= exclB1
22= exclB
ch_ori
avail
YE
ch_ter
avail
YE
ts_res
YE
reser=
vation
i2$2
REJ_REQ
To IN3, (CID)
X
REJ_REQ
To IN3, (CID)
X
error()
res. fail
i2$1

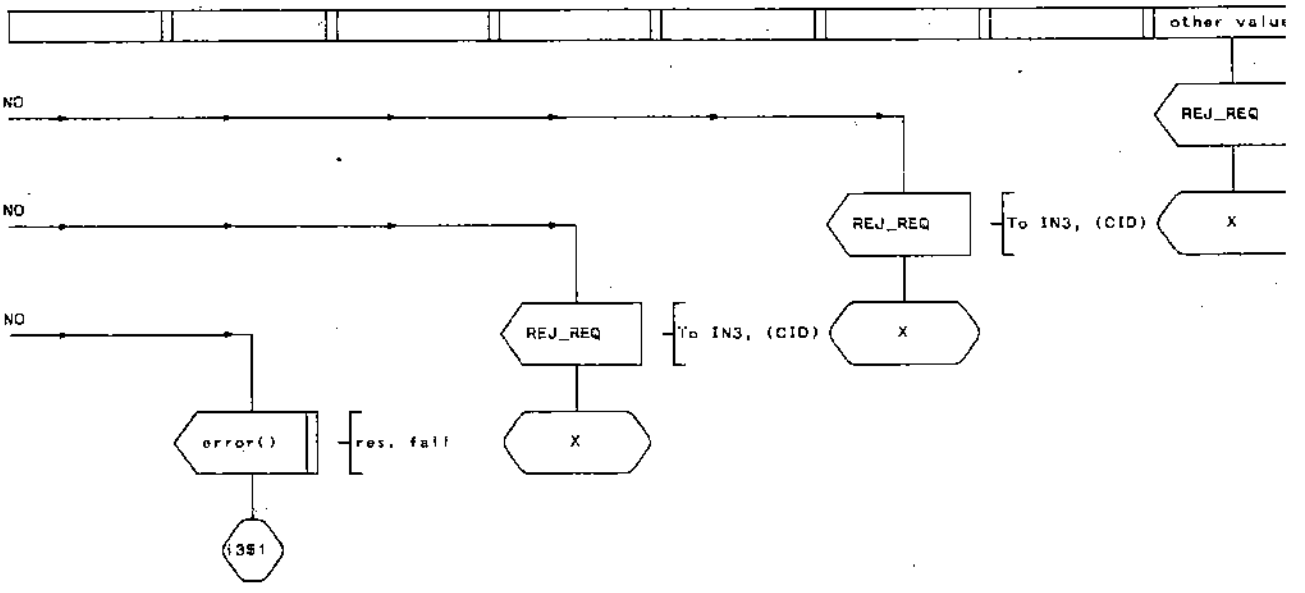

page i2 contd.
i2$1
REJ_REQ
To IN3, (CID)
X


i2$2
*creat
OUT3*
ert_
alloc()
det _itf()
SETUP _RE
ert_
alloc()
det _itf(
SETUP _RI
CC3.1

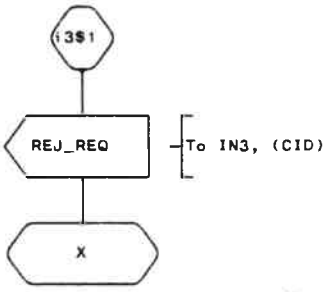
page i3

contd.
from i2

cont
on i



page i3 contd.



for itf1

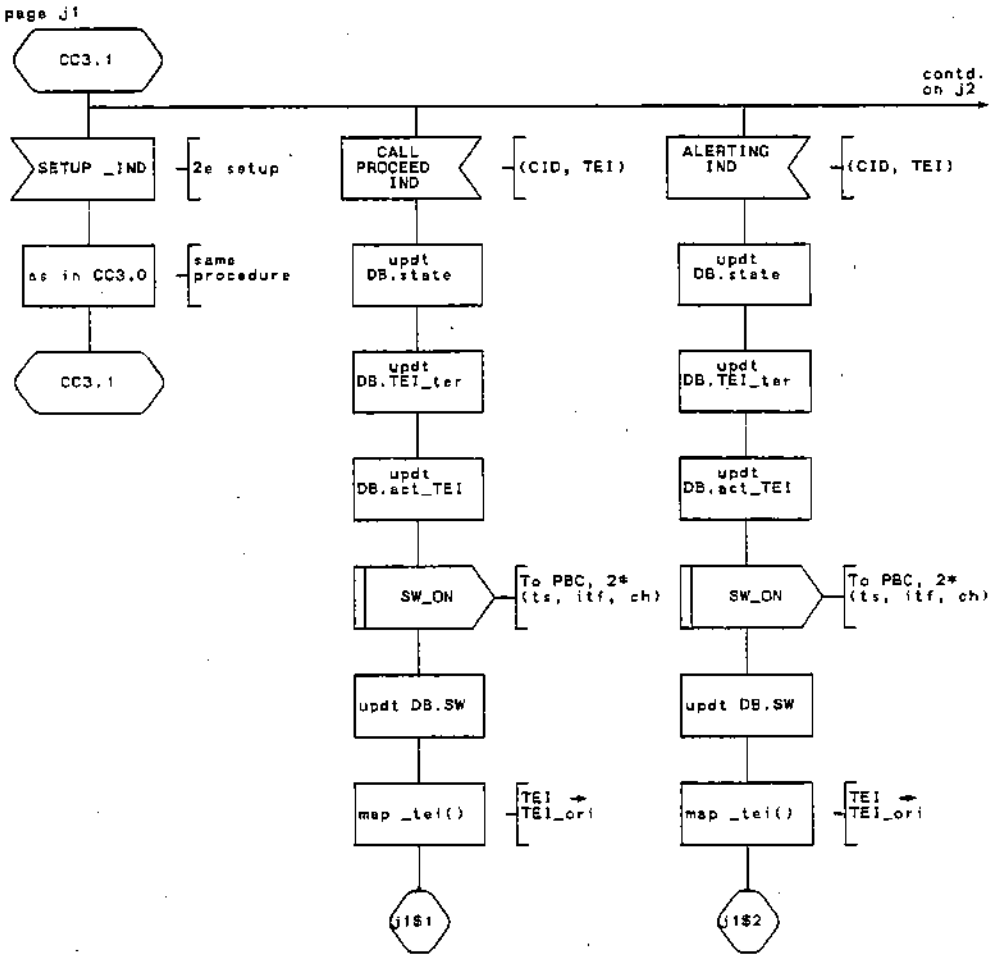
To OUT3,
(CID, itf1,
ch_ter)

for itf2

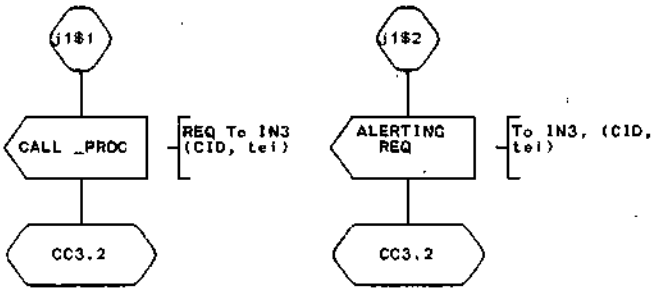
To OUT3,
(CID, itf2,
ch_ter)

page 14

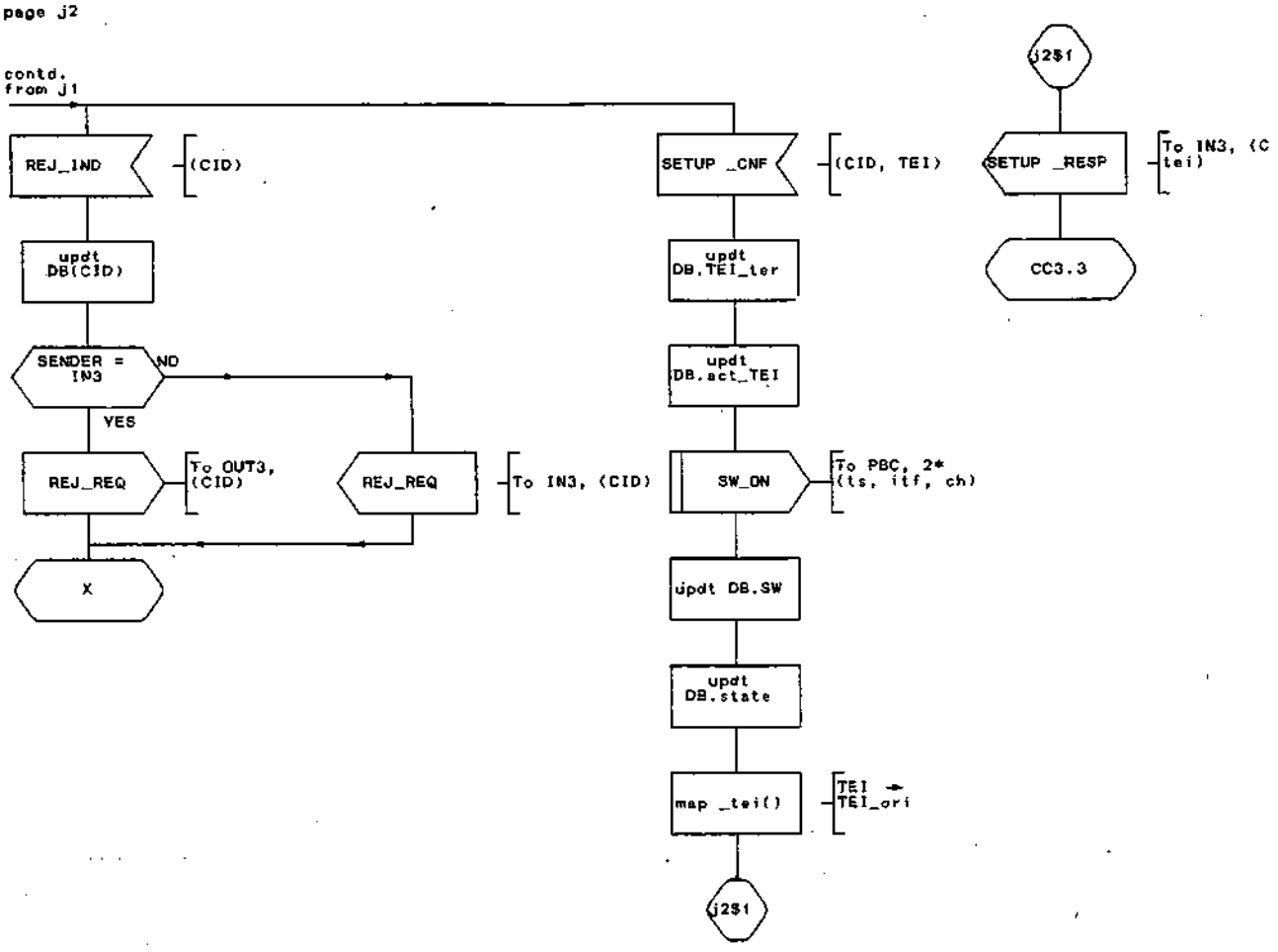
To IN3, (C1D)

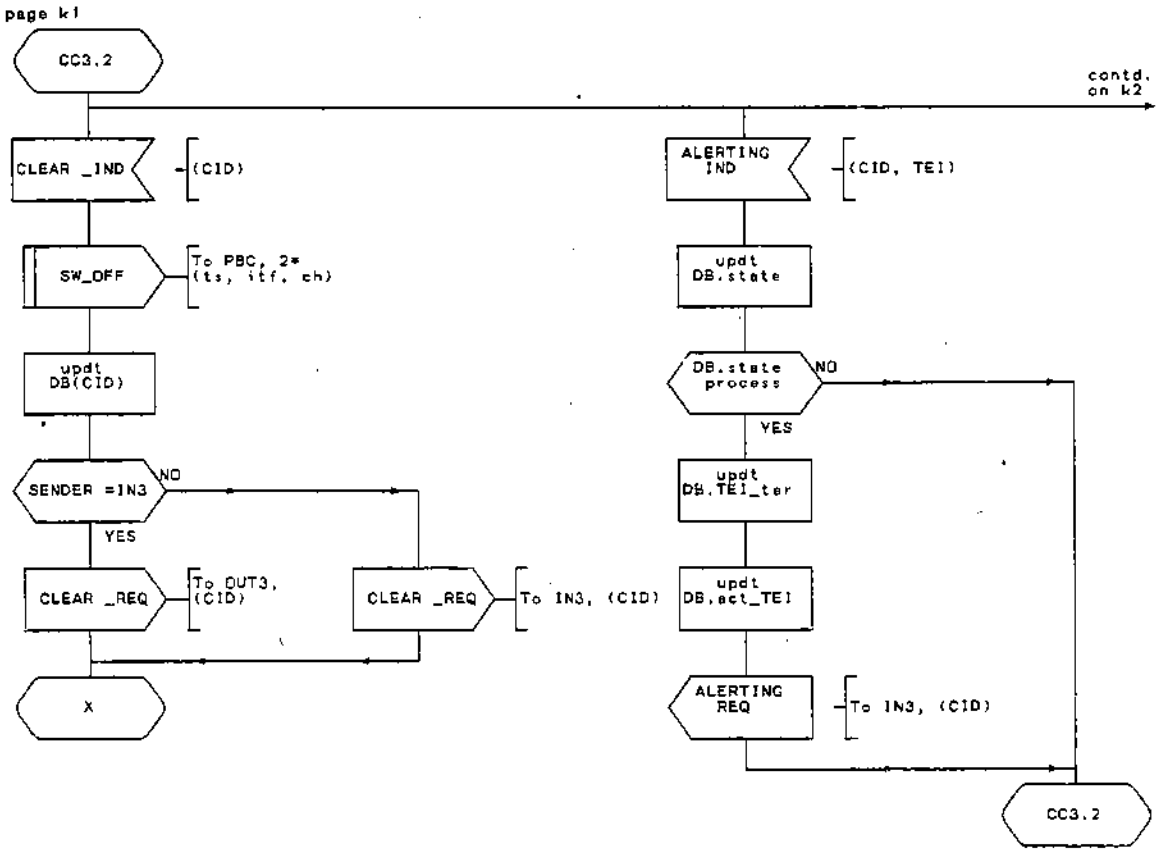

page j1
CC3.1
contd.
on j2
SETUP _IND
2e setup
CALL
PROCEED
IND
(CID, TEI)
ALERTING
IND
(CID, TEI)
as in CC3.0
same
procedure
updt
DB.state
updt
DB.state
CC3.1
updt
DB.TEI_ter
updt
DB.TEI_ter
updt
DB.act_TEI
updt
DB.act_TEI
SW_ON
To PBC, 2*
(ts, itf, ch)
SW_ON
To PBC, 2*
(ts, itf, ch)
updt DB.SW
updt DB.SW
map _tei()
TEI
TEI_ori
map _tei()
TEI
TEI_ori
j1$1
j1$2

page J1 contd.

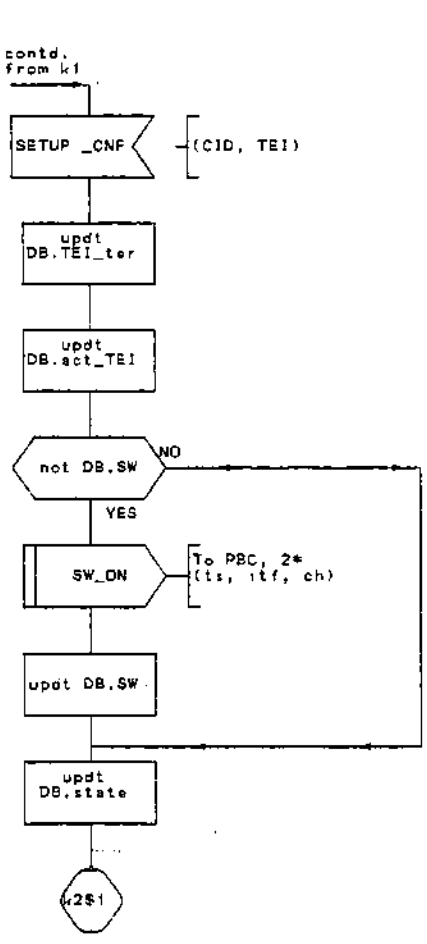


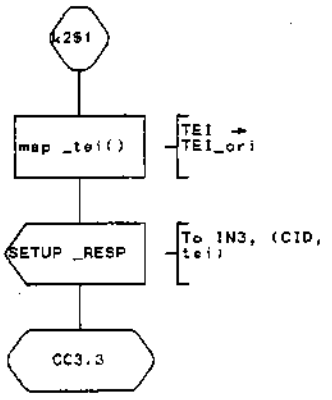
page j2


contd.
from j1
REJ_IND
(CID)
updt
DB(CID)
SENDER =
IN3
NO
YES
REJ_REQ
To OUT3,
(CID)
REJ_REQ
To IN3, (CID)
X
SETUP _CNF
(CID, TEI)
updt
DB.TEI_ter
updt
DB.act_TEI
SW_ON
To PBC, 2*
(ts, itf, ch)
updt DB.SW
updt
DB.state
map _tei()
TEI →
TEI_ori
j2$1
j2$1
SETUP _RESP
To IN3, (C
tei)
CC3.3

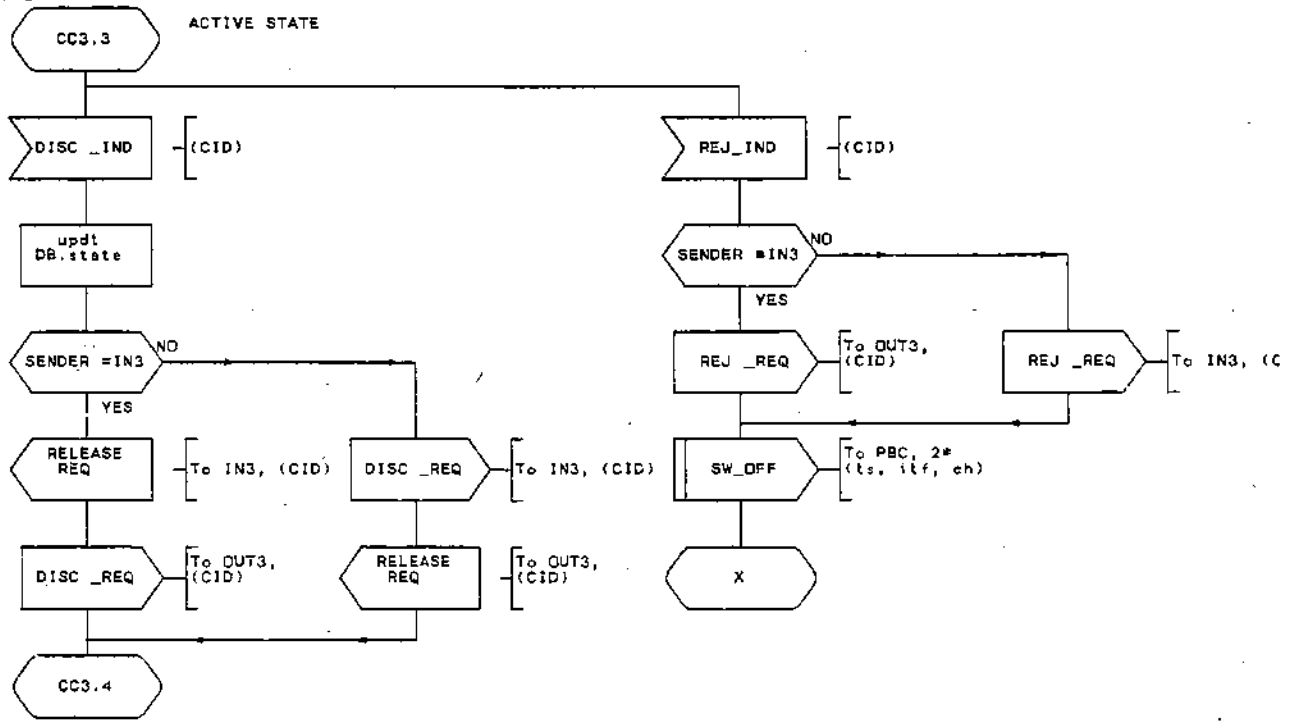

page k1
CC3.2
contd.
on k2
CLEAR _IND
(CID)
ALERTING
IND
(CID, TEI)
SW_OFF
To PBC, 2*
(ts, itf, ch)
updt
DB.state
updt
DB(CID)
DB.state
process
NO
YES
SENDER =IN3
NO
YES
updt
DB.TEI_ter
CLEAR _REQ
To OUT3,
(CID)
CLEAR _REQ
To IN3, (CID)
updt
DB.act_TEI
X
ALERTING
REQ
To IN3, (CID)
CC3.2

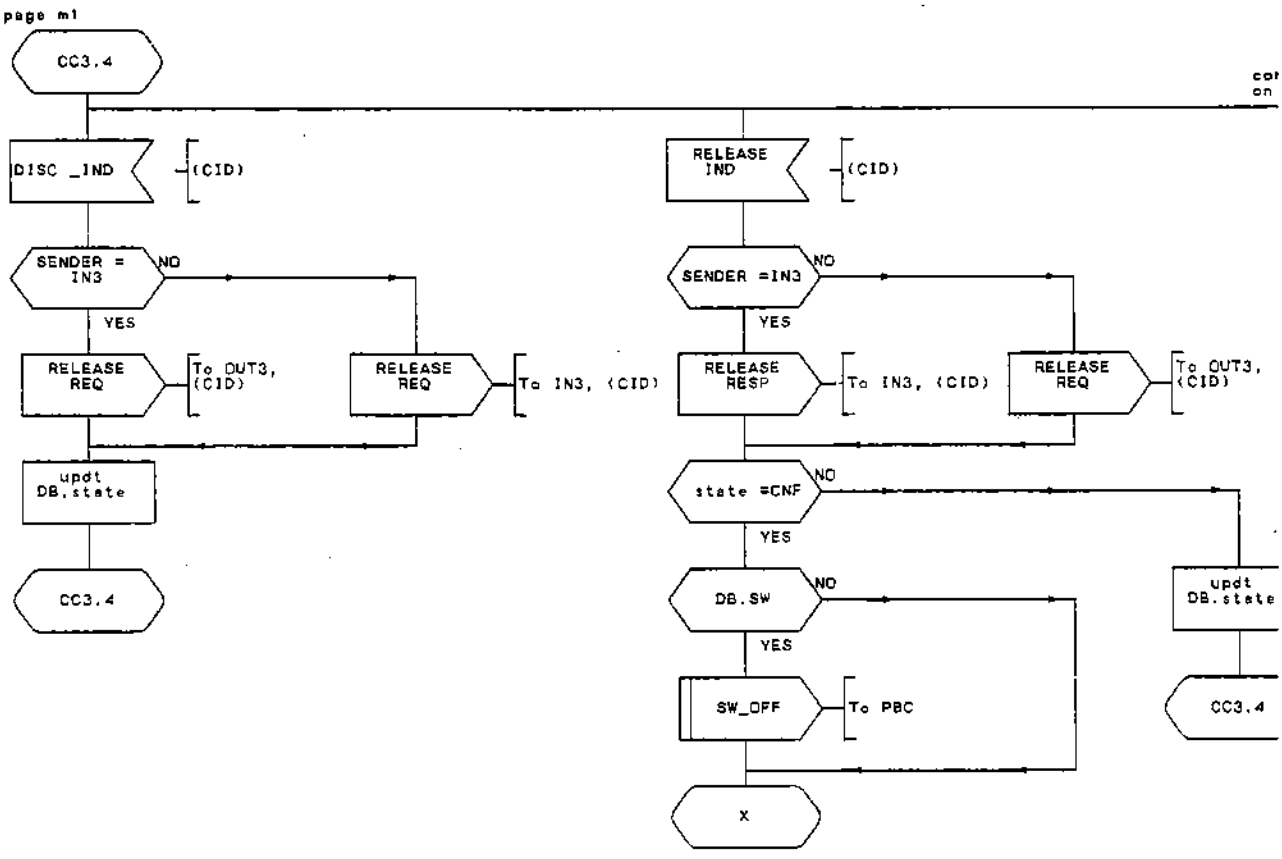
page k2


contd.
from k1
SETUP _CNF
(CID, TEI)
updt
DB.TEI_ter
updt
DB.act_TEI
not DB.SW
NO
YES
SW_ON
To PBC, 2*
(ts, itf, ch)
updt DB.SW
updt
DB.state
k2$1

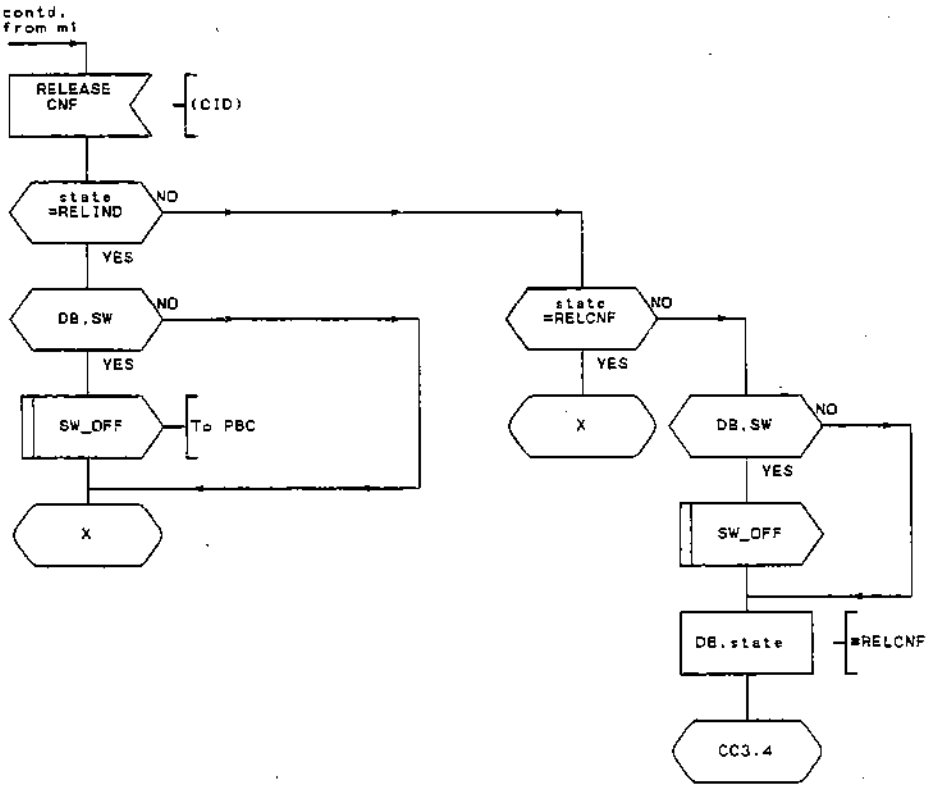

k2$1
map _tei()
TEI →
TEI_ori
SETUP _RESP
To IN3, (CID,
tei)
CC3.3

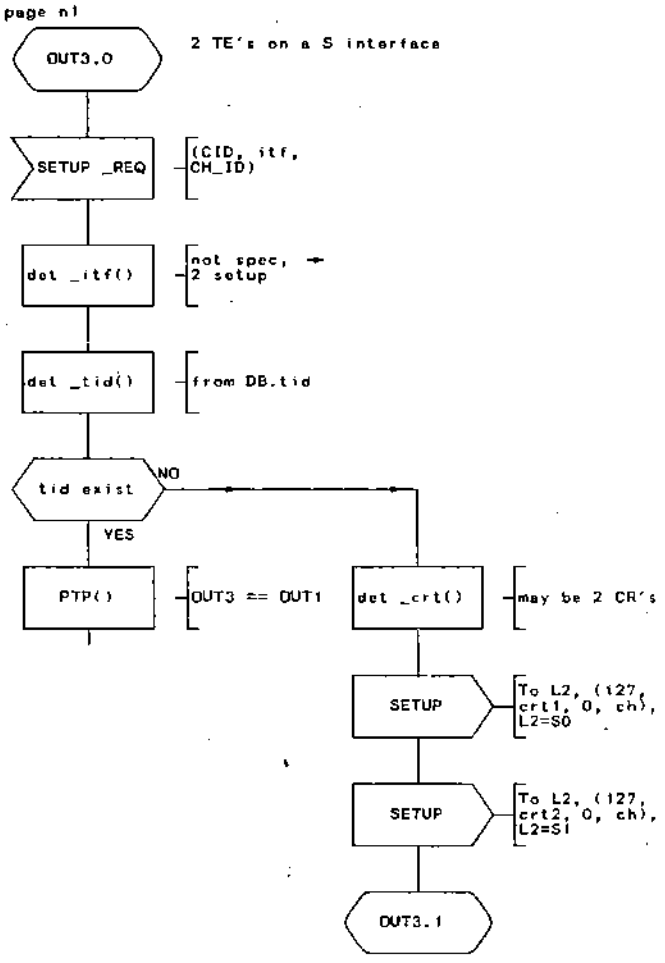
page 11

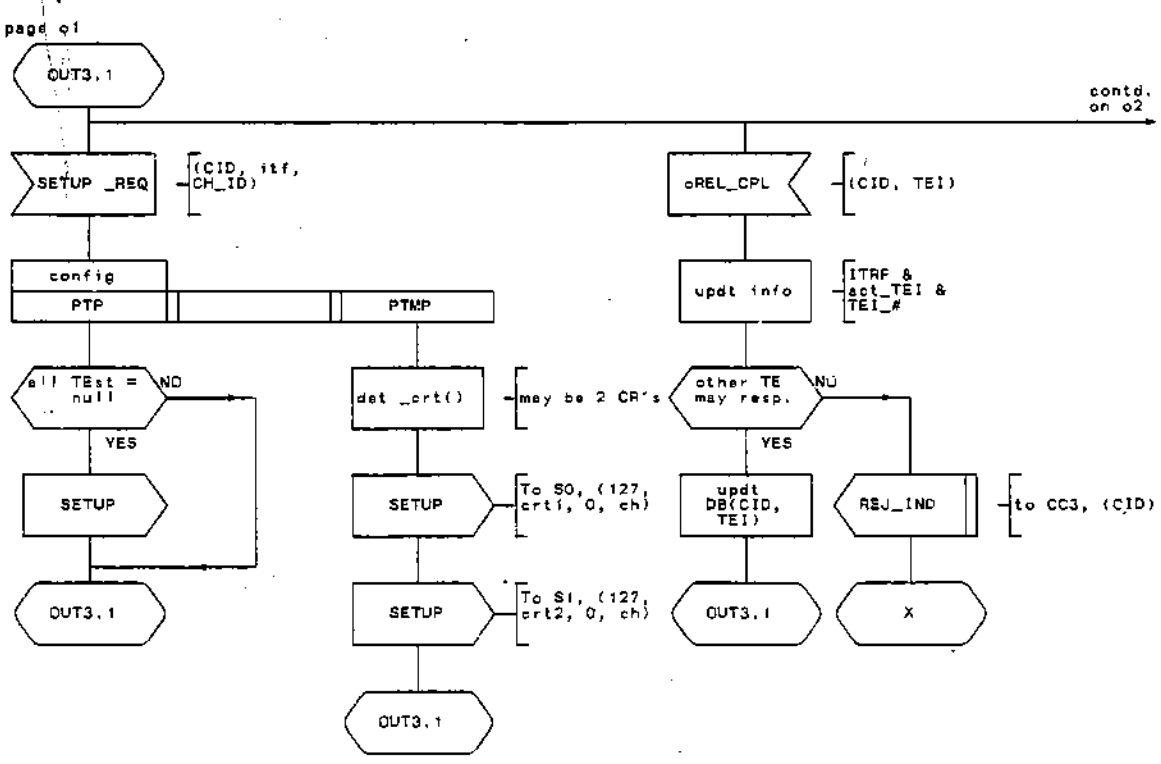

CC3.3
ACTIVE STATE
DISC _IND
(CID)
updt
DB.state
SENDER =IN3
NO
YES
RELEASE
REQ
To IN3, (CID)
DISC _REQ
To IN3, (CID)
DISC _REQ
To OUT3,
(CID)
RELEASE
REQ
To OUT3,
(CID)
CC3.4
REJ_IND
(CID)
SENDER =IN3
NO
YES
REJ _REQ
To OUT3,
(CID)
REJ _REQ
To IN3, (C
SW_OFF
To PBC, 2*
(ts, itf, ch)
X


page m1
CC3.4
DISC _IND
(CID)
SENDER = IN3
NO
YES
RELEASE REQ
To OUT3, (CID)
RELEASE REQ
To IN3, (CID)
updt DB.state
CC3.4
RELEASE IND
(CID)
SENDER =IN3
NO
YES
RELEASE RESP
To IN3, (CID)
RELEASE REQ
To OUT3, (CID)
state =CNF
NO
YES
DB.SW
NO
YES
SW_OFF
To PBC
X
updt DB.state
CC3.4

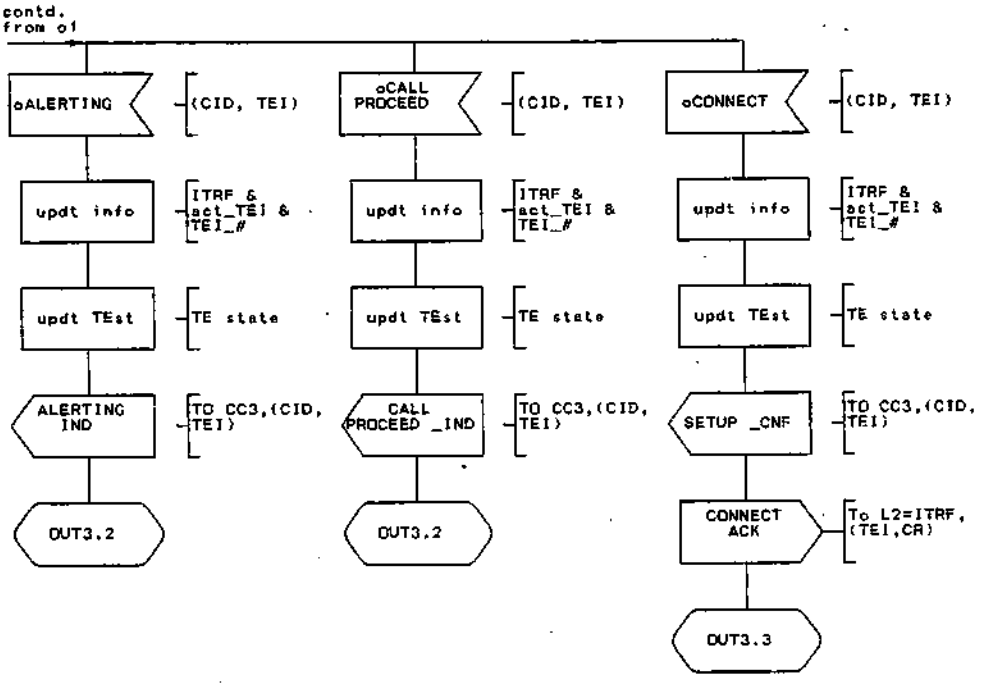
page m2

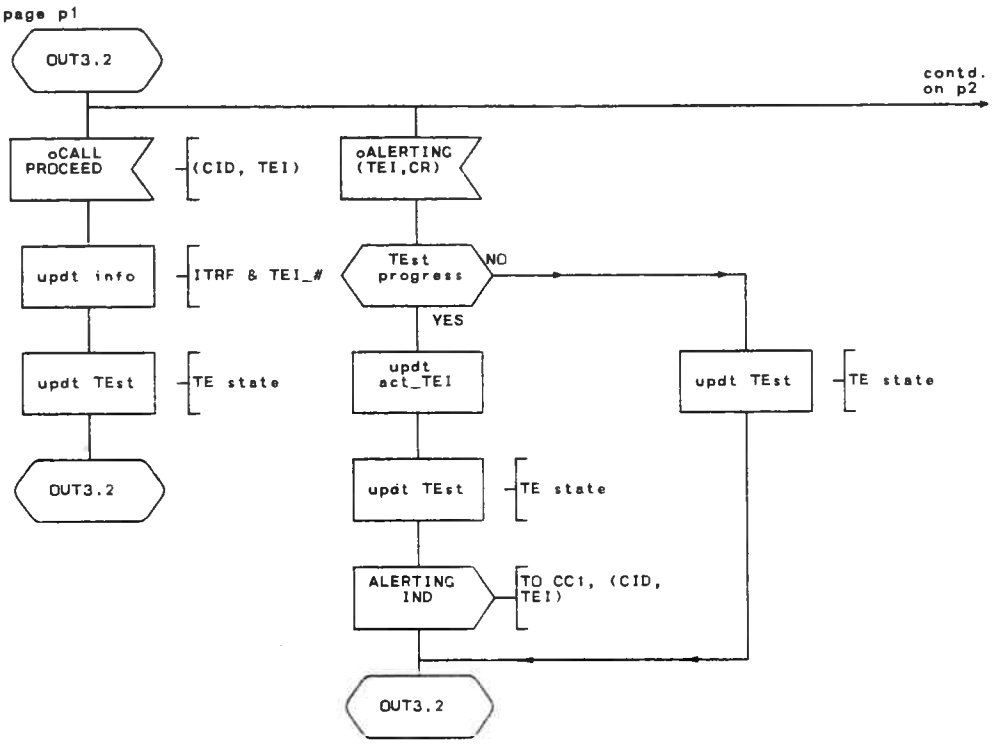

contd.
from m1
RELEASE
CNF
(CID)
state
=RELIND
NO
YES
DB.SW
NO
YES
SW_OFF
To PBC
X
state
=RELCNF
NO
YES
X
DB.SW
NO
YES
SW_OFF
DB.state
=RELCNF
CC3.4


page n1
OUT3.0
2 TE's on a S interface
SETUP _REQ
(CID, itf, CH_ID)
det _itf()
not spec, → 2 setup
det _tid()
from DB.tid
tid exist
NO
YES
PTP()
OUT3 == OUT1
det _crt()
may be 2 CR's
SETUP
To L2, (127, crt1, 0, ch), L2=S0
SETUP
To L2, (127, crt2, 0, ch), L2=S1
OUT3.1

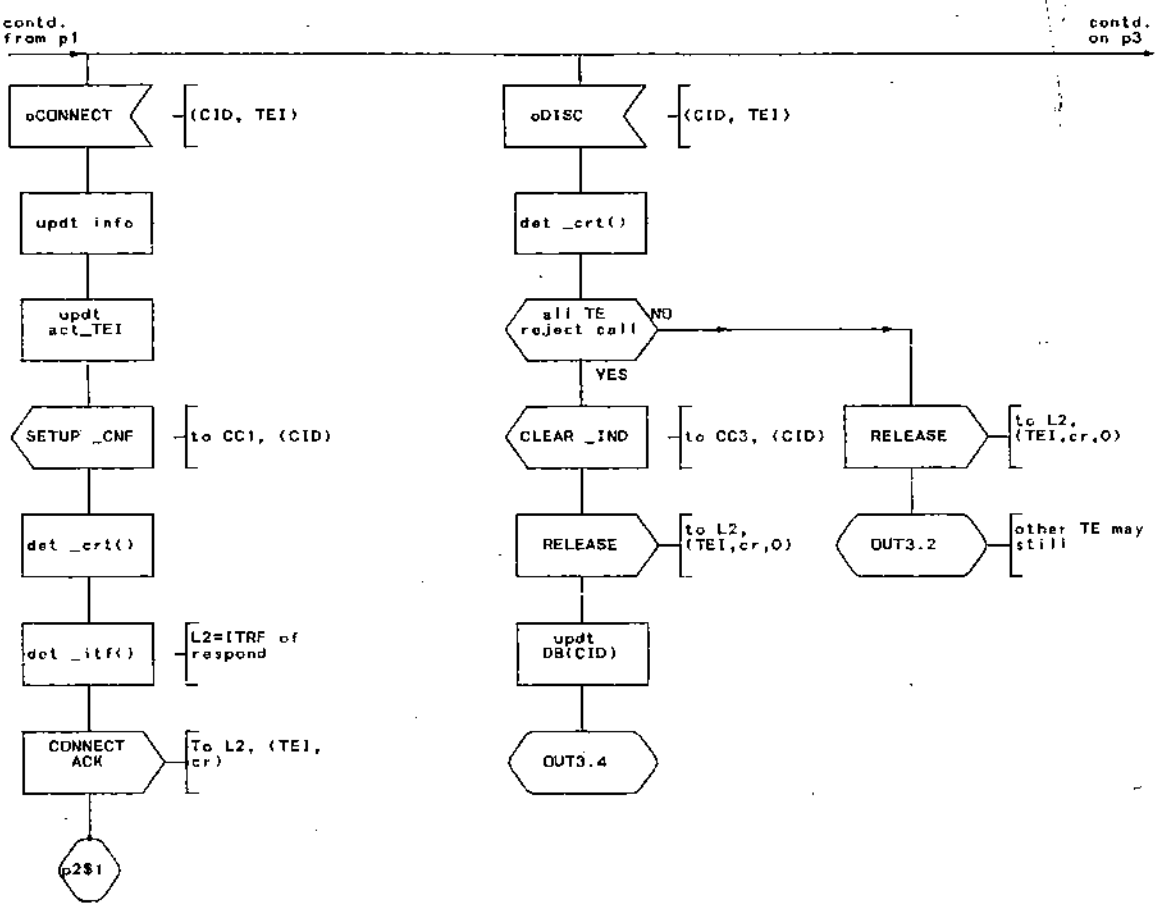

page o1
OUT3.1
contd. on o2
SETUP _REQ
(CID, itf, CH_ID)
oREL_CPL
(CID, TEI)
config
PTP
PTMP
updt info
ITRF & act_TEI & TEI_#
all TEst = null
NO
YES
det _crt()
may be 2 CR's
other TE may resp.
NO
YES
SETUP
To S0, (127, crt1, 0, ch)
updt DB(CID, TEI)
REJ_IND
to CC3, (CID)
SETUP
To S1, (127, crt2, 0, ch)
OUT3.1
OUT3.1
X
OUT3.1

page o2

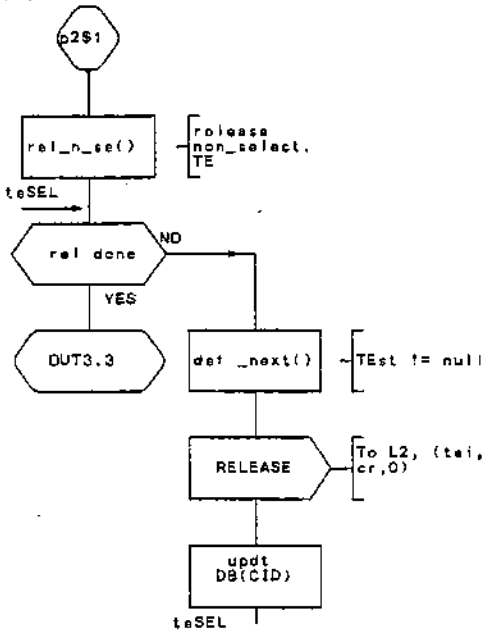

contd.
from o1
oALERTING
(CID, TEI)
updt info
ITRF & act_TEI & TEI_#
updt TEst
TE state
ALERTING IND
TO CC3,(CID, TEI)
OUT3.2
oCALL PROCEED
(CID, TEI)
updt info
ITRF & act_TEI & TEI_#
updt TEst
TE state
CALL PROCEED _IND
TO CC3,(CID, TEI)
OUT3.2
oCONNECT
(CID, TEI)
updt info
ITRF & act_TEI & TEI_#
updt TEst
TE state
SETUP _CNF
TO CC3,(CID, TEI)
CONNECT ACK
To L2=ITRF, (TEI,CR)
OUT3.3


page p1
OUT3.2
contd.
on p2
oCALL
PROCEED
(CID, TEI)
oALERTING
(TEI,CR)
updt info
ITRF & TEI_#
TEst
progress
NO
YES
updt TEst
TE state
updt
act_TEI
updt TEst
TE state
OUT3.2
updt TEst
TE state
ALERTING
IND
TO CC1, (CID,
TEI)
OUT3.2

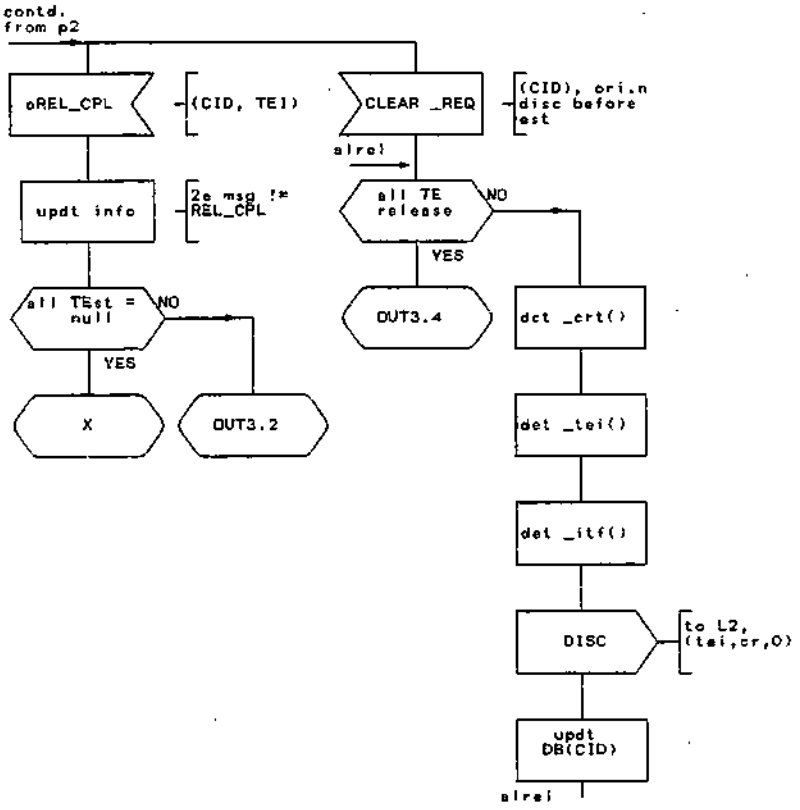
page p2

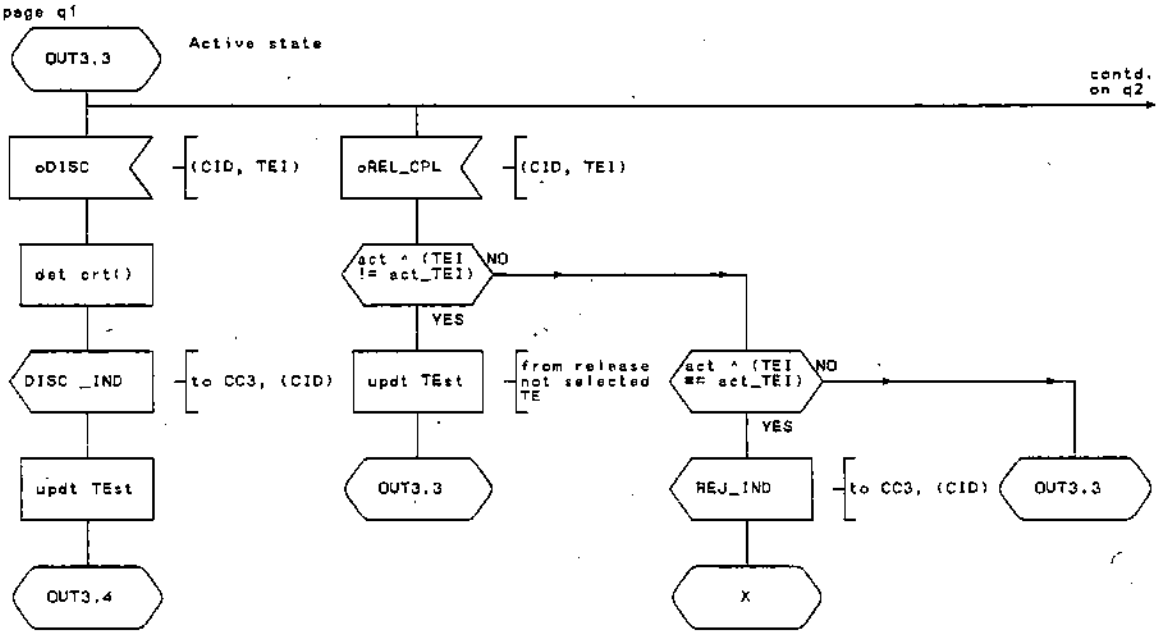

contd.
from p1
contd.
on p3
oCONNECT
(CID, TEI)
updt info
updt
act_TEI
SETUP _CNF
to CC1, (CID)
det _crt()
det _itf()
L2=ITRF of
respond
CONNECT
ACK
To L2, (TEI,
cr)
p2$1
oDISC
(CID, TEI)
det _crt()
all TE
reject call
NO
YES
CLEAR _IND
to CC3, (CID)
RELEASE
to L2,
(TEI,cr,0)
updt
DB(CID)
OUT3.4
RELEASE
to L2,
(TEI,cr,0)
OUT3.2
other TE may
still

page p2 contd.

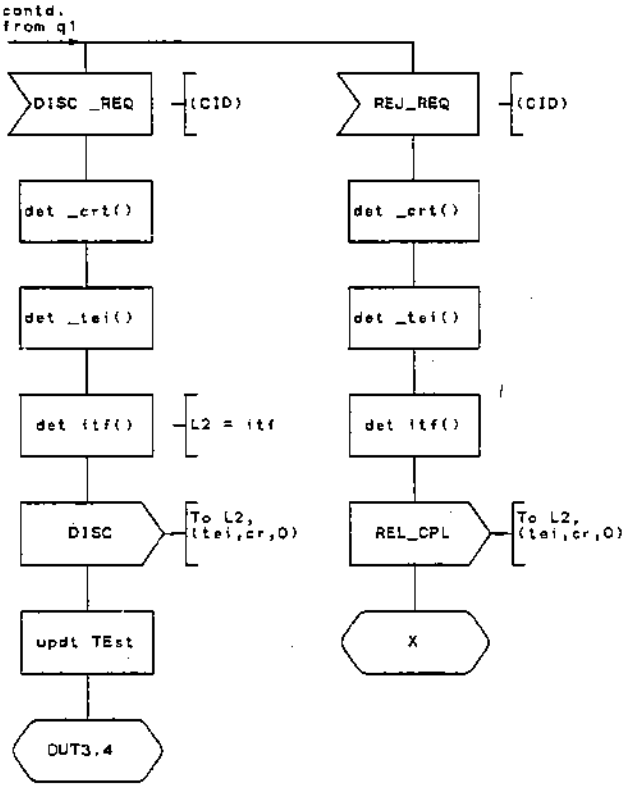


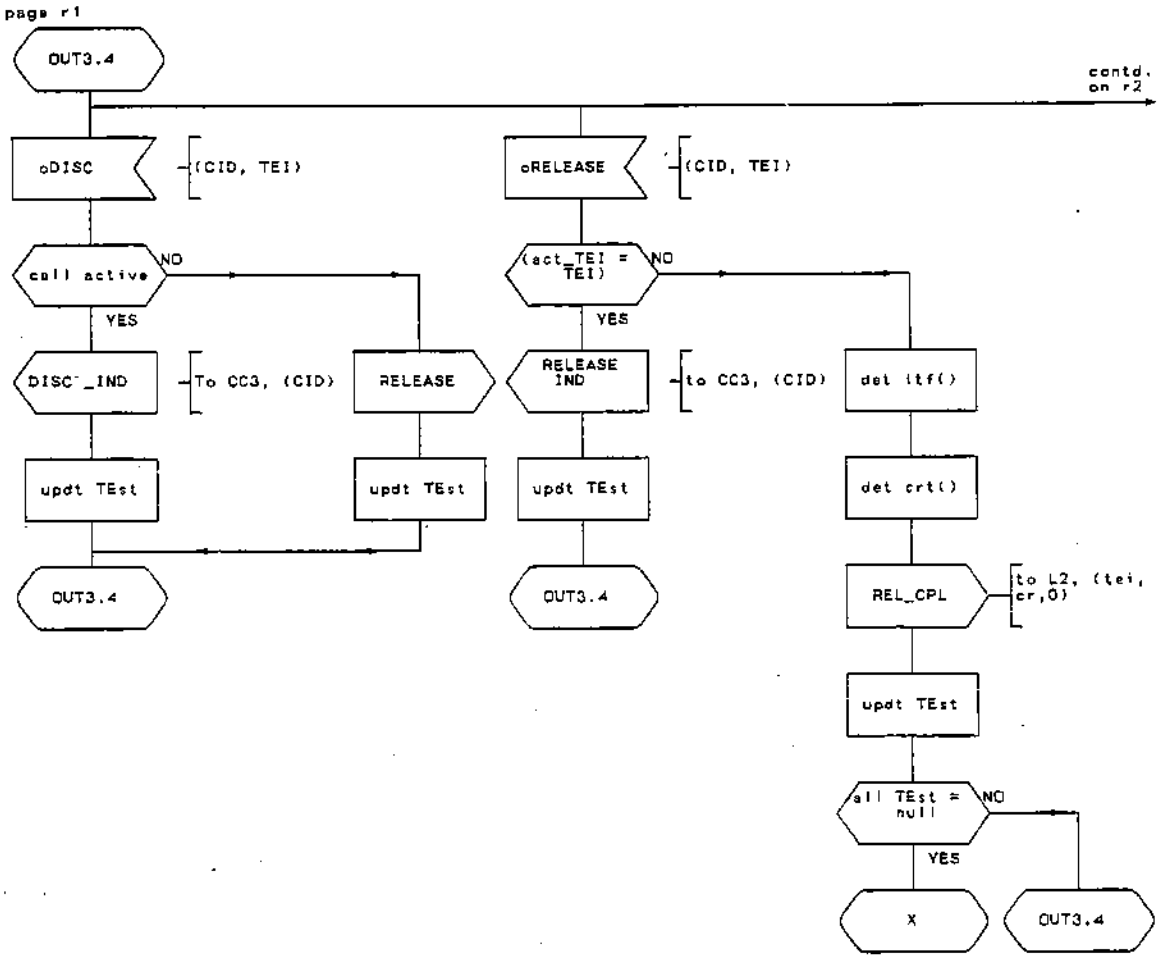
page p3


contd.
from p2
oREL_CPL
(CID, TEI)
CLEAR _REQ
(CID), ori.n
disc before
est
alrel
updt info
2e msg != REL_CPL
all TE release
NO
YES
all TEst = null
NO
YES
OUT3.4
dct _ert()
X
OUT3.2
det _tei()
det _itf()
DISC
to L2,
(tei,cr,0)
updt DB(CID)
alrel

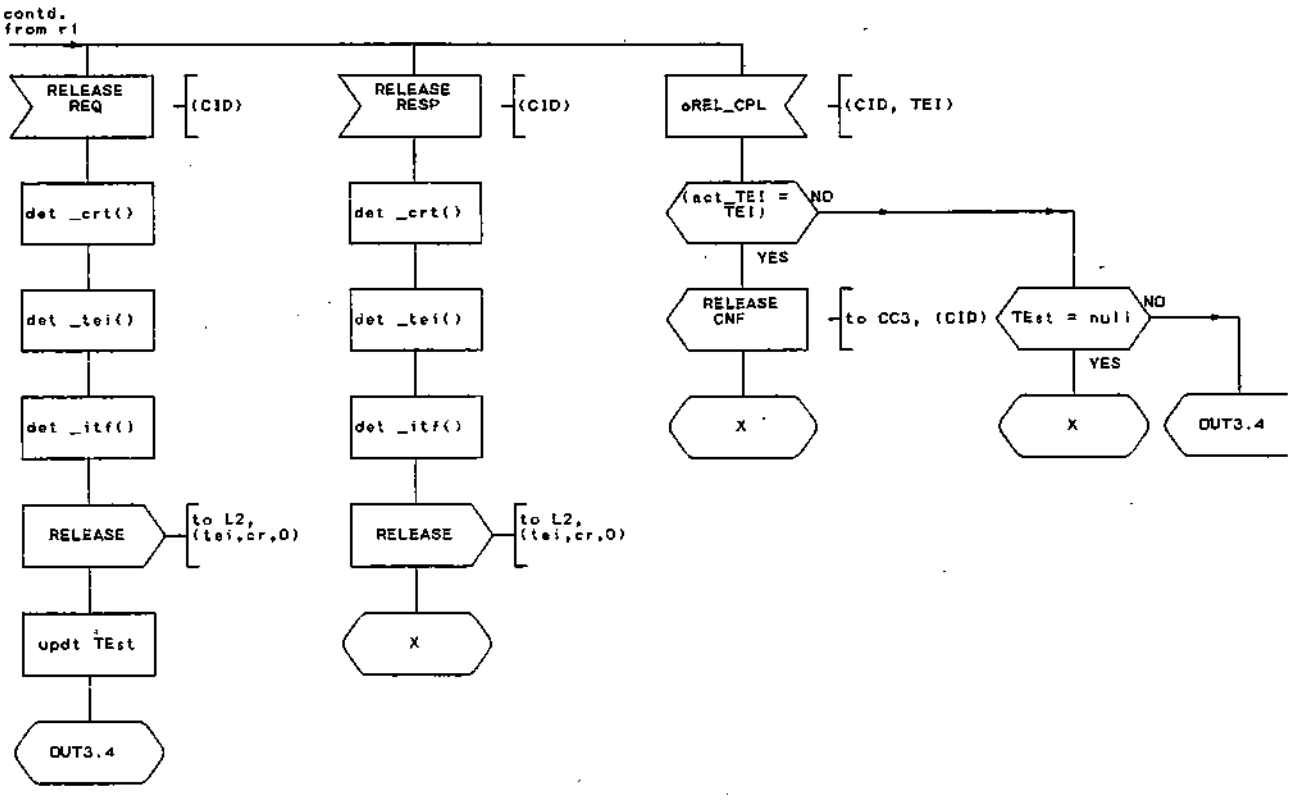

page q1
OUT3.3
Active state
contd.
on q2
oDISC
(CID, TEI)
oREL_CPL
(CID, TEI)
det crt()
act ^ (TEI
!= act_TEI)
NO
YES
DISC _IND
to CC3, (CID)
updt TEst
from release
not selected
TE
act ^ (TEI
== act_TEI)
NO
YES
updt TEst
OUT3.3
REJ_IND
to CC3, (CID)
OUT3.3
OUT3.4
X

page q2


contd.
from q1
DISC _REQ
(CID)
REJ_REQ
(CID)
det _crt()
det _crt()
det _tei()
det _tei()
det itf()
L2 = itf
det itf()
DISC
To L2,
(tei,cr,0)
REL_CPL
To L2,
(tei,cr,0)
updt TEst
X
OUT3.4


page r1
OUT3.4
contd. on r2
oDISC
(CID, TEI)
oRELEASE
(CID, TEI)
call active
NO
YES
(act_TEI = TEI)
NO
YES
DISC_IND
To CC3, (CID)
RELEASE
RELEASE IND
to CC3, (CID)
del itf()
updt TEst
updt TEst
updt TEst
del crt()
OUT3.4
OUT3.4
REL_CPL
to L2, (tei, cr,0)
updt TEst
all TEst = null
NO
YES
X
OUT3.4

page r2


contd.
from r1
RELEASE REQ
(CID)
det _crt()
det _tei()
det _itf()
RELEASE
to L2, (tei,cr,0)
updt TEst
OUT3.4
RELEASE RESP
(CID)
det _crt()
det _tei()
det _itf()
RELEASE
to L2, (tei,cr,0)
X
oREL_CPL
(CID, TEI)
(act_TEI = TEI)
NO
YES
RELEASE CNF
to CC3, (CID)
X
TEst = null
NO
YES
X
OUT3.4

APPENDIX D: SOURCE OF THE TEST PROGRAM (MODE 1)

```
/*
 *      Process arb:  selection of SETUP and other messages,
 *      transfer to IN or OUT process depends on the CR.FLAG;
 *      gen CID and db[CID], creat IN and CC process for the
 *      involved call.
 *
 *      ports:
 *              RC              O
 *              cdb[CID].oinfo.pori   arbport
 *              cdb[CID].tinfo.poro   arbport
 */


#include        "b:msg.h"
#include        "b:def.h"
#include        "b:struct.h"
#include        "b:mac.h"
#include        "b:signals.h"

#define         RC      O       /* from environment             */
                         /* cdb[CID].oinfo.pori to IN process          */
                         /* cdb[CID].tinfo.poro to OUT process  */

struct  CINFO       cdb[MAX_CID];
struct  resources   respool;
struct  status      NT_STA;

int     SELF,ENV;

PROCESS(arb)
        int     CID;
        int     i,TEI,CR,CH_ID,xnr,nwipid,cpid;
        int     FLAG,MOD_FL,n,mode_flag;
        char    var[16];

        *((struct CINFO **)0x110) = cdb;
        *((struct resources **)0x114) = &respool;
        *((struct status **)0x118) = &NT_STA;

        mode_flag = 1;
        SELF = sys(14,"me");
        ENV = sys(14,"/nt");
        NT_STA.arbpid = SELF;

/******    initialization of tid and interface of the assigned tei  ****/

        for (i=1 ; i < 5; i++ ) {
                NT_STA.val_tid[i]=(100+i);
                if (i < 3)
                        NT_STA.val_inf[i]=0;
                else
                        NT_STA.val_inf[i]=1;
        };
STATE(arb_1)
        INPUT SETUP: PARM(&mbuf, "iiiiii",&TEI,&CR,&FLAG,&CH_ID,&MOD_FL,&xnr);
                goto LAR_1;

        INPUT CONNECT_ACK: PARM(&mbuf,"iii",&TEI,&CR,&FLAG);
                goto LAR_2;

        INPUT CALL_PROCEED: PARM(&mbuf,"iiii",&TEI,&CR,&FLAG,&CH_ID);
                goto LAR_3;

        INPUT ALERTING: PARM(&mbuf,"iiii",&TEI,&CR,&FLAG,&CH_ID);
                goto LAR_4;

        INPUT CONNECT: PARM(&mbuf,"iiii",&TEI,&CR,&FLAG,&CH_ID);
                goto LAR_5;

        INPUT DISC: PARM(&mbuf,"iii",&TEI,&CR,&FLAG);
                goto LAR_6;

        INPUT RELEASE: PARM(&mbuf,"iii",&TEI,&CR,&FLAG);
                goto LAR_7;

        INPUT REL_CPL: PARM(&mbuf,"iii",&TEI,&CR.&FLAG);
                goto LAR_8;
```

```
        INPUT CMD:  PARM(&mbuf, "si",var,&n);
                    goto LAR_9;

LAR_1:              if( MOD_FL != 1 ) {
                            error("init mode first");
                            NEXTSTATE arb_1;
                    }
                    if( FLAG != 0 ) {
                            error("CR_FLAG of msg err");
                            NEXTSTATE arb_1;
                    }
                    /* new call */
                    if( recog(SENDER,TEI,CR)==0 ) {

                            CID = CID_alloc();
                            printf("CID for new call: %d0,CID);
                            setrecog(SENDER,TEI,CR,CID);

                            /* CREAT IN1 */
                            if( (nwipid=creat("in1",CID)) < 0 ) {
                                    NEXTSTATE arb_1;
                            } else  {
                                    reset(CID);
                                    iniODB(CID,SENDER,TEI,CR,nwipid);
                            }

                            /* connect arb in1          */

                            if(connect(SELF,cdb[CID].oinfo.pori,NT_STA.inpid[CID],3)<0){
                                    NEXTSTATE arb_1;
                            }

                            /* connect in1 L2 (environment)  */
                            if( connect(NT_STA.inpid[CID],14,ENV,cdb[CID].oinfo.pori+5)<0){
                                    NEXTSTATE arb_1;
                            }
                            /* CREAT CC1 */
                            if( (cpid=creat("cc1",CID)) < 0) {
                                    NEXTSTATE arb_1;
                            } else {
                                    cdb[CID].CCPID = cpid;
                                    NT_STA.ccpid[CID] = cpid;
                            }
                            /*  connect in1 cc1          */
                            if( connect(NT_STA.inpid[CID],4,NT_STA.ccpid[CID],5)<0){
                                    NEXTSTATE arb_1;
                            }

                            OUTPUT(&mbuf, cdb[CID].oinfo.pori, iSETUP,"iii", CID, xnr, CH_ID);
                            NEXTSTATE arb_1;
                    }
                    else {
                            error("SETUP twice for same call ");
                            NEXTSTATE arb_1;
                    }

/* LAR_2        */

LAR_2:              if( recog(SENDER,TEI,CR) == 0 ) {
                            error("no CID, message not expected ");
                            NEXTSTATE arb_1;
                    }
                            /*  CID exist, SENDER,TEI and CR unique  */

                    if( FLAG == 0 ) {
                            CID = recog(SENDER,TEI,CR);
                            OUTPUT(&mbuf,cdb[CID].oinfo.pori, iCONNECT_ACK,"i",CID);
                    } else
                            error("wrong FLAG/msg combination ");
                    NEXTSTATE arb_1;

/* LAR_3        */

LAR_3:              if(FLAG == 1) {
                            if( (CID=det_CID(SENDER,TEI,CR)) < 0)    {
                                    error("can not id msg ");
                            }
                            if( CH_ID != cdb[CID].tinfo.B_CH_T )
```

```
                        error("CH_ID wrongly specified");
                    else
                        OUTPUT(&mbuf,cdb[CID].tinfo.poro,oCALL_PROCEED,"ii",CID,TEI
                } else
                    error("wrong FLAG/msg combination ");
                NEXTSTATE arb_1;

/* LAR_4        */
LAR_4:          if( FLAG == 1)  {
                    if( (CID=det_CID(SENDER,TEI,CR)) < 0)   {
                        error("can not id msg ");
                    }
                    if( CH_ID != cdb[CID].tinfo.B_CH_T )
                        error("CH_ID wrongly specified");
                    else
                        OUTPUT(&mbuf,cdb[CID].tinfo.poro,oALERTING,"ii",CID,TEI);
                } else  {
                    error("wrong FLAG/msg combination ");
                }
                NEXTSTATE arb_1;

/* LAR_5        */

LAR_5:          if( FLAG == 1)  {
                    if( (CID=det_CID(SENDER,TEI,CR)) < 0)   {
                        error("can not id msg ");
                    }
                    if( CH_ID != cdb[CID].tinfo.B_CH_T )
                        error("CH_ID wrongly specified");
                    else
                        OUTPUT(&mbuf,cdb[CID].tinfo.poro,oCONNECT,"ii",CID,TEI);
                } else  {
                    error("wrong FLAG/msg combination ");
                }
                NEXTSTATE arb_1;

/* LAR_6        */
LAR_6:          if( recog(SENDER,TEI,CR)==0)    {
                    error("wrong msg seq ");
                } else {
                    CID = recog(SENDER,TEI,CR);
                    if( FLAG==0) {
                        OUTPUT(&mbuf, cdb[CID].oinfo.pori,iDISC,"i",CID);
                    } else {
                        OUTPUT(&mbuf, cdb[CID].tinfo.poro,oDISC,"ii",CID,TEI);
                    }
                };
                NEXTSTATE arb_1;

/* LAR_7        */

LAR_7:          if( recog(SENDER,TEI,CR)==0)    {
                    error("wrong msg seq ");
                } else {
                    CID = recog(SENDER,TEI,CR);
                    if( FLAG==0)    {
                        OUTPUT(&mbuf, cdb[CID].oinfo.pori,iRELEASE,"i",CID);
                    } else {
                        OUTPUT(&mbuf, cdb[CID].tinfo.poro,oRELEASE,"ii",CID,TEI);
                    }
                }
                NEXTSTATE arb_1;

/* LAR_8        */

LAR_8:          if( FLAG == 1)  {
                    if( (CID=det_CID(SENDER,TEI,CR)) > 0)   {
                        OUTPUT(&mbuf,cdb[CID].tinfo.poro,oREL_CPL,"ii",CID,TEI);
                    } else {
                        error("can not id. msg ");
                    }
                } else  {
                    if( recog(SENDER,TEI,CR)==0)    {
                        error("wrong msg seq ");
                    } else {
                        CID = recog(SENDER,TEI,CR);
```

```
                                        OUTPUT(&mbuf, cdb[CID].oinfo.pori,iREL_CPL,"i",CID);
                                }
                        }
                        NEXTSTATE arb_1;

/* LAR_9                */
LAR_9:                  if( strcmp(var,"mode")==0 ) {
                                switch(n) {
                                        case 1: mode_flag = 1;
                                                break;
                                        case 2:
                                        case 3: error("not installed yet");
                                                break;
                                        default:error("invalid mode");
                                                break;
                                }
                        } else  {
                                error("only cmd
                        }
                        NEXTSTATE arb_1;

ENDSTATE

ENDPROCESS

/*************************************************************/

CID_alloc()
{
        int extcid = 0;

        while( respool.cid[++extcid] != FREE )
                if( extcid > MAX_CID ) error("no free cid");
        respool.cid[extcid] = OCC;
        return(extcid);
}

reset(call_id)
int call_id;
{
        cdb[call_id].oinfo.ts_O = 0;
        cdb[call_id].oinfo.pori = 0;
        cdb[call_id].oinfo.TEI_O = 0;
        cdb[call_id].oinfo.CR_O = 0;
        cdb[call_id].oinfo.INPID = 0;
        cdb[call_id].oinfo.B_CH_O = 0;

        cdb[call_id].sw = FALSE;
        cdb[call_id].state = ST_NULL;
        cdb[call_id].CCPID = 0;

        cdb[call_id].tinfo.ts_T = 0;
        cdb[call_id].tinfo.CR_T = 0;
        cdb[call_id].tinfo.TEI = 0;
        cdb[call_id].tinfo.poro = 0;
        cdb[call_id].tinfo.OUTPID = 0;
        cdb[call_id].tinfo.B_CH_T = 0;
}

iniODB( call_id, send, endpt, clrf, pid )
int     call_id;
int     send;
int     endpt;
int     clrf;
int     pid;
{
        int port;

        cdb[call_id].oinfo.OBIDS = send;
        cdb[call_id].oinfo.OBIDR = send;
        cdb[call_id].oinfo.TEI_O = endpt;
        cdb[call_id].oinfo.CR_O  = clrf;

        port = port_alloc();
        cdb[call_id].oinfo.pori = port;

        cdb[call_id].tinfo.poro = (port+MAX_PORT/2);
```

```
        cdb[call_id].cinfo.INPID = pid;
        NT_STA.inpid[call_id] = pid;
}

port_alloc()
{
        static int ARBPORT = 1;
        int i = 0;

        while( respool.arbport[ARBPORT] != FREE ) {
                ARBPORT++;
                if( ARBPORT >= 4 ) ARBPORT = 1;
                if( ++i > MAX_PORT/2 ) error("no free port");
        }
        respool.arbport[ARBPORT] = OCC;

        return(ARBPORT);
}

creat(prty,cid)
char    prty[];         /* process type IN1, OUT1, CC1  */
{
        int OFFSPRING;
        char    name[16];
        char    number[8];

        strcpy(name,prty);
        sprintf(number,"_%d",cid);
        strcat(name,number);
        if( (OFFSPRING=sys(6,name,prty,20,0,0)) < 0) {
                error("can't create %s",prty);
                return(-1);
        }
        sys(7,OFFSPRING);
        return(OFFSPRING);
}

connect(pro1,port1,pro2,port2)
int     pro1,port1,pro2,port2;
{
        if( sys(3,pro1,port1,pro2,port2) ) {
                printf("connect error %d[%d] %d[%d]0,pro1,port1,pro2,port2);
                return(-1);
        }
        return(1);
}

det_CID( send, endpt, clrf)
int send,endpt,clrf;
{
        if( recog(send,endpt,clrf) != 0)
                return( recog(send,endpt,clrf) );
        if( NT_STA.ter_cr[clrf] !=0 )
                return( setrecog(send,endpt,clrf,NT_STA.ter_cr[clrf]) );
        error("CID can not be recog.");
        return(-1);
}

recog(send,tei,cr)
int send,tei,cr;
{
        int i;

        for(i=0;i<64;i++) {
                if( NT_STA.val_send[i]==send &&
                    NT_STA.val_tei[i]==tei &&
                    NT_STA.val_cr[i]==cr )
                        return( NT_STA.val_cid[i] );
        }
        return(0);
}

setrecog(send,tei,cr,cid)
int send,tei,cr,cid;
{
        register int i;

        for(i=0;i<64;i++) {
```

```
                if( NT_STA.val_send[i]==send &&
                    NT_STA.val_tei[i]==tei &&
                    NT_STA.val_cr[i]==cr )
                        break;
        }
        if( i < 64 ) {          /* exist */
                if( cid == 0 ) {
                        NT_STA.val_send[i] = 0;
                        NT_STA.val_tei[i] = 0;
                        NT_STA.val_cr[i] = 0;
                        return(cid);
                }
                else
                        NT_STA.val_cid[i] = cid;
        }
        if( cid == 0 )                  /* does not exist */
                return(cid);
        for(i=0;i<64;i++) {
                if( NT_STA.val_send[i]==0 &&
                    NT_STA.val_tei[i]==0 &&
                    NT_STA.val_cr[i]==0 )
                        break;
        }
        if( i < 64 ) {
                NT_STA.val_send[i] = send;
                NT_STA.val_tei[i] = tei;
                NT_STA.val_cr[i] = cr;
                NT_STA.val_cid[i] = cid;
                return(cid);
        }
        else
                error("NT_STA.val overflow");
}

error(fmt,x1,x2,x3,x4)
char *fmt;
int x1,x2,x3,x4;
{
        printf("arb error:");
        printf(fmt,x1,x2,x3,x4);
        printf("0);
/*      sys(8,0);       */
}
```

```
/*
 *      Process in:
 *
 *
 *
 *
 *
 *
 */

#include        "b:msg.h"
#include        "b:def.h"
#include        "b:struct.h"
#include        "b:mac.h"
#include        "b:signals.h"

#define         toCC    4
#define         toL2    14

struct  CINFO   *cdb;           /* [MAX_CID] */
struct  resources *respool;
struct  status  *NT_STA;

PROCESS(IN1)
        int     CID,xnr,CH_ID,Itf,tei,cr;
        int     SELF;
        cdb = *((struct CINFO **)0x110);
        respool = *((struct resources **)0x114);
        NT_STA = *((struct status **)0x118);

        SELF = sys(14,"me");

STATE(IN1_1)
        INPUT iSETUP: PARM(&mbuf,"iii",&CID,&xnr,&CH_ID);
                goto LIN1_1;

        INPUT iDISC: PARM(&mbuf,"i",&CID);
                goto LIN1_2;

        INPUT REJ_REQ: PARM(&mbuf,"i",&CID);
                goto LIN1_3;

        INPUT CLEAR_REQ: PARM(&mbuf,"i",&CID);
                goto LIN1_4;

        INPUT SET_RESP: PARM(&mbuf,"i",&CID);
                goto LIN1_5;

        INPUT CALL_P_REQ: PARM(&mbuf,"ii",&CID, &CH_ID);
                goto LIN1_6;

        INPUT ALERT_REQ: PARM(&mbuf,"i",&CID);
                goto LIN1_7;

LIN1_1:                                                         /* 1507 */
        if ((Itf=interface(xnr)) >= 0) {
                OUTPUT( &mbuf, toCC, SET_IND, "iiii", CID, Itf, CH_ID, xnr);
                NEXTSTATE IN1_1;
        } else {
                error("itf of tid %d not found in IN1_1",xnr);
                syskill (cdb[CID].CCPID, CID, 'C');
                syskill (SELF,CID,'I');
        };
LIN1_2:
        OUTPUT( &mbuf, toCC, CLEAR_IND, "i", CID);
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, RELEASE, "iii", tei, cr, 1);
        SET (T1,20);
        NEXTSTATE IN1_3;
LIN1_3:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, REL_CPL, "iii", tei, cr, 1);

        if( may_cl(CID,'O') ) {
                syskill(SELF,CID,'I');
        }
        else {
```

```
                error("LIN1_3: kill(SELF) not allowed");
                NEXTSTATE IN1_1;
        }
LIN1_4:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, DISC, "iii", tei, cr, 1);
        SET(T5, 20);
        NEXTSTATE IN1_3;

LIN1_5:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        CH_ID = cdb[CID].oinfo.B_CH_O;
        OUTPUT( &mbuf, toL2, CONNECT, "iiii", tei, cr, 1, CH_ID);
        NEXTSTATE IN1_2;

LIN1_6:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, CALL_PROCEED, "iiii", tei, cr, 1, CH_ID);
        NEXTSTATE IN1_1;

LIN1_7:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        CH_ID = cdb[CID].oinfo.B_CH_O;
        OUTPUT( &mbuf, toL2, ALERTING, "iiii", tei, cr, 1, CH_ID);
        NEXTSTATE IN1_1;

ENDSTATE

STATE(IN1_2)

        INPUT iCONNECT_ACK: PARM(&mbuf,"i",&CID);
                goto LIN1_21;

        INPUT iDISC: PARM(&mbuf,"i",&CID);
                goto LIN1_22;

        INPUT iREL_CPL: PARM(&mbuf,"i",&CID);
                goto LIN1_23;

        INPUT DISC_REQ: PARM(&mbuf,"i",&CID);
                goto LIN1_24;

        INPUT REJ_REQ: PARM(&mbuf,"i",&CID);
                goto LIN1_25;

LIN1_21:
        if( cdb[CID].state != ST_ACTIVE)
                error("CCstate not conseq. updated. ");
        NEXTSTATE IN1_2;
LIN1_22:
        OUTPUT( &mbuf, toCC, DISC_IND, "i", CID);
        NEXTSTATE IN1_3;
LIN1_23:                                                    /* 1507 */
        OUTPUT( &mbuf, toCC, REJ_IND, "i", CID);
        syskill(SELF,CID,'I');
LIN1_24:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, DISC, "iii", tei, cr, 1);
        SET(T5, 10);
        NEXTSTATE IN1_3;
LIN1_25:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, REL_CPL, "iii", tei, cr, 1);
        if( may_cl(CID,'O') ) {
                syskill(SELF,CID,'I');
        }
        else {
                error("LIN1_25: kill(SELF) not allowed");
                NEXTSTATE IN1_2;
        }

ENDSTATE
```

```
STATE(IN1_3)

        INPUT iDISC: PARM(&mbuf, "i", &CID);
                goto LIN1_31;
        INPUT iRELEASE: PARM(&mbuf, "i", &CID);
                goto LIN1_32;
        INPUT iREL_CPL: PARM(&mbuf, "i", &CID);
                goto LIN1_33;
        INPUT REL_REQ: PARM(&mbuf, "i", &CID);
                goto LIN1_34;
        INPUT REL_RESP: PARM(&mbuf, "i", &CID);
                goto LIN1_35;
        INPUT T1:
                goto LIN1_36;
        INPUT T2:
                goto LIN1_37;
        INPUT T5:
                goto LIN1_38;
LIN1_31:
        RESET(T5);
        if( cdb[CID].state == ST_ACTIVE )
                OUTPUT( &mbuf, toCC, DISC_IND, "i", CID);
        else {
                OUTPUT( &mbuf, toCC, CLEAR_IND,"i", CID);
                tei = det_Otei(CID);
                cr = det_Ocr(CID);
                OUTPUT( &mbuf, toL2, RELEASE, "iii", tei, cr, 1);
                SET(T1, 20);
        }
        NEXTSTATE IN1_3;
LIN1_32:
        RESET(T5);
        if(( cdb[CID].state >= ST_ACTIVE ) && NT_STA->ccpid[CID] !=0 ){
                OUTPUT( &mbuf, toCC, REL_IND, "i", CID);
                NEXTSTATE IN1_3;
        }
        else {
                tei = det_Otei(CID);
                cr = det_Ocr(CID);
                OUTPUT( &mbuf, toL2, REL_CPL, "iii", tei, cr, 1);
                if( may_cl(CID,'O') ) {
                        syskill(SELF,CID,'I');
                }
                else {
                        error("LIN1_32: kill(SELF) not allowed");
                        NEXTSTATE IN1_3;
                }
        }


LIN1_33:
        RESET (T1);
        RESET (T2);
        if( cdb[CID].sw )
                OUTPUT( &mbuf, toCC, REL_CNF, "i", CID);
        else
                OUTPUT( &mbuf, toCC, CLEAR_IND,"i",CID);
        syskill(SELF,CID,'I');

LIN1_34:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, RELEASE, "iii", tei, cr, 1);
        SET(T1, 10);
        NEXTSTATE IN1_3;

LIN1_35:
        tei = det_Otei(CID);
        cr = det_Ocr(CID);
        OUTPUT( &mbuf, toL2, REL_CPL, "iii", tei, cr, 1);
        syskill(SELF,CID,'I');
LIN1_36:
        /*      1e timeout                                              */
            OUTPUT (&mbuf, toL2,RELEASE, "iii", tei,cr,1);
            SET(T2, 10);
            NEXTSTATE IN1_3;
LIN1_37:
        /*              2e timeout                                      */
```

```
        if( NT_STA->ccpid[CID]!=0 )
                syskill( NT_STA->ccpid[CID],CID,'C');
        syskill(SELF,CID,'I');
LIN1_38:
        /*              expired of T5 (T305)                            */
            OUTPUT (&mbuf, toL2,RELEASE, "iii", tei,cr,1);
            SET(T1,10);
            NEXTSTATE IN1_3;
ENDSTATE

ENDPROCESS

/*****************************************************************/

det_0tei(call_id)
int     call_id;
{
        return(cdb[call_id].oinfo.TEI_0);
}

det_0cr(call_id)
int     call_id;
{
        return(cdb[call_id].oinfo.CR_0);
}

may_cl(call_id, option )
int     call_id;
char    option;
{
        switch(option) {

        case '0':
        case 'C':
                    if ((cdb[call_id].state == ST_NULL) ||
                        (cdb[call_id].state == ST_CLEAR) ||
                        (cdb[call_id].state == ST_RLCP) ||
                        (cdb[call_id].state == ST_RELCNF) ) {
                                return(TRUE);
                    } else {
                                return(FALSE);
                    }
        case 'T':
                    if (cdb[call_id].tinfo.T_state == st_relcpl)
                                return(TRUE);
                    else
                                return(FALSE);

        }
}

clean(call_id)
int     call_id;
{
        int TEI1,TEI2,CR;

        CR = cdb[call_id].tinfo.CR_T;
        TEI1 = cdb[call_id].tinfo.TEI;

        NT_STA->ter_cr[CR] = 0;
        setrecog(0,TEI1,CR,0);

        cdb[call_id].tinfo.T_state = st_null;
        cdb[call_id].tinfo.TEI = 0;
        cdb[call_id].tinfo.act_tei = 0;
        cdb[call_id].tinfo.tid = 0;
        rel_crt(call_id);

        CR = cdb[call_id].oinfo.CR_0;
        TEI2 = cdb[call_id].oinfo.TEI_0;
        setrecog(0,TEI2,CR,0);
        cdb[call_id].oinfo.TEI_0 = 0;
        cdb[call_id].oinfo.CR_0 = 0;

        cdb[call_id].sw = FALSE;
        cdb[call_id].state = ST_NULL;
}
```

```
rel_cid(call_id)
int     call_id;
{
        respool->cid[call_id] = FREE;
}

syskill(pid,cid,option)
int pid,cid;
char option;
{
        switch(option) {
                case 'I':
                        if( NT_STA->outpid[cid]==0 && NT_STA->ccpid[cid]==0 )
                                clear( cid );
                        NT_STA->inpid[cid] = 0;
                        kill(pid);
                        break;
                case 'C':
                        if( NT_STA->outpid[cid]==0 && NT_STA->inpid[cid]==0 )
                                clear( cid );
                        NT_STA->ccpid[cid] = 0;
                        kill(pid);
                        break;
                case 'O':
                        if( NT_STA->ccpid[cid]==0 && NT_STA->inpid[cid]==0 )
                                clear( cid );
                        NT_STA->outpid[cid] = 0;
                        kill(pid);
                        break;
        }
}

clear(cid)
int cid;
{
        clean(cid);
        rel_chts(cid);
        rel_arb(cid);
        rel_cid(cid);
}

rel_arb(cid)
int cid;
{
        int port1,port2;

        port1 = cdb[cid].oinfo.pori;
        port2 = cdb[cid].tinfo.poro;
        respool->arbport[port1] = FREE;
        respool->arbport[port2] = FREE;
}

rel_chts (call_id)              /* release B_ch & ts    */
int     call_id;
{
        int     tijds11, tijds12;
        int     ch1, ch2;
        int     int1, int2;

        NT_STA->n_acca--;
        tijds11  =  cdb[call_id].oinfo.ts_O;
        ch1  = cdb[call_id].oinfo.B_CH_O;
        int1 = cdb[call_id].oinfo.ITRF_O;
        tijds12  =  cdb[call_id].tinfo.ts_T;
        ch2  =  cdb[call_id].tinfo.B_CH_T;
        int2 =  cdb[call_id].tinfo.ITRF_T;

        NT_STA->b_ch[int1][ch1][call_id]  =  0;
        NT_STA->b_ch[int2][ch2][call_id]  =  0;

        respool->ch[int1][ch1]  =  FREE;
        respool->ch[int2][ch2]  =  FREE;
        respool->ts[tijds11]  =  FREE;
        respool->ts[tijds12]  =  FREE;
}

rel_crt(call_id)
int     call_id;
```

```
{
        int     crv;

        crv = cdb[call_id].tinfo.CR_T;
        if( crv != 0 ) {
                NT_STA->ter_cr[crv] = 0;
                respool->cr_t[crv] = FREE;
                cdb[call_id].tinfo.CR_T = 0;
        }
}

kill(pid)
int     pid;
{
        sys(8,pid);
}

setrecog(send,tei,cr,cid)
int send,tei,cr,cid;
{
        register int i;

        for(i=0;i<64;i++) {
                if( NT_STA->val_send[i]==send &&
                    NT_STA->val_tei[i]==tei &&
                    NT_STA->val_cr[i]==cr )
                        break;
        }
        if( i < 64 ) {          /* exist */
                if( cid == 0 ) {
                        NT_STA->val_send[i] = 0;
                        NT_STA->val_tei[i] = 0;
                        NT_STA->val_cr[i] = 0;
                        return(cid);
                }
                else
                        NT_STA->val_cid[i] = cid;
        }
        if( cid == 0 )                  /* does not exist */
                return(cid);
        for(i=0;i<64;i++) {
                if( NT_STA->val_send[i]==0 &&
                    NT_STA->val_tei[i]==0 &&
                    NT_STA->val_cr[i]==0 )
                        break;
        }
        if( i < 64 ) {
                NT_STA->val_send[i] = send;
                NT_STA->val_tei[i] = tei;
                NT_STA->val_cr[i] = cr;
                NT_STA->val_cid[i] = cid;
                return(cid);
        }
        else
                error("NT_STA->val overflow");
}

error(fmt,x1,x2,x3,x4)
char *fmt;
int x1,x2,x3,x4;
{
        printf("in1 error:");
        printf(fmt,x1,x2,x3,x4);
        printf("0);
/*      sys(8,0);       */
}

/****    1507  ****************************************/

se_tei(tid)
int     tid;
{
        int     index;

        for( index=1; index<5; index++)
                if( NT_STA->val_tid[index] == tid ) return(index);
        error(" TEI?s value of TID not recog");
        return (-1);
```

```
}

se_itf (tei)
int     tei;
{
        int     y;

        y = NT_STA->val_inf[tei];

        if (y < 0) {
                error("interface of tei= %d not recog", tei);
                return (-1);
        }  else
                return (y);
}

interface (tid)
int     tid;
{
        int     tei,itf;

        if( (tei=se_tei(tid)) > 0 && (itf=se_itf(tei)) >= 0 )
                return(itf);
        return(-1);
}
```

```
/*
 *      Process CC
 *      ts#, crt, b_ch allocation
 *
 *
 *
 *
 *
 */

#include        "b:msg.h"
#include        "b:def.h"
#include        "b:struct.h"
#include        "b:mac.h"
#include        "b:signals.h"


/*      define  port            */

#define         fromOUT         6
#define         fromIN          5
#define         toIN            5
#define         toOUT           6
#define         OUTtoL2         8
#define         ARBtoOUT        9
#define         toPBC           15

struct  CINFO   *cdb;           /* [MAX_CID] */
struct  resources *respool;
struct  status  *NT_STA;
int     ENV;

PROCESS( CC1 )
        int     CID,TEI,Itf,CH_ID,y,T,O,ch_ter,ch_ori,xnr,i;
        int     SELF;

        cdb = *((struct CINFO **)0x110);
        respool = *((struct resources **)0x114);
        NT_STA = *((struct status **)0x118);

        SELF = sys(14,"me");
        ENV = sys(14,"/nt");

STATE( CC1_O )                                                  /* 1507 */
        INPUT SET_IND: PARM(&mbuf,"iiii",&CID,&Itf,&CH_ID,&xnr);

        if (NT_STA->n_acca ==2) {
                        /* 2 active call, no CH avail. */
                error("NT12 has two internal call already, no capacity left");
                OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                syskill(SELF, CID, 'C');
        };
                        /*      xnr busy ?   */

        for (i=1; i<= MAX_CID; i++) {
                if (respool->cid[i] == OCC) {
                        if (cdb[i].tinfo.tid == xnr) {
                                if (cdb[i].state == ST_ACTIVE)
                                        error("xnr %d busy", xnr);
                                else
                                        error("xnr %d busy", xnr);
                                OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                                syskill(SELF, CID, 'C');
                        }
                }
        }
        /*   det origination interface y */

                if ( (y=se_itf(cdb[CID].oinfo.TEI_O)) < 0) {
                        error("origination intf not found");
                        OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                        syskill(SELF, CID, 'C');
                };

                if( (CH_ID==00) || (CH_ID==11) || (CH_ID==12) )
                        goto LC1_10;
                else {
                        if ( (CH_ID==21) || (CH_ID==22) )
```

```
                                goto LC1_09;
                        else {
                                error("CH_ID not spec");
                                OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                                syskill(SELF,CID,'C');
                        }
                };
/****  channel selection on preference basic  *******/
/******    1707 ******/
LC1_09:
        if( (ch_ori=bch_alloc(CID,'D',y,CH_ID)) > 0 ) {

                if( (ch_ter=bch_alloc(CID,'T',Itf,0)) > 0 ) {

                        if( adm_reservation(CID,Itf,xnr) < 0) {
                                error("reservation fail");
                                OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                                syskill( SELF, CID, 'C');
                        } else {
                                NEXTSTATE CC1_1;
                        }
                } else {
                        error("bch-alloc ter. fail at itf=%d CH_ID=0", Itf);
                        OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                        syskill(SELF, CID, 'C');
                }
        } else {
                error("bch-alloc ori fail at itf=%d ch=%d", y, CH_ID);
                OUTPUT( &mbuf,toIN, REJ_REQ,"i", CID );
                syskill(SELF,CID,'C');
        }

LC1_10:                 /* channel selection depends on CH_ID */
/* 1507 */
        if( (ch_ter=bch_alloc(CID,'T',Itf,0)) > 0 ) {

                if( (ch_ori=bch_alloc(CID,'D',y,CH_ID)) > 0 ) {

                        if( adm_reservation(CID,Itf,xnr) < 0) {
                                error("reservation fail");
                                OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                                syskill( SELF, CID, 'C');
                        } else {
                                NEXTSTATE CC1_1;
                        }
                } else {
                        error("bch-alloc ori fail at itf=%d ch=%d", y, CH_ID);
                        OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                        syskill(SELF, CID, 'C');
                }
        } else {
                error("bch-alloc ter. fail at itf=%d CH_ID=0", Itf);
                OUTPUT( &mbuf,toIN, REJ_REQ,"i", CID );
                syskill(SELF,CID,'C');
        }
ENDSTATE

STATE( CC1_1 )
        INPUT CALL_P_IND: PARM(&mbuf, "ii", &CID,&TEI);
                goto LC1_11;
        INPUT ALERT_IND: PARM(&mbuf, "ii", &CID,&TEI);
                goto LC1_12;
        INPUT CLEAR_IND: PARM(&mbuf, "i", &CID);
                goto LC1_13;
        INPUT SET_CNF: PARM(&mbuf, "ii", &CID,&TEI);
                goto LC1_14;

LC1_11:
        cdb[CID].state = ST_CLPR;
        cdb[CID].tinfo.act_tei = TEI;
        sw_PBC( CID, "ON" );
        NEXTSTATE CC1_2;
LC1_12:
        cdb[CID].state = ST_ALER;
        cdb[CID].tinfo.act_tei = TEI;
        if( cdb[CID].sw == FALSE ) {
                sw_PBC( CID, "ON" );
        }
```

```
        OUTPUT( &mbuf, toIN, ALERT_REQ, "i",CID);
        NEXTSTATE CC1_2;
LC1_13:
        cdb[CID].state = ST_RLCP;
        cdb[CID].tinfo.act_tei = 0;
        if( cdb[CID].sw == TRUE ) {
                sw_PBC( CID,"OFF" );
        }
        if( SENDER == fromOUT )
                OUTPUT( &mbuf, toIN, CLEAR_REQ, "i", CID);
        else
                OUTPUT( &mbuf, toOUT, CLEAR_REQ, "i", CID);
        syskill(SELF,CID,'C');          /* clean(CID) perform by IN or OUT */

LC1_14:
        cdb[CID].state = ST_ACTIVE;
        cdb[CID].tinfo.act_tei = TEI;
        if( cdb[CID].sw == FALSE )
                sw_PBC( CID, "ON" );
        OUTPUT( &mbuf, toIN, SET_RESP, "i", CID);
        NEXTSTATE CC1_3;
ENDSTATE


STATE(CC1_2)
        INPUT CLEAR_IND: PARM(&mbuf, "i", &CID); /* TEI not needed */
                goto LC1_21;
        INPUT ALERT_IND: PARM(&mbuf, "ii", &CID, &TEI);
                goto LC1_22;
        INPUT SET_CNF: PARM(&mbuf, "ii", &CID, &TEI);
                goto LC1_23;
LC1_21:
        cdb[CID].state = ST_CLEAR;
        if( SENDER == fromIN ) {

                        /* clear from in_ ori side */

                if( cdb[CID].sw == TRUE )
                        sw_PBC( CID, "OFF" );
                OUTPUT( &mbuf, toOUT, CLEAR_REQ, "i", CID);
                syskill(SELF,CID,'C');          /* clean(CID) perform by IN or OUT */
        } else {
                if( SENDER == fromOUT) {

                        /* clear from out_ter side */

                        cdb[CID].tinfo.act_tei = 0;
                        if( cdb[CID].sw == TRUE )
                                sw_PBC( CID, "OFF" );
                        OUTPUT( &mbuf, toIN, CLEAR_REQ, "i", CID);
                        syskill(SELF,CID,'C');  /* clean(CID) perform by IN or OUT */
                } else {
                        error("CID - in/out SENDER error ");
                        NEXTSTATE CC1_2;
                }
        }

LC1_22:
        cdb[CID].state = ST_ALER;
        cdb[CID].tinfo.act_tei = TEI;
        OUTPUT( &mbuf, toIN, ALERT_REQ, "i", CID);
        NEXTSTATE CC1_2;
LC1_23:
        cdb[CID].state = ST_ACTIVE;
        cdb[CID].tinfo.act_tei = TEI;
        if( cdb[CID].sw == FALSE )
                sw_PBC ( CID,"ON" );
        OUTPUT( &mbuf, toIN, SET_RESP, "i", CID);
        NEXTSTATE CC1_3;

ENDSTATE


STATE(CC1_3)
        INPUT DISC_IND: PARM (&mbuf, "i", &CID);
                goto LC1_31;
        INPUT REJ_IND: PARM (&mbuf, "i", &CID);
                goto LC1_32;
```

```
LC1_31:
        cdb[CID].state = ST_DISCRQ;
        if( SENDER == fromIN ) {
                OUTPUT( &mbuf, toIN, REL_REQ, "i", CID);
                OUTPUT( &mbuf, toOUT, DISC_REQ, "i", CID);
                NEXTSTATE CC1_4;
        } else {
                if( SENDER == fromOUT) {
                        OUTPUT( &mbuf, toOUT, REL_REQ, "i", CID);
                        OUTPUT( &mbuf, toIN, DISC_REQ, "i", CID);
                        NEXTSTATE CC1_4;
                } else {
                        error("CID - in/out SENDER error in DISC");
                }
        }

LC1_32:
        if( SENDER == fromIN) {
                OUTPUT( &mbuf, toOUT, REJ_REQ, "i", CID);
        } else {
                if( SENDER == fromOUT) {
                        OUTPUT( &mbuf, toIN, REJ_REQ, "i", CID);
                } else {
                        error("CID - in/out SENDER error in DISC");
                        NEXTSTATE CC1_3;
                }
        }
        sw_PBC(CID, "OFF");
        syskill(SELF,CID,'C');
ENDSTATE

STATE(CC1_4)
        INPUT DISC_IND: PARM(&mbuf,"i",&CID);
                goto LC1_41;
        INPUT REL_IND: PARM(&mbuf,"i",&CID);
                goto LC1_42;
        INPUT REL_CNF: PARM(&mbuf,"i",&CID);
                goto LC1_43;
LC1_41:
        cdb[CID].state = ST_DISC;
        if( SENDER == fromIN) {
                OUTPUT( &mbuf, toOUT, REL_REQ, "i", CID);
        } else {
                if( SENDER == fromOUT) {
                        OUTPUT( &mbuf, toIN, REL_REQ, "i", CID);
                } else {
                        error("CID - in/out SENDER error in CC1_4 ");
                }
        }
        NEXTSTATE CC1_4;
LC1_42:
        if( SENDER == fromIN) {
                OUTPUT( &mbuf, toIN, REL_RESP, "i", CID);
        } else {
                if( SENDER == fromOUT) {
                        OUTPUT( &mbuf, toOUT, REL_RESP, "i", CID);
                } else {
                        error("CID - in/out SENDER error in CC1_4 ");
                }
        }
        if( cdb[CID].state == ST_RELCNF) {
                if( cdb[CID].sw == TRUE)
                        sw_PBC( CID, "OFF");
                syskill(SELF,CID,'C');  /* clean(CID), rel_crt by IN, OUT */
        } else {
                /*  case  STATE= DISCRQ  */
                cdb[CID].state = ST_RELIND;
                NEXTSTATE CC1_4;                /* wait fo REL_CNF */
        }

LC1_43:
        if( cdb[CID].state == ST_RELIND) {
                if( cdb[CID].sw == TRUE)
                        sw_PBC( CID, "OFF");
                syskill(SELF,CID,'C');
        } else {
                if( cdb[CID].state == ST_RELCNF) {
                        syskill(SELF,CID,'C');
```

```
                } else {   /* in DSICRQ or DISC STATE */
                        if( cdb[CID].sw == TRUE )
                                sw_PBC( CID, "OFF");
                        cdb[CID].state = ST_RELCNF;
                        NEXTSTATE CC1_4;
                }
        }
ENDSTATE

ENDPROCESS

/*****************************************************************/

bch_alloc( call_id, call_pty, intf, option )
int call_id, intf, option;
char call_pty;
                                /* O/T b_ch for ori/ter */
                                /* intf= 0,1,2          */
                                /* option               */
                                /* 0    any             */
                                /* 11   any/prefer B1   */
                                /* 12   any/prefer B2   */
                                /* 21   prefer B1       */
                                /* 22   prefer B2       */
                                /* result = -1  fail    */
                                /*          1   success */
{
        char    channel=0;

        switch(option) {
        case 0:
        case 11:   if ( respool->ch[intf][1] == FREE )  {
                                channel  =  1;
                                respool->ch[intf][1]  =  OCC;
                        } else {
                                if( respool->ch[intf][2] == FREE ) {
                                        channel  =  2;
                                        respool->ch[intf][2]  =  OCC;
                                } else {
                                        return ( -1 );
                                }
                        }
                   break;
        case 12:
                   if ( respool->ch[intf][2] == FREE ) {
                                channel  =  2;
                                respool->ch[intf][2]  =  OCC;
                        } else {
                                if( respool->ch[intf][1] == FREE ) {
                                        channel  =  1;
                                        respool->ch[intf][1]  =  OCC;
                                } else {
                                        return ( -1 );
                                }
                        }
                   break;
        case 21:
                        if( respool->ch[intf][1] == FREE ) {
                                channel  =  1;
                                respool->ch[intf][1]  =  OCC;
                        } else {
                                return ( -1 );
                        }
                        break;
        case 22:
                   if ( respool->ch[intf][2] == FREE ) {
                                channel  =  2;
                                respool->ch[intf][2]  =  OCC;
                        } else {
                                return ( -1 );
                        }
                   break;
        }
        if (call_pty == 'T') {  /* terminated side */
                cdb[call_id].tinfo.TBIDR =  intf;
                cdb[call_id].tinfo.B_CH_T =  channel;
```

```
                cdb[call_id].tinfo.ITRF_T = intf;
        } else {
                if (call_pty == '0') {  /* originated side      */
                        cdb[call_id].oinfo.OBIDS =  intf;
                        cdb[call_id].oinfo.B_CH_O =  channel;
                        cdb[call_id].oinfo.ITRF_O =  intf;
                } else {
                        error("call_pty not correct");
                }
        }
        return(channel);
}

adm_reservation (call_id,Itf,xnr)
int call_id,Itf,xnr;
{
        int crter,nwopid,SELF,ch_ori,ch_ter;
        struct mbuf mbuf;

        SELF = sys(14,"me");
        if( res_ts(call_id) > 0 ) {
            if( (crter=crt_alloc(call_id,xnr)) > 0 ) {
                if( (nwopid=creat("out1",call_id) ) > 0 ) {
                                /*  connect arb-out      */
                                /*  connect cc-out       */
                                /*  connect out_L2       */
                                /*  connect cc-pbc       */
                    if( connect(NT_STA->arbpid, cdb[call_id].tinfo.poro, nwopid, ARBtoOUT)
                        return (-1);
                    }
                    if( connect(SELF, toOUT, nwopid, 7) < 0) {
                        return (-1);
                    }

                /* ENV == cdb[call_id].tinfo.TBIDR  */

                    if( connect(nwopid,OUTtoL2,ENV,cdb[call_id].oinfo.pori+11) < 0) {
                        return (-1);
                    }

                    if( connect(SELF,toPBC,ENV,cdb[call_id].oinfo.pori+8) < 0) {
                        return (-1);
                    }
                    NT_STA->n_acca += 1 ;

/* 1507 */
                    ch_ori = cdb[call_id].oinfo.B_CH_O;
                    ch_ter = cdb[call_id].tinfo.B_CH_T;

                    OUTPUT(&mbuf,toIN, CALL_P_REQ,"ii"; call_id, ch_ori);
                    OUTPUT(&mbuf,toOUT, SET_REQ,"iii", call_id, Itf, ch_ter);
                    NT_STA->outpid[call_id] = nwopid;
                    return (1);

                } else {
                        error("creat OUT error");
                        return (-1);
                }
        } else  {
                error("crter alloc error");
                return (-1);
        }
    } else {
        error("ts alloc error, or no ts avail");
        return (-1);
    }
}

crt_alloc (call_id,xnr)
                                /*  -1 if fail, update tid */
                                /* return crt if success   */
int     call_id, xnr;
{
        int crv = 129;

        while ( respool->cr_t[crv] != FREE ) {
                if ( crv == 256 ) {
                        error("no crt gen. ");
```

```
                        return(-1);
                }
                ++crv;
        }
        respool->cr_t[crv]  =  OCC;
        NT_STA->ter_cr[crv]  =  call_id;
        cdb[call_id].tinfo.CR_T  = crv;
        cdb[call_id].tinfo.tid = xnr ;

        if ( (cdb[call_id].tinfo.TEI=se_tei(xnr)) > 0)
                return (crv);
        else
                return(-1);
}

res_ts (call_id)
                                /*  -1 if fail          */
                                /* 2 ts# must be reserved */
                                /* only internal adm.   */
                                /* return 1 if success  */
int     call_id;
{
        int index = 1;
        int itf,bid;

        while( (respool->ts[index] != FREE) && (index <= 64) ) {
                if( index == 64 ) {
                        error("no ts# available");
                        return(-1);
                }
                ++index;
        }

        if( index < 64 ) {
                respool->ts[index] =  OCC;
                cdb[call_id].oinfo.ts_O =  index;       /* originated side */
                itf =  cdb[call_id].oinfo.ITRF_O;
                bid = cdb[call_id].oinfo.B_CH_O;
                NT_STA->b_ch[itf][bid][call_id] = index;
                index++;

                while( (respool->ts[index] != FREE) && ( index <= 64 ) ) {
                        ++index;
                }
                if (( index <= 64) && (respool->ts[index] == FREE)) {
                        respool->ts[index] = OCC;
                        cdb[call_id].tinfo.ts_T = index; /* terminated side */
                        itf = cdb[call_id].tinfo.ITRF_T;
                        bid = cdb[call_id].tinfo.B_CH_T;
                        NT_STA->b_ch[itf][bid][call_id] = index;
                        return(1);
                } else {
                        error("no ts# available");
                        return(-1);
                }
        }
}

sw_PBC( call_id, option )
int call_id;
char option[];
{
        int O_ch,O_itf,O_ts,T_ch,T_itf,T_ts;
        struct mbuf mbuf;

        O_ch =  cdb[call_id].oinfo.B_CH_O;
        O_itf =  cdb[call_id].oinfo.ITRF_O;
        O_ts =  cdb[call_id].oinfo.ts_O;

        T_ch =  cdb[call_id].tinfo.B_CH_T;
        T_itf =  cdb[call_id].tinfo.ITRF_T;
        T_ts =  cdb[call_id].tinfo.ts_T;

        if ( strcmp(option,"ON") == 0 ) {
                OUTPUT(&mbuf, toPBC, SW_ON, "iiiiii", O_ch, O_itf, O_ts,T_ch, T_itf, T_ts);
                cdb[call_id].sw = TRUE;
        } else {
                if( strcmp(option,"OFF") == 0 ) {
```

```
                        OUTPUT (&mbuf, toPBC, SW_OFF, "iiiiii", O_ch, O_itf, O_ts,T_ch, T_i
                        cdb[call_id].sw =  FALSE;
                } else {
                        error("PBC CMD option error");
                }
        }
}

rel_chts (call_id)              /* release B_ch & ts    */
int     call_id;
{
        int     tijds11, tijds12;
        int     ch1, ch2;
        int     int1, int2;

        NT_STA->n_acca--;
        tijds11  =  cdb[call_id].oinfo.ts_O;
        ch1  = cdb[call_id].oinfo.B_CH_O;
        int1 = cdb[call_id].oinfo.ITRF_O;
        tijds12  =  cdb[call_id].tinfo.ts_T;
        ch2  =  cdb[call_id].tinfo.B_CH_T;
        int2 =  cdb[call_id].tinfo.ITRF_T;

        NT_STA->b_ch[int1][ch1][call_id]  =  0;
        NT_STA->b_ch[int2][ch2][call_id]  =  0;

        respool->ch[int1][ch1]  =  FREE;
        respool->ch[int2][ch2]  =  FREE;
        respool->ts[tijds11]  =  FREE;
        respool->ts[tijds12]  =  FREE;
}

clean(call_id)
int     call_id;
{
        int TEI1,TEI2,CR;

        CR = cdb[call_id].tinfo.CR_T;
        TEI1 = cdb[call_id].tinfo.TEI;

        NT_STA->ter_cr[CR] = 0;
        setrecog(0,TEI1,CR,0);

        cdb[call_id].tinfo.T_state = st_null;
        cdb[call_id].tinfo.TEI = 0;
        cdb[call_id].tinfo.act_tei = 0;
        cdb[call_id].tinfo.tid = 0;                     /* 1507 */
        rel_crt(call_id);

        CR = cdb[call_id].oinfo.CR_O;
        TEI1 = cdb[call_id].oinfo.TEI_O;
        setrecog(0,TEI1,CR,0);
        cdb[call_id].oinfo.TEI_O = 0;
        cdb[call_id].oinfo.CR_O = 0;

        cdb[call_id].sw = FALSE;
        cdb[call_id].state = ST_NULL;
}

connect(pro1,port1,pro2,port2)
int     pro1,port1,pro2,port2;
{
        if( sys(3,pro1,port1,pro2,port2) ) {
                printf("connect error %d[%d] %d[%d]0,pro1,port1,pro2,port2);
                return(-1);
        }
        return(1);
}

rel_cid(call_id)
int     call_id;
{
        respool->cid[call_id] = FREE;
}

rel_crt(call_id)
int     call_id;
{
```

```
        int     crv;

        crv = cdb[call_id].tinfo.CR_T;
        if( crv != 0 ) {
                NT_STA->ter_cr[crv] = 0;
                respool->cr_t[crv] = FREE;
                cdb[call_id].tinfo.CR_T = 0;
        }
}

creat(prty,cid)
char    prty[];         /* process type IN1, OUT1, CC1  */
{
        int OFFSPRING;
        char    name[16];
        char    number[8];

        strcpy(name,prty);
        sprintf(number,"_%d",cid);
        strcat(name,number);
        if( (OFFSPRING=sys(6,name,prty,20,0,0)) < 0) {
                 error("can't create %s",prty);
                 return(-1);
        }
        sys(7,OFFSPRING);
        return(OFFSPRING);
}

syskill(pid,cid,option)
int pid,cid;
char option;
{
        switch(option) {
                case 'I':
                        if( NT_STA->outpid[cid]==0 && NT_STA->ccpid[cid]==0 )
                                clear( cid );
                        NT_STA->inpid[cid] = 0;
                        kill(pid);
                        break;
                case 'C':
                        if( NT_STA->outpid[cid]==0 && NT_STA->inpid[cid]==0 )
                                clear( cid );
                        NT_STA->ccpid[cid] = 0;
                        kill(pid);
                        break;
                case 'O':
                        if( NT_STA->ccpid[cid]==0 && NT_STA->inpid[cid]==0 )
                                clear( cid );
                        NT_STA->outpid[cid] = 0;
                        kill(pid);
                        break;
        }
}

clear(cid)
int cid;
{
        clean(cid);
        rel_chts(cid);
        rel_arb(cid);
        rel_cid(cid);
}

rel_arb(cid)
int cid;
{
        int port1,port2;

        port1 = cdb[cid].cinfo.pori;
        port2 = cdb[cid].tinfo.poro;
        respool->arbport[port1] = FREE;
        respool->arbport[port2] = FREE;
}

kill(pid)
int     pid;
{
        sys(8,pid);
```

```
}

setrecog(send,tei,cr,cid)
int send,tei,cr,cid;
{
        register int i;

        for(i=0;i<64;i++) {
                if( NT_STA->val_send[i]==send &&
                    NT_STA->val_tei[i]==tei &&
                    NT_STA->val_cr[i]==cr )
                        break;
        }
        if( i < 64 ) {          /* exist */
                if( cid == 0 ) {
                        NT_STA->val_send[i] = 0;
                        NT_STA->val_tei[i] = 0;
                        NT_STA->val_cr[i] = 0;
                        return(cid);
                }
                else
                        NT_STA->val_cid[i] = cid;
        }
        if( cid == 0 )                  /* does not exist */
                return(cid);
        for(i=0;i<64;i++) {
                if( NT_STA->val_send[i]==0 &&
                    NT_STA->val_tei[i]==0 &&
                    NT_STA->val_cr[i]==0 )
                        break;
        }
        if( i < 64 ) {
                NT_STA->val_send[i] = send;
                NT_STA->val_tei[i] = tei;
                NT_STA->val_cr[i] = cr;
                NT_STA->val_cid[i] = cid;
                return(cid);
        }
        else
                error("NT_STA->val overflow");
}

error(fmt,x1,x2,x3,x4)
char *fmt;
int x1,x2,x3,x4;
{
        printf("cc1 error: ");
        printf(fmt,x1,x2,x3,x4);
        printf("0);
/*      sys(8,0);       */
}

/*******    1507  ****************/
se_itf (tei)
int     tei;
{
        int     y;

        y = NT_STA->val_inf[tei];

        if (y < 0) {
                error("interface of tei= %d not recog", tei);
                return (-1);
        }  else
                return (y);
}


se_tei(tid)
int     tid;
{
        int     index=1;

        while (NT_STA->val_tid[index] != tid)
                index++;

        if (index > 4) {
                error(" TEI7s value of TID not recog");
```

```
                return (-1);
        } else
                return (index);
}
```

```
/*
 *      Process OUT
 *      perform messages selection,
 *      clean(CID)
 *
 *
 *
 */

#include        "b:msg.h"
#include        "b:def.h"
#include        "b:struct.h"
#include        "b:mac.h"
#include        "b:signals.h"

/*      define  port            */

#define         frCC    7
#define         toCC    7
#define         fromARB 9
#define         toL2    8


struct  CINFO   *cdb;           /* [MAX_CID] */
struct  resources *respool;
struct  status  *NT_STA;
int     ENV;

PROCESS( OUT1 )
        int     CID,TEI,CH_ID,CR,itf,cha_ter,tid;
        int     SELF;

        cdb = *((struct CINFO **)0x110);
        respool = *((struct resources **)0x114);
        NT_STA = *((struct status **)0x118);

        SELF = sys(14,"me");
        ENV = sys(14,"/nt");

STATE( OUT1_0 )
        INPUT SET_REQ: PARM(&mbuf,"iii",&CID,&itf,&cha_ter);
                CR = cdb[CID].tinfo.CR_T;
                tid = cdb[CID].tinfo.tid;
                TEI = cdb[CID].tinfo.TEI;
                OUTPUT( &mbuf, toL2, SETUP, "iiiiii",TEI,CR,0,cha_ter,1,tid);

                /*              SET(T303);              */

                NEXTSTATE OUT1_1;
ENDSTATE

STATE( OUT1_1 )
        INPUT oREL_CPL: PARM(&mbuf,"ii",&CID,&TEI);
                goto LOU1_11;
        INPUT oCALL_PROCEED: PARM(&mbuf,"ii",&CID,&TEI);
                goto LOU1_12;
        INPUT oALERTING: PARM(&mbuf,"ii",&CID,&TEI);
                goto LOU1_13;
        INPUT oCONNECT: PARM(&mbuf,"ii",&CID,&TEI);
                goto LOU1_14;
        INPUT CLEAR_REQ: PARM(&mbuf,"i",&CID);
                goto LOU1_24;
LOU1_11:
        OUTPUT( &mbuf, toCC, CLEAR_IND, "i", CID);
        syskill(SELF,CID,'0');

LOU1_12:
                                        /*  RESET (t303) */
                                        /*  SET (10)     */
        if (TEI == cdb[CID].tinfo.TEI) {
                cdb[CID].tinfo.T_state = st_calprc;
                cdb[CID].tinfo.act_tei = TEI;
                OUTPUT( &mbuf, toCC, CALL_P_IND, "ii", CID, TEI);
                NEXTSTATE OUT1_2;
        } else {
                error("TEI in msg not expected");
                NEXTSTATE OUT1_1;
        }
```

```
LOU1_13:
                                                    /* RESET (T303) */
        if (TEI == cdb[CID].tinfo.TEI) {
                cdb[CID].tinfo.T_state = st_alert;
                cdb[CID].tinfo.act_tei = TEI;
                OUTPUT( &mbuf, toCC, ALERT_IND, "ii", CID, TEI);
                NEXTSTATE OUT1_2;
        } else {
                error("TEI in msg not expected");
                NEXTSTATE OUT1_1;
        }

LOU1_14:
                                                    /* RESET(T303)  */
        if (TEI == cdb[CID].tinfo.TEI) {
                cdb[CID].tinfo.T_state = st_active;
                cdb[CID].tinfo.act_tei = TEI;
                CR = cdb[CID].tinfo.CR_T;
                OUTPUT( &mbuf, toCC, SET_CNF, "ii", CID, TEI);
                OUTPUT( &mbuf, toL2, CONNECT_ACK, "iii", TEI, CR, 0);
                NEXTSTATE OUT1_3;
        } else {
                error("TEI in msg not expected");
                NEXTSTATE OUT1_1;
        }

ENDSTATE

STATE(OUT1_2)

        INPUT oALERTING: PARM(&mbuf,"ii",&CID,&TEI);
                goto LOU1_21;
        INPUT oCONNECT: PARM(&mbuf,"ii",&CID,&TEI);
                goto LOU1_22;
        INPUT oDISC: PARM(&mbuf,"ii",&CID,&TEI);
                goto LOU1_23;
        INPUT CLEAR_REQ: PARM(&mbuf,"i",&CID);
                goto LOU1_24;

LOU1_21:
        if(TEI == cdb[CID].tinfo.TEI ) {
                cdb[CID].tinfo.T_state = st_alert;
                cdb[CID].tinfo.act_tei = TEI;
                OUTPUT( &mbuf, toCC, ALERT_IND, "ii", CID, TEI);
        } else
                error("TEI in msg not expected");
        NEXTSTATE OUT1_2;

LOU1_22:
        if( TEI == cdb[CID].tinfo.TEI ) {
                cdb[CID].tinfo.T_state = st_active;
                cdb[CID].tinfo.act_tei = TEI;
                OUTPUT( &mbuf, toCC, SET_CNF, "ii", CID, TEI);
                CR = cdb[CID].tinfo.CR_T;
                OUTPUT( &mbuf, toL2, CONNECT_ACK, "iii", TEI, CR, 0);
                NEXTSTATE OUT1_3;
        } else {
                error("TEI in msg not expected");
                NEXTSTATE OUT1_2;
        }
LOU1_23:
        if( cdb[CID].tinfo.TEI == TEI) {
                cdb[CID].tinfo.T_state = st_disc;
                CR = cdb[CID].tinfo.CR_T;
                OUTPUT( &mbuf, toL2, RELEASE, "iii", TEI, CR, 0);
                SET(T3, 10);
                OUTPUT( &mbuf, toCC, CLEAR_IND, "i", CID);
                NEXTSTATE OUT1_4;
        } else {
                error("TEI in msg not expected");
                NEXTSTATE OUT1_2;
        }
LOU1_24:
        CR = cdb[CID].tinfo.CR_T;
        OUTPUT( &mbuf, toL2, DISC, "iii", cdb[CID].tinfo.TEI, CR, 0);
        SET(T6, 10);
        NEXTSTATE OUT1_4;
```

```
ENDSTATE

STATE(OUT1_3)

        INPUT oDISC: PARM(&mbuf,"ii", &CID,&TEI);
                goto LOU1_31;
        INPUT oREL_CPL: PARM(&mbuf,"ii", &CID,&TEI);
                goto LOU1_32;
        INPUT DISC_REQ: PARM(&mbuf,"i", &CID);
                goto LOU1_33;
        INPUT REJ_REQ: PARM(&mbuf,"i", &CID);
                goto LOU1_34;

LOU1_31:
        if( TEI == cdb[CID].tinfo.act_tei ) {
                OUTPUT( &mbuf, toCC, DISC_IND, "i", CID);
                cdb[CID].tinfo.T_state = st_discrq;
                NEXTSTATE OUT1_4;
        } else {
                error ("In LOU1_31 TEI's value error in DISC msg");
                NEXTSTATE OUT1_3;
        };

LOU1_32:
        if( TEI == cdb[CID].tinfo.act_tei ) {
                OUTPUT( &mbuf, toCC, REJ_IND, "i", CID);
                syskill(SELF,CID,'D');
        } else {
                error("TEI in msg not expected");
                NEXTSTATE OUT1_3;
        }

LOU1_33:
        if( cdb[CID].tinfo.T_state == st_active)  {
                cdb[CID].tinfo.T_state = st_discrq;
                TEI = cdb[CID].tinfo.act_tei;
                CR = cdb[CID].tinfo.CR_T;
                OUTPUT( &mbuf, toL2, DISC, "iii", TEI, CR, 0);
                SET(T6, 10);
                NEXTSTATE OUT1_4;
        } else {
                error("in LOU1_33: T_state not correct ");
                NEXTSTATE OUT1_3;
        }
LOU1_34:
        TEI = cdb[CID].tinfo.act_tei;
        CR = cdb[CID].tinfo.CR_T;
        OUTPUT( &mbuf, toL2, REL_CPL, "iii", TEI, CR, 0);
        syskill(SELF,CID,'D');

ENDSTATE

STATE(OUT1_4)
        INPUT oDISC: PARM( &mbuf, "ii", &CID, &TEI);
                goto LOU1_41;

        INPUT oRELEASE: PARM(&mbuf, "ii", &CID,&TEI);
                goto LOU1_42;

        INPUT REL_REQ: PARM(&mbuf, "i", &CID);
                goto LOU1_43;

        INPUT REL_RESP: PARM(&mbuf,"i", &CID);
                goto LOU1_44;

        INPUT oREL_CPL: PARM(&mbuf,"ii",&CID, &TEI);
                goto LOU1_45;
        INPUT T3:
                goto LOU1_46;
        INPUT T4:
                goto LOU1_47;
        INPUT T6:
               goto LOU1_48;
LOU1_41:
        RESET(T6);
        if( (cdb[CID].state>=ST_ACTIVE) && (TEI==cdb[CID].tinfo.act_tei) ) {
                        /* call was active  */
                OUTPUT( &mbuf, toCC, DISC_IND, "i", CID);
```

```
                up_te_st( CID, TEI, oDISC) ;

        } else {   /* call was not active, resp of CLEAR_REQ */

                CR = cdb[CID].tinfo.CR_T;
                OUTPUT( &mbuf, toL2, RELEASE, "iii", TEI, CR, 0);
                SET(T3, 10);
                up_te_st (CID,TEI, oDISC) ;
        }
        NEXTSTATE OUT1_4;

LOU1_42:
         RESET(T6);
        if( cdb[CID].state>=ST_ACTIVE && cdb[CID].state!=ST_CLEAR &&
            TEI==cdb[CID].tinfo.act_tei ) {
                        /* call was active  */
                OUTPUT( &mbuf, toCC, REL_IND, "i", CID);
                up_te_st( CID, TEI, oRELEASE);
                NEXTSTATE OUT1_4;
        } else {
                        /* call was not active.  */
                CR =  cdb[CID].tinfo.CR_T;
                OUTPUT( &mbuf, toL2, REL_CPL, "iii", TEI, CR, 0);
                syskill(SELF,CID,'0');
        }
LOU1_43:
        TEI = cdb[CID].tinfo.act_tei;
        CR = cdb[CID].tinfo.CR_T;
        OUTPUT( &mbuf, toL2, RELEASE, "iii", TEI, CR, 0);
        SET(T3,10);
        up_te_st (CID, TEI, oRELEASE);
        NEXTSTATE OUT1_4;
LOU1_44:
        TEI = cdb[CID].tinfo.act_tei;
        CR = cdb[CID].tinfo.CR_T;
        OUTPUT( &mbuf, toL2, REL_CPL, "iii", TEI, CR, 0);
        syskill(SELF,CID,'0');

LOU1_45:
        RESET(T3);
        RESET(T4);
        if( cdb[CID].state>=ST_ACTIVE && cdb[CID].state!=ST_CLEAR &&
            TEI==cdb[CID].tinfo.act_tei ) {
                        /* call was active  */
                OUTPUT( &mbuf, toCC, REL_CNF, "i", CID);
                syskill(SELF,CID,'0');

        } else {
                        /* call was not active.  */
                syskill(SELF,CID,'0');
        }
LOU1_46:
        /*      ie timeout of T3                                        */
        OUTPUT(&mbuf, toL2, RELEASE, "iii", TEI, CR, 0);
        SET(T4, 10);
        NEXTSTATE OUT1_4;
LOU1_47:
        /*      timeout of T4                                           */
        if( NT_STA->ccpid[CID]!=0 )
                syskill( NT_STA->ccpid[CID],CID,'C');
        syskill(SELF,CID,'0');

LOU1_48:
        /*      timeout of T6                                           */
        OUTPUT(&mbuf, toL2, RELEASE, "iii", TEI, CR, 0);
        SET(T3,10);
        NEXTSTATE OUT1_4;
ENDSTATE

ENDPROCESS

/*****************************************************************/

up_te_st(CID,TEI,rec_msg)  /* update TE status only on term. side */
int CID,TEI,rec_msg;
{
        int st;
```

```
        switch(rec_msg) {
                case oDISC:             st = st_discrq; break;
                case oCALL_PROCEED:     st = st_calprc; break;
                case oALERTING:         st = st_alert; break;
                case oCONNECT:          st = st_active; break;
                case oRELEASE:          st = st_disc; break;
                case oREL_CPL:          st = st_relcpl; break;
        }
        if( TEI == cdb[CID].tinfo.TEI) {
                cdb[CID].tinfo.T_state = st;
        } else {
                error("TEI not recognized in up_te_st()");
        }
}

clean(call_id)
int     call_id;
{
        int TEI1,TEI2,CR;

        CR = cdb[call_id].tinfo.CR_T;
        TEI1 = cdb[call_id].tinfo.TEI;
        NT_STA->ter_cr[CR] = 0;
        setrecog(0,TEI1,CR,0);
        cdb[call_id].tinfo.T_state = st_null;
        cdb[call_id].tinfo.TEI = 0;
        cdb[call_id].tinfo.act_tei = 0;
        cdb[call_id].tinfo.tid = 0;
        rel_crt(call_id);

        CR = cdb[call_id].oinfo.CR_O;
        TEI2 = cdb[call_id].oinfo.TEI_O;
        setrecog(0,TEI2,CR,0);
        cdb[call_id].oinfo.TEI_O = 0;
        cdb[call_id].oinfo.CR_O = 0;

        cdb[call_id].sw = FALSE;
        cdb[call_id].state = ST_NULL;
}

rel_crt(call_id)
int     call_id;
{
        int     crv;

        crv = cdb[call_id].tinfo.CR_T;
        if( crv != 0 ) {
                NT_STA->ter_cr[crv] = 0;
                respool->cr_t[crv] = FREE;
                cdb[call_id].tinfo.CR_T = 0;
        }
}

rel_cid(call_id)
int     call_id;
{
        respool->cid[call_id] = FREE;
}

syskill(pid,cid,option)
int pid,cid;
char option;
{
        switch(option) {
                case 'I':
                        if( NT_STA->outpid[cid]==0 && NT_STA->ccpid[cid]==0 )
                                clear( cid );
                        NT_STA->inpid[cid] = 0;
                        kill(pid);
                        break;
                case 'C':
                        if( NT_STA->outpid[cid]==0 && NT_STA->inpid[cid]==0 )
                                clear( cid );
                        NT_STA->ccpid[cid] = 0;
                        kill(pid);
                        break;
                case 'O':
                        if( NT_STA->ccpid[cid]==0 && NT_STA->inpid[cid]==0 )
```

```
                        clear( cid );
                    NT_STA->outpid[cid] = 0;
                    kill(pid);
                    break;
            }
}

clear(cid)
int cid;
{
        clean(cid);
        rel_chts(cid);
        rel_arb(cid);
        rel_cid(cid);
}

rel_arb(cid)
int cid;
{
        int port1,port2;

        port1 = cdb[cid].oinfo.pori;
        port2 = cdb[cid].tinfo.poro;
        respool->arbport[port1] = FREE;
        respool->arbport[port2] = FREE;
}

rel_chts (call_id)              /* RELEASE B_ch & ts     */
int     call_id;
{
        int     tijdsl1, tijdsl2;
        int     ch1, ch2;
        int     int1, int2;

        NT_STA->n_acca--;
        tijdsl1  =  cdb[call_id].oinfo.ts_O;
        ch1  = cdb[call_id].oinfo.B_CH_O;
        int1 = cdb[call_id].oinfo.ITRF_O;
        tijdsl2  =  cdb[call_id].tinfo.ts_T;
        ch2  =  cdb[call_id].tinfo.B_CH_T;
        int2 =  cdb[call_id].tinfo.ITRF_T;

        NT_STA->b_ch[int1][ch1][call_id]  =  0;
        NT_STA->b_ch[int2][ch2][call_id]  =  0;

        respool->ch[int1][ch1]  =  FREE;
        respool->ch[int2][ch2]  =  FREE;
        respool->ts[tijdsl1]  =  FREE;
        respool->ts[tijdsl2]  =  FREE;
}

kill(pid)
int     pid;
{
        sys(8,pid);
}

setrecog(send,tei,cr,cid)
int send,tei,cr,cid;
{
        register int i;

        for(i=0;i<64;i++) {
                if( NT_STA->val_send[i]==send &&
                    NT_STA->val_tei[i]==tei &&
                    NT_STA->val_cr[i]==cr )
                        break;
        }
        if( i < 64 ) {          /* exist */
                if( cid == 0 ) {
                        NT_STA->val_send[i] = 0;
                        NT_STA->val_tei[i] = 0;
                        NT_STA->val_cr[i] = 0;
                        return(cid);
                }
                else
                        NT_STA->val_cid[i] = cid;
        }
```

```
        if( cid == 0 )                  /* does not exist */
                return(cid);
        for(i=0;i<64;i++) {
                if( NT_STA->val_send[i]==0 &&
                    NT_STA->val_tei[i]==0 &&
                    NT_STA->val_cr[i]==0 )
                        break;
        }
        if( i < 64 ) {
                NT_STA->val_send[i] = send;
                NT_STA->val_tei[i] = tei;
                NT_STA->val_cr[i] = cr;
                NT_STA->val_cid[i] = cid;
                return(cid);
        }
        else
                error("NT_STA->val overflow");
}

error(fmt,x1,x2,x3,x4)
char *fmt;
int x1,x2,x3,x4;
{
        printf("out1 error: ");
        printf(fmt,x1,x2,x3,x4);
        printf("0);
/*      sys(8,0);       */
}
```

```
/*
 *
 *  Definition of some external variable values
 *
 *
 *
 */


#define         MAX_CID         10
#define         MAX_PORT        16
#define         MAX_CRT         257
#define         MAX_TS          65
#define         ITRFC           3
#define         B_CH            3
#define         MAX_CRV         257
#define         B_ID            3
#define         ETAS            5
#define         SNDR            4

#define         ST_NULL         0
#define         ST_CLPR         1
#define         ST_ALER         2
#define         ST_ACTIVE       3
#define         ST_DISCRQ       4
#define         ST_DISC         5
#define         ST_RELIND       6
#define         ST_RELCNF       7
#define         ST_RLCP         8
#define         ST_CLEAR        9

#define         st_null         0
#define         st_calprc       1
#define         st_alert        2
#define         st_active       3
#define         st_discrq       4
#define         st_disc         5
#define         st_relcpl       6

#define         FREE            0
#define         OCC             1
```

```
/* Implementation of a proposed call data structure used in model;      */
/* Further resources and status data structure used in the NT12.        */

struct  ODB     {
        int     ts_O;                   /*      1-64    */
        int     CR_O;                   /*      1-128   */
        int     TEI_O;                  /*      11-15   */
        int     port;                   /*      1-20    */
        int     INPID;
        int     OBIDS;                  /* L2 pid  0-2  */
        int     OBIDR;                  /* L2 pid  5-7  */
        char    B_CH_O;                 /* 1=B1, 2=B2   */
        char    ITRF_O;                 /* 0=S1,1=S2,2=U*/
};

struct  TDB     {
        int     tid;                    /*    101-104   */
        int     ts_T;                   /*      1-64    */
        int     CR_T;                   /*      129-256 */
        int     TEI;                    /*      11-15   */
        int     poro;                   /*      51-70   */
        int     OUTPID;
        int     TBIDS;                  /* L2 pid  0-2  */
        int     TBIDR;                  /* L2 pid  5-7  */
        char    T_state;
        char    act_tei;
        char    B_CH_T;                 /* 1=B1, 2=B2   */
        char    ITRF_T;                 /* 0=S1,1=S2,2=U*/
};

struct  CINFO   {
        char    sw;                     /* TRUE, FALSE  */
        char    state;
        int     CCPID;
        struct  ODB     oinfo;
        struct  TDB     tinfo;
};

struct resources {
        char    cid[MAX_CID];                   /* 0=FREE, 1=OCC        */
        char    cr_t[MAX_CRT];                  /* 0=FREE, 1=OCC        */
        char    ts[MAX_TS];                     /* 0=FREE, 1=OCC        */
        char    ch[ITRFC][B_CH];                /* 0=FREE, 1=OCC        */
        char    arbport[21];                    /* 0=FREE, 1=OCC        */
};

struct  status {
        int     n_acca;                         /* n active call        */
        int     ter_cr[MAX_CRV];                /* =CID, for ie respond */
                                                /* max_crt=128, aantal  */
                                                /* CID  1-100           */
        int     inpid[MAX_CID];                 /* =INPID for certain CID */
                                                /* max_cid=200, aantal  */
        int     outpid[MAX_CID];                /* =OUTPID for certain CID */
        int     ccpid[MAX_CID];                 /* =CCPID for certain CID */
        int     b_ch[ITRFC][B_ID][MAX_CID];     /* = 0, channel available */
                                                /* ITRFC 0-2            */
                                                /* b_id  1-2            */
                                                /* else res.ts# is recorded */
        int     val_send[64];
        int     val_tei[64];
        int     val_cr[64];
        int     val_cid[64];

        int     val_inf[5];                     /* val_inf[1]=val_inf[2]=0      */
                                                /* val_inf[3]=val_inf[4]=1      */
        int     val_tid[5];
        int     teiass[ITRFC][ETAS];            /* =fixed mapped TEI    */
        int     arbpid;
};
```

```
/*
 *
 *
 *  signal table:
 *  signal namesused in the program, and amount of parameters
 *
 *
 */
struct sigtab {
        char *sig_name;
        char *sig_parm;
} sigtab[] = {
{ "DUMMY",              "" },           /* 0 */
{ "SETUP",              "iiiiii" },     /* 1 */
{ "CONNECT_ACK",        "iii" },        /* 2 */
{ "DISC",               "iii" },        /* 3 */
{ "RELEASE",            "iii" },        /* 4 */
{ "REL_CPL",            "iii" },        /* 5 */
{ "ALERTING",           "iiii" },       /* 6 */
{ "CALL_PROCEED",       "iiii" },       /* 7 */
{ "CONNECT",            "iiii" },       /* 8 */
{ "CMD",                "si" },         /* 9 */
{ "iSETUP",             "iii" },        /* 10 */
{ "iDISC",              "i" },          /* 11 */
{ "iRELEASE",           "i" },          /* 12 */
{ "iREL_CPL",           "i" },          /* 13 */
{ "iCONNECT_ACK",       "i" },          /* 14 */
{ "oREL_CPL",           "ii" },         /* 15 */
{ "oALERTING",          "ii" },         /* 16 */
{ "oCALL_PROCEED",      "ii" },         /* 17 */
{ "oCONNECT",           "ii" },         /* 18 */
{ "oDISC",              "ii" },         /* 19 */
{ "oRELEASE",           "ii" },         /* 20 */
{ "SET_RESP",           "i" },          /* 21 */
{ "DISC_REQ",           "i" },          /* 22 */
{ "REL_RESP",           "i" },          /* 23 */
{ "REL_REQ",            "i" },          /* 24 */
{ "REJ_REQ",            "i" },          /* 25 */
{ "CLEAR_REQ",          "i" },          /* 26 */
{ "ALERT_REQ",          "i" },          /* 27 */
{ "CALL_P_REQ",         "ii" },         /* 28 */
{ "SET_IND",            "iiii" },       /* 29 */
{ "DISC_IND",           "i" },          /* 30 */
{ "REL_IND",            "i" },          /* 31 */
{ "REL_CNF",            "i" },          /* 32 */
{ "SET_REQ",            "iii" },        /* 33 */
{ "",                   "" },           /* 34 */
{ "",                   "" },           /* 35 */
{ "",                   "" },           /* 36 */
{ "",                   "" },           /* 37 */
{ "SET_CNF",            "ii" },         /* 38 */
{ "ALERT_IND",          "ii" },         /* 39 */
{ "CALL_P_IND",         "ii" },         /* 40 */
{ "CLEAR_IND",          "i" },          /* 41 */
{ "REJ_IND",            "i" },          /* 42 */
{ "SW_ON",              "iiiiii" },     /* 43 */
{ "SW_OFF",             "iiiiii" },     /* 44 */
{ (char *) 0,           ""      },
};
```

_APPENDIX_E:_THE_TEST_MESSAGE_SEQUENCES_

```
/*
 *
 *      test sequence generator
 *
 *
 */

#include "b:sys.h"
#include "b:msg.h"
#include "b:sigtab.h"

#define NULL ((char *)0)

/**** TEST 0 ********************************************************/
/*       tid 101 chan 1      connect       tid 102 chan 2             */
/*       tid 101 chan 1      connect       tid 103 chan 1             */
/*       tid 101 chan 2      connect       tid 104 chan 1             */
/*       tid 102 chan 2      connect       tid 104 chan 1             */

char *test0[] = {

"send 0 SETUP 1 1 0 0 1 102" ,
                                "send 0 ALERTING 2 129 1 1",
                                "send 0 CONNECT  2 129 1 1",
"send 0 CONNECT_ACK 1 1 0",
"send 0 DISC 1 1 0",
"send 0 REL_CPL 1 1 0",
                                "send 0 RELEASE  2 129 1",


"send 0 SETUP 1 2 0 0 1 103",
                                "send 0 ALERTING 3 129 1 1",
                                "send 0 CONNECT  3 129 1 1",
"send 0 CONNECT_ACK 1 2 0",
                                "send 0 DISC 3 129 1",
"send 0 RELEASE 1 2 0",
                                "send 0 REL_CPL 3 129 1",

        }


"send 0 SETUP 1 3 0 22 1 104",
                                "send 0 CALL_PROCEED 4 129 1 1",
                                "send 0 ALERTING 4 129 1 1",
                                "send 0 CONNECT  4 129 1 1",
"send 0 CONNECT_ACK 1 3 0",
"send 0 DISC 1 3 0",
                                "send 0 RELEASE  4 129 1",
"send 0 REL_CPL 1 3 0",


"send 0 SETUP 2 4 0 22 1 104" ,
                                "send 0 ALERTING 2 129 1 1",
                                "send 0 ALERTING 4 129 1 1",
                                "send 0 CONNECT  4 129 1 1",
"send 0 CONNECT_ACK 2 4 0",
"send 0 DISC 2 4 0",
"send 0 REL_CPL 2 4 0",
                                "send 0 RELEASE  4 129 1",
"",
};

/**** TEST 1 ********************************************************/
/*   tid 101 ch 1 - tid 102 ch 2,  tid 103 ch 1 - tid 104 ch 2      */
/*   tid 101 ch 1 - tid 103 ch 1,  tid 102 ch 2 - tid 104 ch 2      */

char *test1[] = {
"send 0 SETUP 1 11 0 0 1 102" ,
                                "send 0 ALERTING 2 129 1 1",
                                "send 0 CONNECT  2 129 1 1",
"send 0 CONNECT_ACK 1 11 0",
"send 0 SETUP 3 12 0 21 1 104" ,
                                "send 0 CONNECT  4 130 1 2",
"send 0 CONNECT_ACK 3 12 0",


"send 0 DISC 1 11 0",
"send 0 REL_CPL 1 11 0",
                                "send 0 RELEASE  2 129 1",
```

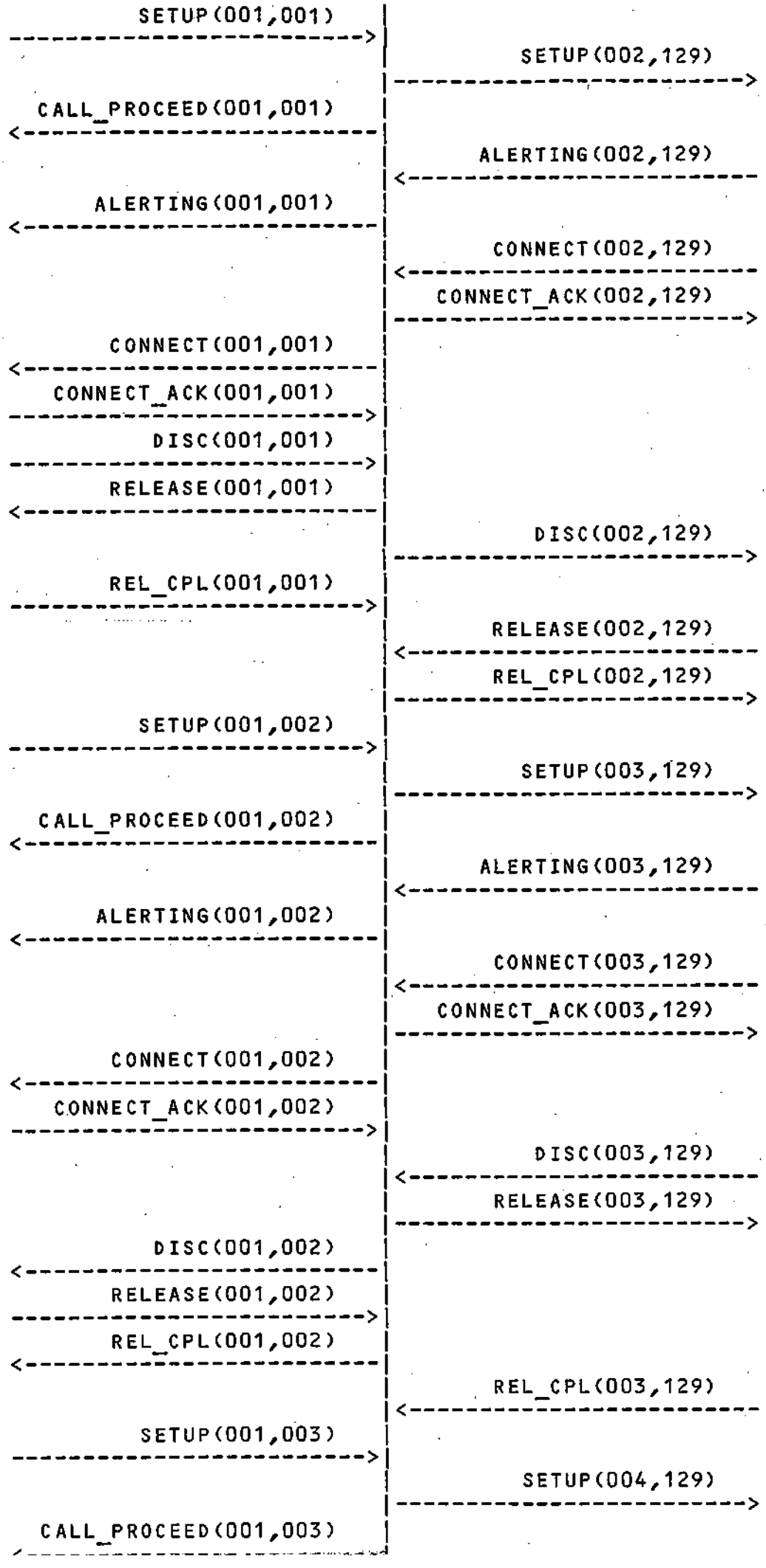
```
"send 0 DISC 3 12 0",
"send 0 REL_CPL 3 12 0",
                                "send 0 RELEASE  4 130 1",

"send 0 SETUP 1 11 0 0 1 103",
                                "send 0 CONNECT  3 129 1 1",
"send 0 CONNECT_ACK 1 11 0",
"send 0 SETUP 2 12 0 22 1 104",
                                "send 0 CONNECT  4 130 1 2",
"send 0 CONNECT_ACK 2 12 0",

"send 0 DISC 2 12 0",
"send 0 REL_CPL 2 12 0",
                                "send 0 RELEASE  4 130 1",
                                "send 0 DISC 3 129 1",
"send 0 RELEASE 1 11 0",
                                "send 0 REL_CPL 3 129 1",

"",
};
/***********   testgen 2   ******************/
/*   tid 101 ch 2 - tid 104 ch 1,  tid 102 ch 1 - tid 103 ch 2        */
/*   tid 102 ch 2 - tid 104 ch 1,  tid 101 ch 1 - tid 103 ch 2        */

char *test2[] = {

"send 0 SETUP 1 13 0 22 1 104",
                                "send 0 CALL_PROCEED 4 129 1 1",
                                "send 0 ALERTING 4 129 1 1",
"send 0 SETUP 2 16 0 12 1 104",
"send 0 SETUP 2 17 0 12 1 103",
                                "send 0 CONNECT  4 129 1 1",
                                "send 0 ALERTING 3 130 1 2",
"send 0 CONNECT_ACK 1 13 0",
                                "send 0 CONNECT  3 130 1 2",
"send 0 CONNECT_ACK 2 17 0",


"send 0 DISC 1 13 0",
                                "send 0 RELEASE  4 129 1",
"send 0 DISC 2 17 0",
"send 0 REL_CPL 1 13 0",
                                "send 0 RELEASE  3 130 1",
"send 0 REL_CPL 2 17 0",


"send 0 SETUP 2 14 0 12 1 104" ,
                                "send 0 ALERTING 2 129 1 1",
                                "send 0 ALERTING 4 129 1 1",
                                "send 0 CONNECT  4 129 1 1",
"send 0 CONNECT_ACK 2 14 0",
"send 0 SETUP 1 17 0 12 1 104",
"send 0 SETUP 1 18 0 12 1 103" ,
                                "send 0 ALERTING 3 130 1 2",
                                "send 0 CONNECT  3 130 1 2",
"send 0 CONNECT_ACK 1 18 0",



"send 0 DISC 2 14 0",
                                "send 0 DISC 3 130 1",
"send 0 REL_CPL 2 14 0",
"send 0 RELEASE 1 18 0",
                                "send 0 RELEASE 4 129 1",
                                "send 0 REL_CPL 3 130 1",

"",
};

/**** TEST 3*************************************************************/
/************* test release procedure of active call  *****************/

char *test3[] = {
"send 0 SETUP 1 13 0 22 1 104",
                                "send 0 ALERTING 4 129 1 1",
"send 0 SETUP 2 17 0 12 1 103",
                                "send 0 CONNECT  4 129 1 1",
"send 0 CONNECT_ACK 1 13 0",
```

```
                                "send O CONNECT  3 130 1 2",
"send O CONNECT_ACK 2 17 O",
"send O SETUP 1 15 O O 1 103",

"send O DISC 1 13 O ",
"send O REL_CPL 1 13 O ",
                                "send O RELEASE 4 129 1 ",

                                "send O DISC 3 130 1 ",
                                "send O REL_CPL 3 130 1 ",
"send O RELEASE 2 17 O ",


"send O SETUP 1 14 O 22 1 104",
                                "send O ALERTING 4 129 1 1",
"send O SETUP 2 18 O 12 1 103",
                                "send O CONNECT  4 129 1 1",
"send O CONNECT_ACK 1 14 O",
                                "send O CONNECT  3 130 1 2",
"send O CONNECT_ACK 2 18 O",


"send O DISC 1 14 O ",
                                "send O DISC 3 130 1 ",
                                "send O RELEASE 4 129 1 ",
                                "send O REL_CPL 3 130 1 ",
"send O REL_CPL 1 14 O ",
"send O RELEASE 2 18 O ",
"",
};

/*** TEST 4*************************************************************/
/********  test release procedure of call which are not active yet  *****/
char *test4[] = {
"send O SETUP 1 13 O 22 1 104",
                                "send O ALERTING 4 128 1 1",
"send O SETUP 2 17 O 12 1 103",
                                "send O CALL_PROCEED 3 130 1 2",
                                "send O ALERTING 3 130 1 2",
"send O DISC 1 13 O ",
                                "send O RELEASE 4 129 1 ",
                                "send O DISC 3 130 1 ",
"send O REL_CPL 1 13 O ",
                                "send O REL_CPL 3 130 1 ",
"send O RELEASE 2 17 O ",
"send O SETUP 1 18 O 22 1 104",
"send O SETUP 3 19 O 12 1 102",
                                "send O CONNECT 4 129 1 1",
                                "send O CONNECT 2 130 1 1",

"send O DISC 1 18 O ",
                                "send O DISC 2 130 1 ",
                                "send O RELEASE 4 129 1 ",

"send O REL_CPL 1 18 O ",
                                "send O REL_CPL 2 130 1 ",
"send O RELEASE 3 19 O ",

"",
};
/*** TEST 5*************************************************************/
/* test the release procedure  */

char *test5[] = {
"send O SETUP 1 13 O 22 1 104",
                                "send O ALERTING 4 129 1 1",
"send O SETUP 2 17 O 12 1 103",
                                "send O CALL_PROCEED 3 130 1 2",
                                "send O ALERTING 3 130 1 2",
"send O DISC 1 13 O ",
                                "send O RELEASE 4 129 1 ",
                                "send O DISC 3 130 1 ",
"send O REL_CPL 1 13 O ",
                                "send O REL_CPL 3 130 1 ",
"send O RELEASE 2 17 O ",


"send O SETUP 1 1 O O 1 104",
```

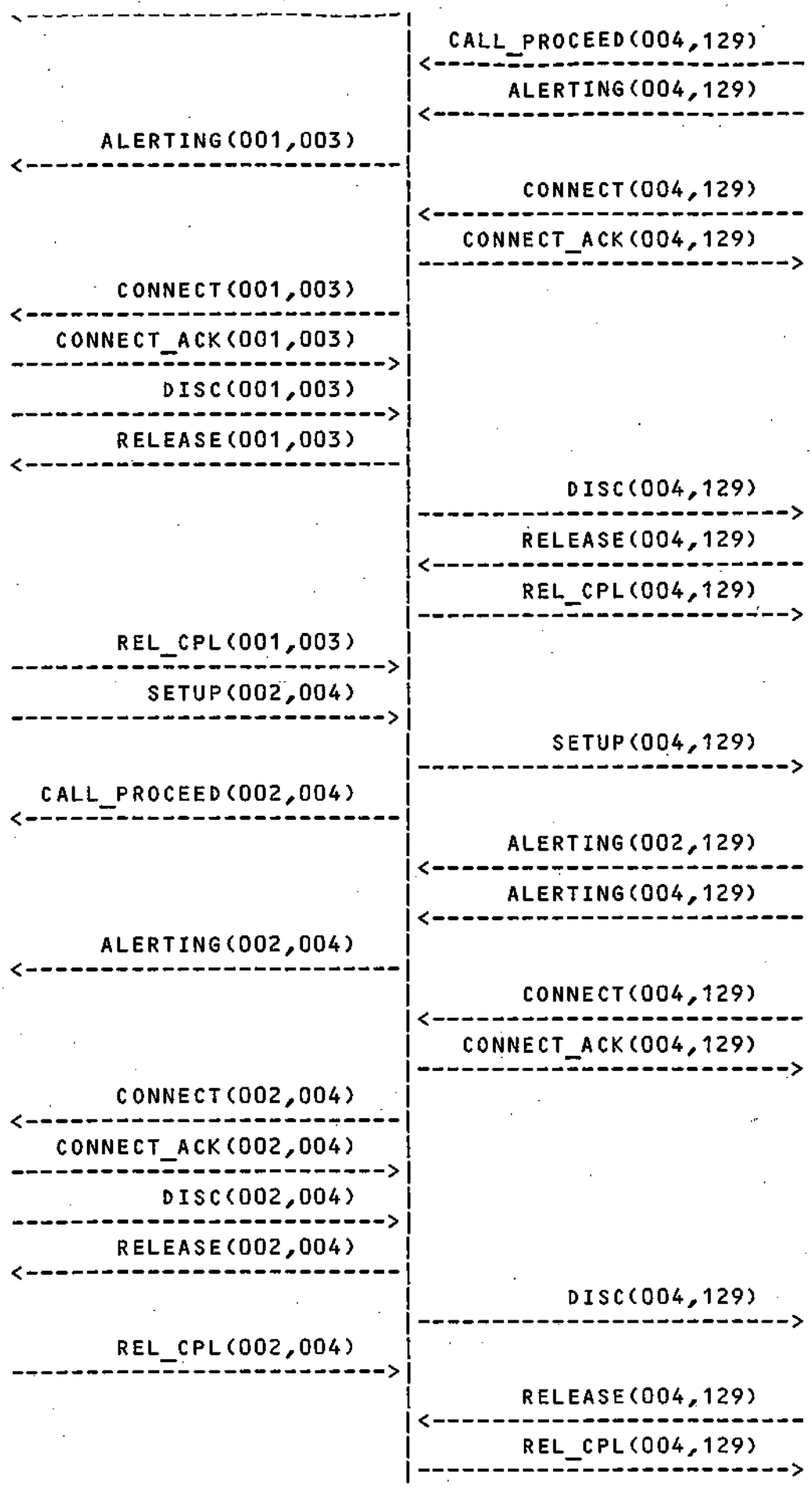
```
                                        "send 0 REL_CPL 4 129 1",
"send 0 RELEASE 1 1 0",
"",
};


char **test[] = {
test0,
test1,
test2,
test3,
test4,
test5,
};

main(argc,argv)
int argc;
char *argv[];
{
        char    *av[32];        /* pointer array to arguments(argv) */
        int     ac;             /* argument count (argc) */
        char **tst;
        int nr;
        int i;

        if( argc != 2 ) {
                printf("usage: testgen 'nr'0);
                return;
        }
        nr = atoi(argv[1]);
        if( nr<0 || nr>=(sizeof(test)/sizeof(test[0])) ) {
                printf("testgen: test nr out of range0);
                return;
        }
        tst = test[nr];
        for(i=0; *tst[i]; i++) {
                printf("%s0,tst[i]);
                ac = getcmd( av, tst[i] );
                send(ac,av);
                while( getchar()!='0 );
        }
}

send(argc,argv)
int argc;
char *argv[];
{
        struct mbuf mbuf;
        struct sigtab *st;
        int n,i,size,port;
        char c,*p,*q,*fmt;
        char line[128];
        int parm[12];

        if( argc < 3 ) {
                printf("usage: send 'port' 'msg_id' ['parms']0);
                return;
        }
        port = atoi(argv[1]);
        for(st=sigtab; st->sig_name; st++)
                if(strcmp(argv[2],st->sig_name)==0) break;
        if( !st->sig_name ) {
                printf("signal name not found0);
                return;
        }
        n = 0;
        for(fmt=st->sig_parm;*fmt;fmt++) {
                if( n+3 >= argc ) {
                        printf("argument(s) missing0);
                        printf("format = %s0,st->sig_parm);
                        return;
                }
                switch(*fmt) {
                        case 'i': parm[n] = atoi(argv[n+3]); break;
                        case 's': parm[n] = (int)argv[n+3]; break;
                        case 'c': parm[n] = (int)argv[n+3][0]; break;
/* changed for demo:    case '*': parm[n] = atoi(argv[n+3]); break;     */
```

```
                        case '*': parm[n] = strlen(argv[n+3]);
                                argv--;
                                argc++;
                                break;
                        case 'b': parm[n] = (int)argv[n+3]; break;
                }
                n++;
        }
        txmsg(&mbuf,port,st-sigtab,st->sig_parm,parm[0],parm[1],parm[2],
                parm[3],parm[4],parm[5],parm[6],parm[7]);
}

char    txt1[256];      /* argument buffer */

getcmd(av,cmdbuf)
char    *av[];
char *cmdbuf;
{
        int     ac;
        char    c,*p,*bp,qflag,aflag,quote ;
        av[0] = (char *)0 ;
        ac = 0 ;
        p = txt1 ;
        qflag = aflag = 0 ;
        bp = cmdbuf;
        while ((c = *bp++) != 0) {
                switch (c) {
                case ' ':
                        if (qflag) *p++ = c ;
                        else if (aflag) *p++ = aflag = 0 ;
                        break ;
                case '
                case ''':
                        quote = c ;
                        if (aflag==0) {
                                av[ac++] = p ;
                                aflag = 1 ;
                        }
                        while ((c = *bp++)!=quote) {
                                *p++ = c ;
                                if (c=='') p[-1] = (c = *bp++) ;
                                if (c=='0) { p-- ; }
                        }
                        break ;
                case '':
                        *p++ = (c = *bp++) ;
                        if (c=='0) { p-- ; }
                        break ;
                default:
                        if (aflag==0) {
                                av[ac++] = p ;
                                aflag = 1 ;
                        }
                        *p++ = c ;
                        break ;
                }
        }
        *p++ = 0 ;
        av[ac] = (char *) 0 ;
        return(ac);
}
```

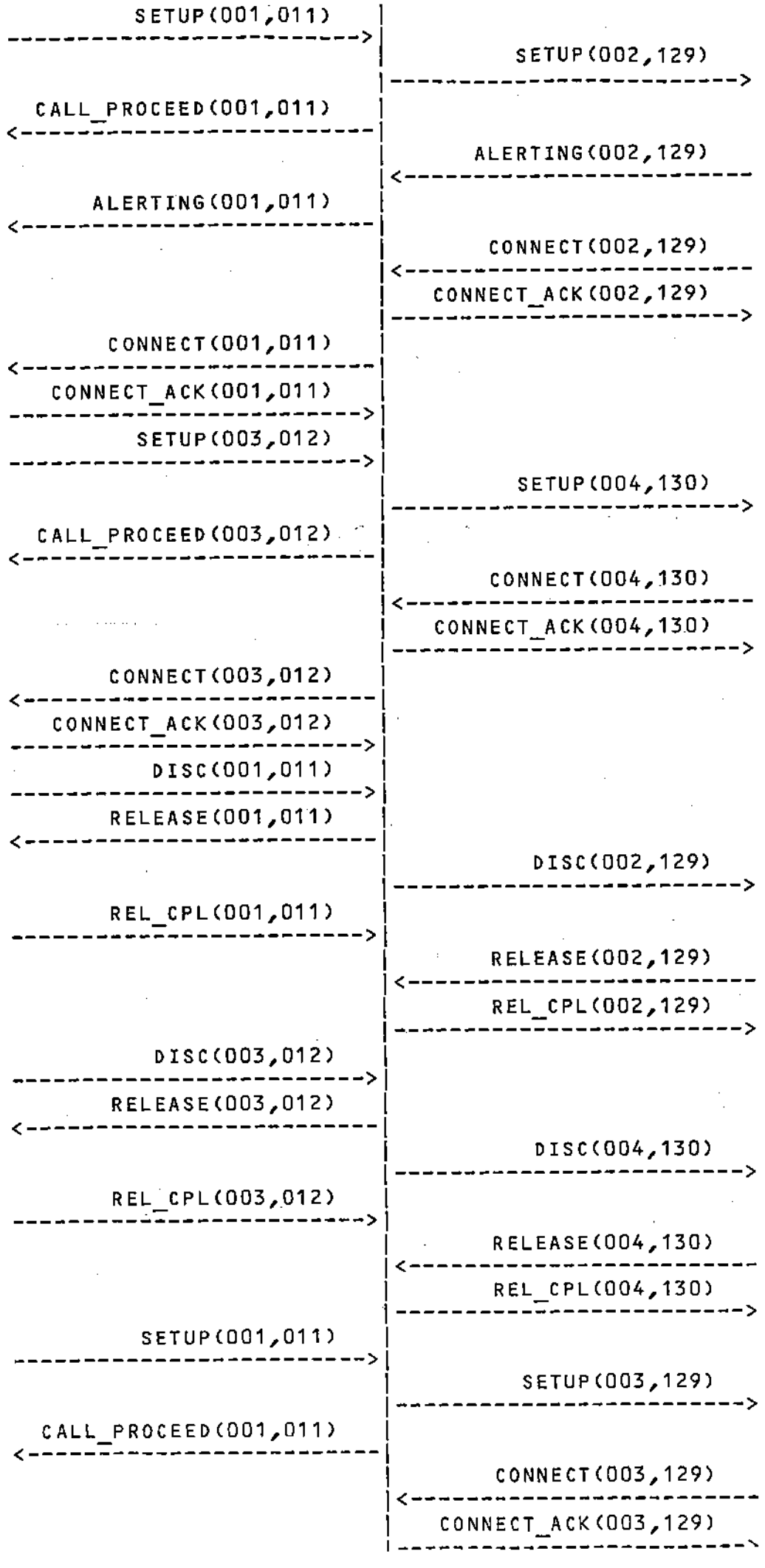
```
  testgen 0
send 0 SETUP 1 1 0 0 1 102
CID for new call: 1
env[0]:SETUP 2 129 0 1 1 102
env[0]:CALL_PROCEED 1 1 1 2

send 0 ALERTING 2 129 1 1
env[0]:SW_ON 2 0 1 1 0 2
env[0]:ALERTING 1 1 1 2

send 0 CONNECT  2 129 1 1
env[0]:CONNECT_ACK 2 129 0
env[0]:CONNECT 1 1 1 2

send 0 CONNECT_ACK 1 1 0

send 0 DISC 1 1 0
env[0]:RELEASE 1 1 1
env[0]:DISC 2 129 0

send 0 REL_CPL 1 1 0
env[0]:SW_OFF 2 0 1 1 0 2

send 0 RELEASE  2 129 1
env[0]:REL_CPL 2 129 0

send 0 SETUP 1 2 0 0 1 103
CID for new call: 1
env[0]:SETUP 3 129 0 1 1 103
env[0]:CALL_PROCEED 1 2 1 1

send 0 ALERTING 3 129 1 1
env[0]:SW_ON 1 0 1 1 1 2
env[0]:ALERTING 1 2 1 1

send 0 CONNECT  3 129 1 1
env[0]:CONNECT_ACK 3 129 0
env[0]:CONNECT 1 2 1 1

send 0 CONNECT_ACK 1 2 0

send 0 DISC 3 129 1
env[0]:RELEASE 3 129 0
env[0]:DISC 1 2 1

send 0 RELEASE 1 2 0
env[0]:REL_CPL 1 2 1

send 0 REL_CPL 3 129 1
env[0]:SW_OFF 1 0 1 1 1 2

send 0 SETUP 1 3 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 3 1 2

send 0 CALL_PROCEED 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2

send 0 ALERTING 4 129 1 1
```


SETUP(001,001)
SETUP(002,129)
CALL_PROCEED(001,001)
ALERTING(002,129)
ALERTING(001,001)
CONNECT(002,129)
CONNECT_ACK(002,129)
CONNECT(001,001)
CONNECT_ACK(001,001)
DISC(001,001)
RELEASE(001,001)
DISC(002,129)
REL_CPL(001,001)
RELEASE(002,129)
REL_CPL(002,129)
SETUP(001,002)
SETUP(003,129)
CALL_PROCEED(001,002)
ALERTING(003,129)
ALERTING(001,002)
CONNECT(003,129)
CONNECT_ACK(003,129)
CONNECT(001,002)
CONNECT_ACK(001,002)
DISC(003,129)
RELEASE(003,129)
DISC(001,002)
RELEASE(001,002)
REL_CPL(001,002)
REL_CPL(003,129)
SETUP(001,003)
SETUP(004,129)
CALL_PROCEED(001,003)

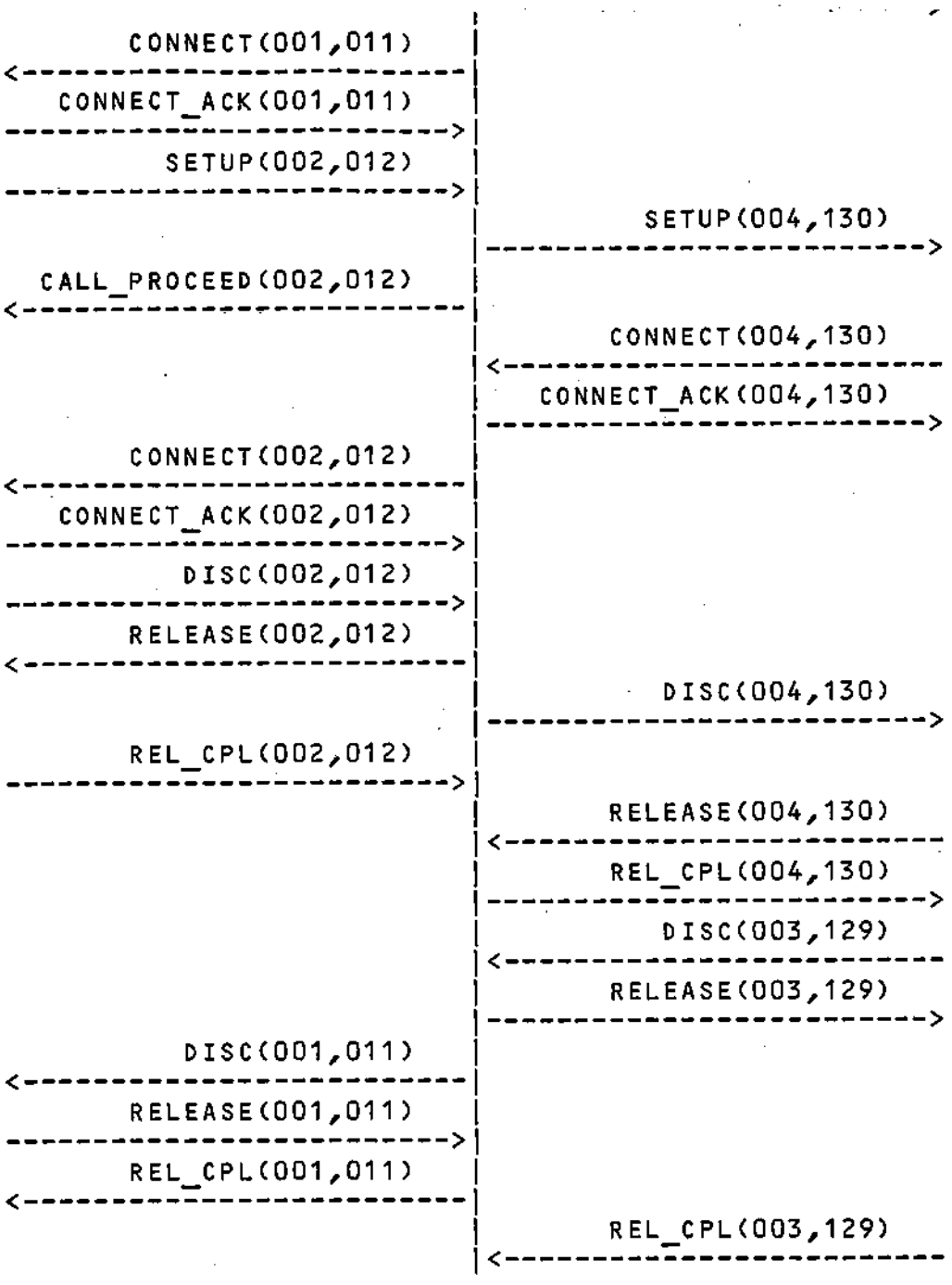
```
env[0]:ALERTING 1 3 1 2

send 0 CONNECT  4 129 1 1
env[0]:CONNECT_ACK 4 129 0
env[0]:CONNECT 1 3 1 2

send 0 CONNECT_ACK 1 3 0

send 0 DISC 1 3 0
env[0]:RELEASE 1 3 1
env[0]:DISC 4 129 0

send 0 RELEASE  4 129 1
env[0]:REL_CPL 4 129 0

send 0 REL_CPL 1 3 0
env[0]:SW_OFF 2 0 1 1 1 2

send 0 SETUP 2 4 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 2 4 1 2

send 0 ALERTING 2 129 1 1
out1 error: TEI in msg not expected

send 0 ALERTING 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2
env[0]:ALERTING 2 4 1 2

send 0 CONNECT  4 129 1 1
env[0]:CONNECT_ACK 4 129 0
env[0]:CONNECT 2 4 1 2

send 0 CONNECT_ACK 2 4 0

send 0 DISC 2 4 0
env[0]:RELEASE 2 4 1
env[0]:DISC 4 129 0

send 0 REL_CPL 2 4 0
env[0]:SW_OFF 2 0 1 1 1 2

send 0 RELEASE  4 129 1
env[0]:REL_CPL 4 129 0

>>
```


CALL_PROCEED(004,129)
ALERTING(004,129)
ALERTING(001,003)
CONNECT(004,129)
CONNECT_ACK(004,129)
CONNECT(001,003)
CONNECT_ACK(001,003)
DISC(001,003)
RELEASE(001,003)
DISC(004,129)
RELEASE(004,129)
REL_CPL(004,129)
REL_CPL(001,003)
SETUP(002,004)
SETUP(004,129)
CALL_PROCEED(002,004)
ALERTING(002,129)
ALERTING(004,129)
ALERTING(002,004)
CONNECT(004,129)
CONNECT_ACK(004,129)
CONNECT(002,004)
CONNECT_ACK(002,004)
DISC(002,004)
RELEASE(002,004)
DISC(004,129)
REL_CPL(002,004)
RELEASE(004,129)
REL_CPL(004,129)

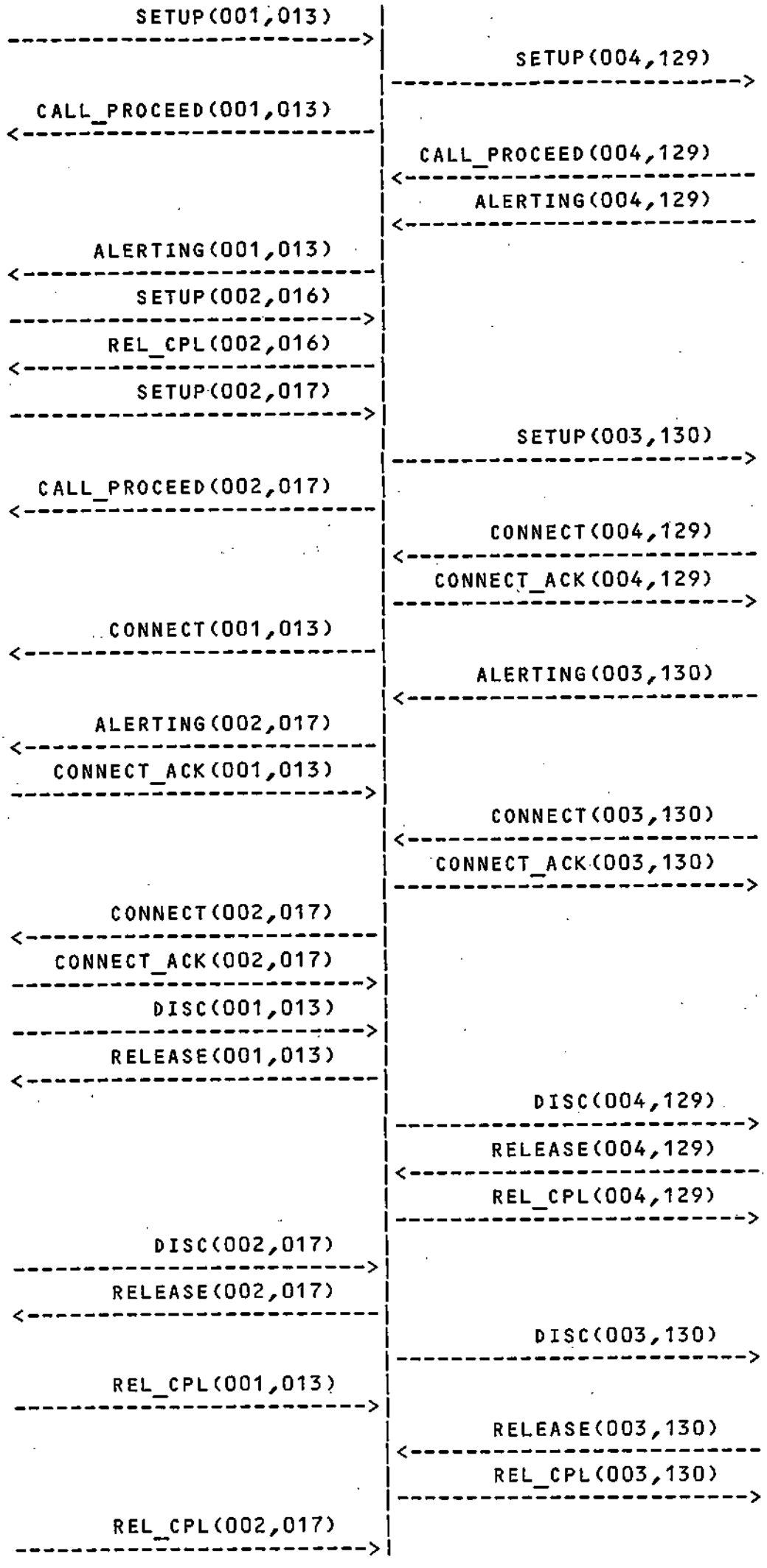
```
  testgen 1
send 0 SETUP 1 11 0 0 1 102
CID for new call: 1
env[0]:SETUP 2 129 0 1 1 102
env[0]:CALL_PROCEED 1 11 1 2

send 0 ALERTING 2 129 1 1
env[0]:SW_ON 2 0 1 1 0 2
env[0]:ALERTING 1 11 1 2

send 0 CONNECT  2 129 1 1
env[0]:CONNECT_ACK 2 129 0
env[0]:CONNECT 1 11 1 2

send 0 CONNECT_ACK 1 11 0

send 0 SETUP 3 12 0 21 1 104
CID for new call: 2
env[0]:SETUP 4 130 0 2 1 104
env[0]:CALL_PROCEED 3 12 1 1

send 0 CONNECT  4 130 1 2
env[0]:CONNECT_ACK 4 130 0
env[0]:SW_ON 1 1 3 2 1 4
env[0]:CONNECT 3 12 1 1

send 0 CONNECT_ACK 3 12 0

send 0 DISC 1 11 0
env[0]:RELEASE 1 11 1
env[0]:DISC 2 129 0

send 0 REL_CPL 1 11 0
env[0]:SW_OFF 2 0 1 1 0 2

send 0 RELEASE  2 129 1
env[0]:REL_CPL 2 129 0

send 0 DISC 3 12 0
env[0]:RELEASE 3 12 1
env[0]:DISC 4 130 0

send 0 REL_CPL 3 12 0
env[0]:SW_OFF 1 1 3 2 1 4

send 0 RELEASE  4 130 1
env[0]:REL_CPL 4 130 0

send 0 SETUP 1 11 0 0 1 103
CID for new call: 1
env[0]:SETUP 3 129 0 1 1 103
env[0]:CALL_PROCEED 1 11 1 1

send 0 CONNECT  3 129 1 1
env[0]:CONNECT_ACK 3 129 0
env[0]:SW_ON 1 0 1 1 1 2
env[0]:CONNECT 1 11 1 1

send 0 CONNECT_ACK 1 11 0

send 0 SETUP 2 12 0 22 1 104
CID for new call: 2
env[0]:SETUP 4 130 0 2 1 104
env[0]:CALL_PROCEED 2 12 1 2

send 0 CONNECT  4 130 1 2
```


SETUP(001,011)
SETUP(002,129)
CALL_PROCEED(001,011)
ALERTING(002,129)
ALERTING(001,011)
CONNECT(002,129)
CONNECT_ACK(002,129)
CONNECT(001,011)
CONNECT_ACK(001,011)
SETUP(003,012)
SETUP(004,130)
CALL_PROCEED(003,012)
CONNECT(004,130)
CONNECT_ACK(004,130)
CONNECT(003,012)
CONNECT_ACK(003,012)
DISC(001,011)
RELEASE(001,011)
DISC(002,129)
REL_CPL(001,011)
RELEASE(002,129)
REL_CPL(002,129)
DISC(003,012)
RELEASE(003,012)
DISC(004,130)
REL_CPL(003,012)
RELEASE(004,130)
REL_CPL(004,130)
SETUP(001,011)
SETUP(003,129)
CALL_PROCEED(001,011)
CONNECT(003,129)
CONNECT_ACK(003,129)

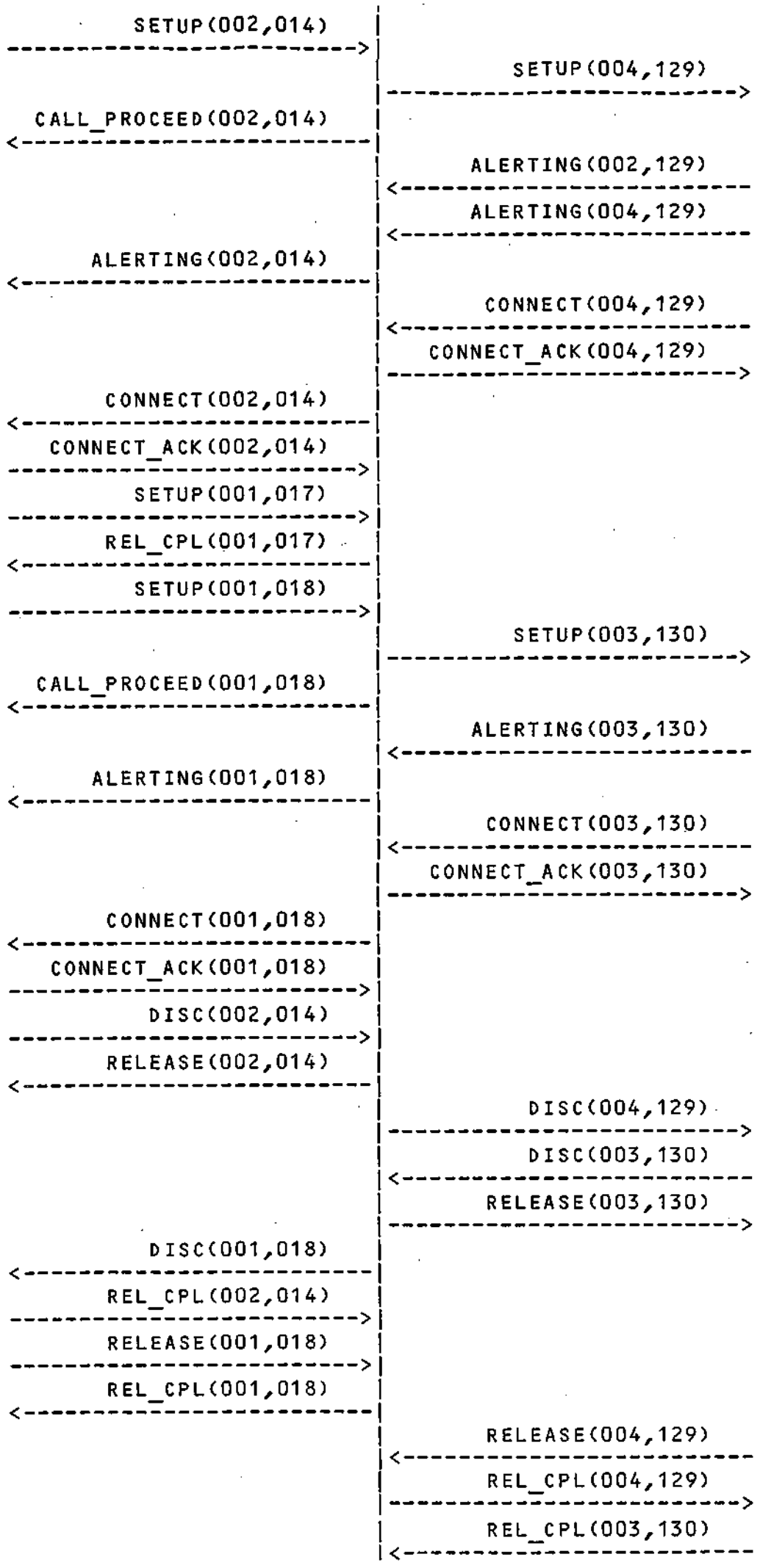
```
env[0]:CONNECT_ACK 4 130 0
env[0]:SW_ON 2 0 3 2 1 4
env[0]:CONNECT 2 12 1 2

send 0 CONNECT_ACK 2 12 0

send 0 DISC 2 12 0
env[0]:RELEASE 2 12 1
env[0]:DISC 4 130 0

send 0 REL_CPL 2 12 0
env[0]:SW_OFF 2 0 3 2 1 4

send 0 RELEASE  4 130 1
env[0]:REL_CPL 4 130 0

send 0 DISC 3 129 1
env[0]:RELEASE 3 129 0
env[0]:DISC 1 11 1

send 0 RELEASE 1 11 0
env[0]:REL_CPL 1 11 1

send 0 REL_CPL 3 129 1
env[0]:SW_OFF 1 0 1 1 1 2

>>
```


CONNECT(001,011)
CONNECT_ACK(001,011)
SETUP(002,012)
SETUP(004,130)
CALL_PROCEED(002,012)
CONNECT(004,130)
CONNECT_ACK(004,130)
CONNECT(002,012)
CONNECT_ACK(002,012)
DISC(002,012)
RELEASE(002,012)
DISC(004,130)
REL_CPL(002,012)
RELEASE(004,130)
REL_CPL(004,130)
DISC(003,129)
RELEASE(003,129)
DISC(001,011)
RELEASE(001,011)
REL_CPL(001,011)
REL_CPL(003,129)

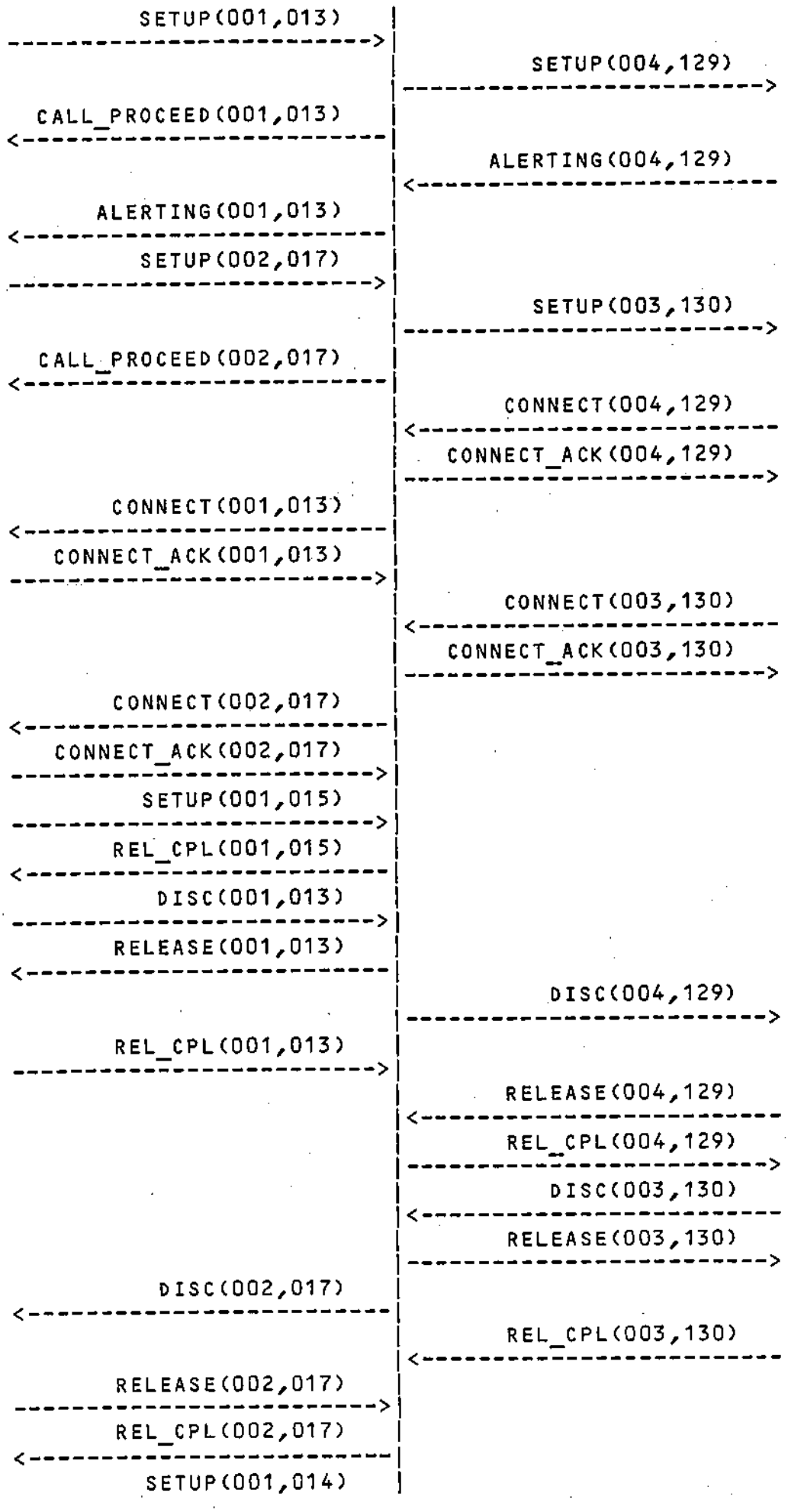
```
  testgen 2
send 0 SETUP 1 13 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 13 1 2

send 0 CALL_PROCEED 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2

send 0 ALERTING 4 129 1 1
env[0]:ALERTING 1 13 1 2

send 0 SETUP 2 16 0 12 1 104
CID for new call: 2
cc1 error: xnr 104 busy
env[0]:REL_CPL 2 16 1

send 0 SETUP 2 17 0 12 1 103
CID for new call: 2
env[0]:SETUP 3 130 0 2 1 103
env[0]:CALL_PROCEED 2 17 1 1

send 0 CONNECT  4 129 1 1
env[0]:CONNECT_ACK 4 129 0
env[0]:CONNECT 1 13 1 2

send 0 ALERTING 3 130 1 2
env[0]:SW_ON 1 0 3 2 1 4
env[0]:ALERTING 2 17 1 1

send 0 CONNECT_ACK 1 13 0

send 0 CONNECT  3 130 1 2
env[0]:CONNECT_ACK 3 130 0
env[0]:CONNECT 2 17 1 1

send 0 CONNECT_ACK 2 17 0

send 0 DISC 1 13 0
env[0]:RELEASE 1 13 1
env[0]:DISC 4 129 0

send 0 RELEASE  4 129 1
env[0]:REL_CPL 4 129 0

send 0 DISC 2 17 0
env[0]:RELEASE 2 17 1
env[0]:DISC 3 130 0

send 0 REL_CPL 1 13 0
env[0]:SW_OFF 2 0 1 1 1 2

send 0 RELEASE  3 130 1
env[0]:REL_CPL 3 130 0

send 0 REL_CPL 2 17 0
env[0]:SW_OFF 1 0 3 2 1 4

send 0 SETUP 2 14 0 12 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 2 14 1 2

send 0 ALERTING 2 129 1 1
out1 error: TEI in msg not expected
```


SETUP(001,013)
SETUP(004,129)
CALL_PROCEED(001,013)
CALL_PROCEED(004,129)
ALERTING(004,129)
ALERTING(001,013)
SETUP(002,016)
REL_CPL(002,016)
SETUP(002,017)
SETUP(003,130)
CALL_PROCEED(002,017)
CONNECT(004,129)
CONNECT_ACK(004,129)
CONNECT(001,013)
ALERTING(003,130)
ALERTING(002,017)
CONNECT_ACK(001,013)
CONNECT(003,130)
CONNECT_ACK(003,130)
CONNECT(002,017)
CONNECT_ACK(002,017)
DISC(001,013)
RELEASE(001,013)
DISC(004,129)
RELEASE(004,129)
REL_CPL(004,129)
DISC(002,017)
RELEASE(002,017)
DISC(003,130)
REL_CPL(001,013)
RELEASE(003,130)
REL_CPL(003,130)
REL_CPL(002,017)

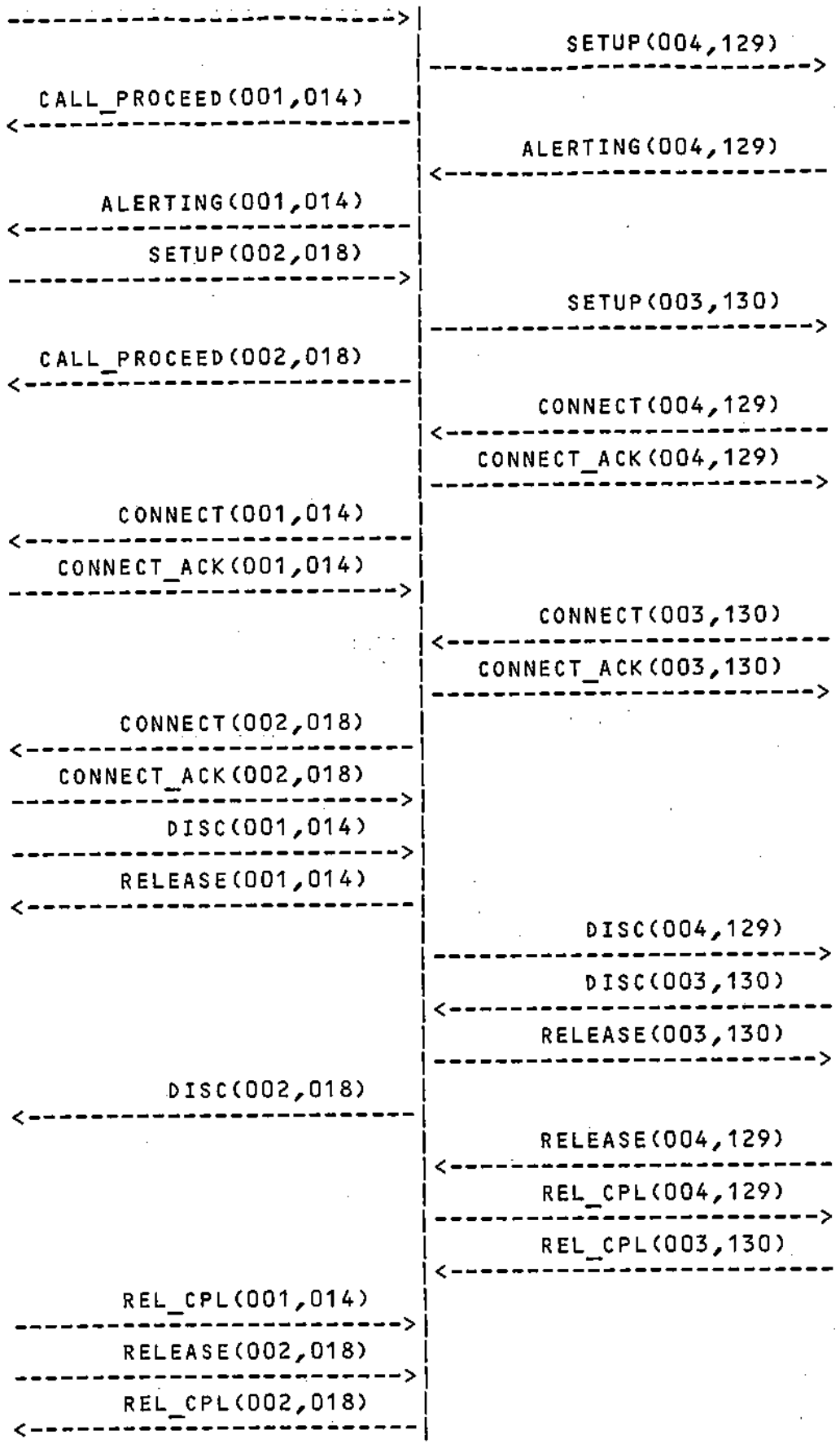
```
send 0 ALERTING 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2
env[0]:ALERTING 2 14 1 2

send 0 CONNECT  4 129 1 1
env[0]:CONNECT_ACK 4 129 0
env[0]:CONNECT 2 14 1 2

send 0 CONNECT_ACK 2 14 0

send 0 SETUP 1 17 0 12 1 104
CID for new call: 2
cc1 error: xnr 104 busy
env[0]:REL_CPL 1 17 1

send 0 SETUP 1 18 0 12 1 103
CID for new call: 2
env[0]:SETUP 3 130 0 2 1 103
env[0]:CALL_PROCEED 1 18 1 1

send 0 ALERTING 3 130 1 2
env[0]:SW_ON 1 0 3 2 1 4
env[0]:ALERTING 1 18 1 1

send 0 CONNECT  3 130 1 2
env[0]:CONNECT_ACK 3 130 0
env[0]:CONNECT 1 18 1 1

send 0 CONNECT_ACK 1 18 0

send 0 DISC 2 14 0
env[0]:RELEASE 2 14 1
env[0]:DISC 4 129 0

send 0 DISC 3 130 1
env[0]:RELEASE 3 130 0
env[0]:DISC 1 18 1

send 0 REL_CPL 2 14 0
env[0]:SW_OFF 2 0 1 1 1 2

send 0 RELEASE 1 18 0
env[0]:REL_CPL 1 18 1

send 0 RELEASE 4 129 1
env[0]:REL_CPL 4 129 0

send 0 REL_CPL 3 130 1
env[0]:SW_OFF 1 0 3 2 1 4

>>
```


SETUP(002,014)
SETUP(004,129)
CALL_PROCEED(002,014)
ALERTING(002,129)
ALERTING(004,129)
ALERTING(002,014)
CONNECT(004,129)
CONNECT_ACK(004,129)
CONNECT(002,014)
CONNECT_ACK(002,014)
SETUP(001,017)
REL_CPL(001,017)
SETUP(001,018)
SETUP(003,130)
CALL_PROCEED(001,018)
ALERTING(003,130)
ALERTING(001,018)
CONNECT(003,130)
CONNECT_ACK(003,130)
CONNECT(001,018)
CONNECT_ACK(001,018)
DISC(002,014)
RELEASE(002,014)
DISC(004,129)
DISC(003,130)
RELEASE(003,130)
DISC(001,018)
REL_CPL(002,014)
RELEASE(001,018)
REL_CPL(001,018)
RELEASE(004,129)
REL_CPL(004,129)
REL_CPL(003,130)

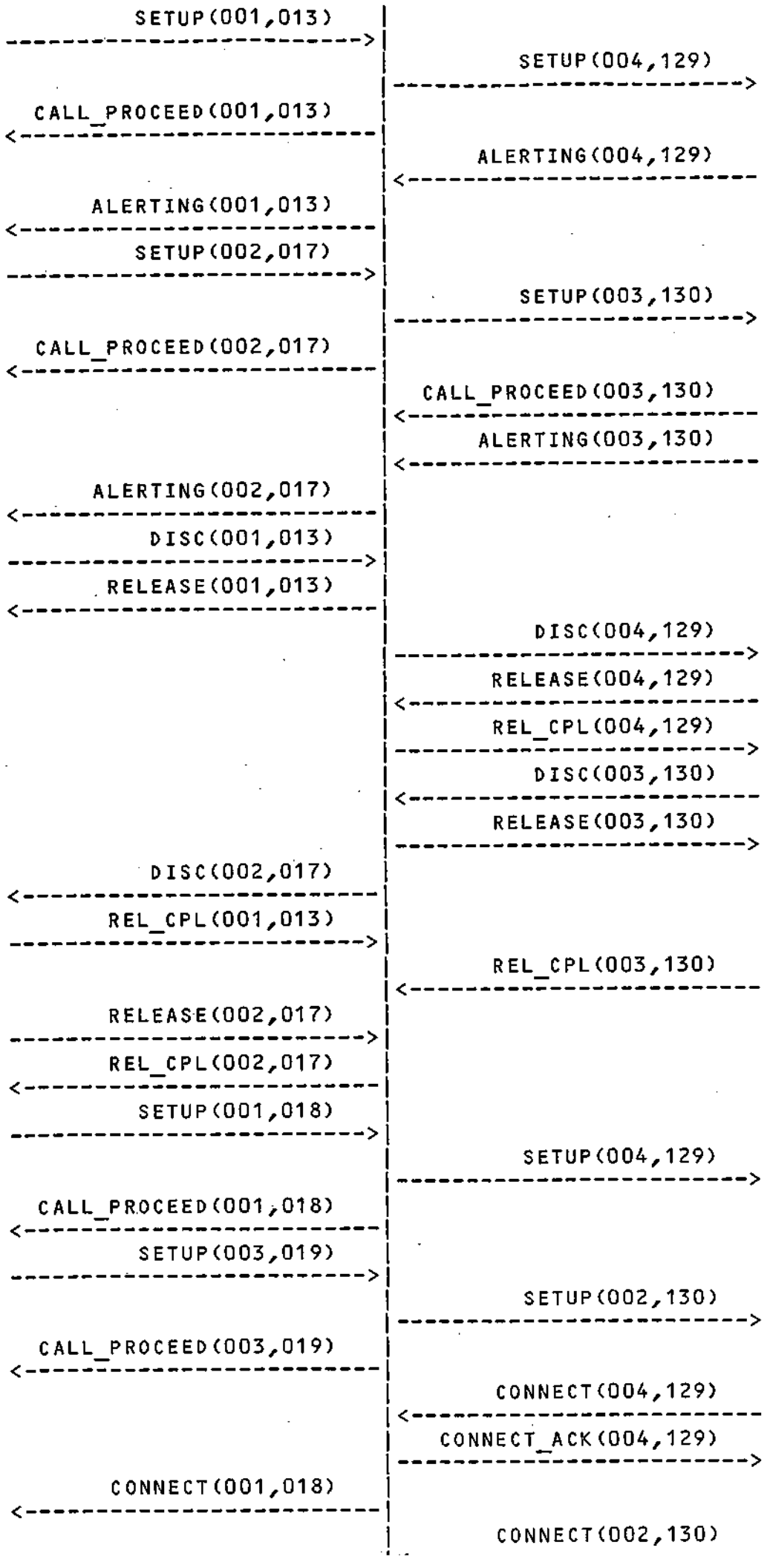
```
  testgen 3
send 0 SETUP 1 13 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 13 1 2

send 0 ALERTING 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2
env[0]:ALERTING 1 13 1 2

send 0 SETUP 2 17 0 12 1 103
CID for new call: 2
env[0]:SETUP 3 130 0 2 1 103
env[0]:CALL_PROCEED 2 17 1 1

send 0 CONNECT  4 129 1 1
env[0]:CONNECT_ACK 4 129 0
env[0]:CONNECT 1 13 1 2

send 0 CONNECT_ACK 1 13 0

send 0 CONNECT  3 130 1 2
env[0]:CONNECT_ACK 3 130 0
env[0]:SW_ON 1 0 3 2 1 4
env[0]:CONNECT 2 17 1 1

send 0 CONNECT_ACK 2 17 0

send 0 SETUP 1 15 0 0 1 103
CID for new call: 3
cc1 error: xnr 103 busy
env[0]:REL_CPL 1 15 1

send 0 DISC 1 13 0
env[0]:RELEASE 1 13 1
env[0]:DISC 4 129 0

send 0 REL_CPL 1 13 0
env[0]:SW_OFF 2 0 1 1 1 2

send 0 RELEASE 4 129 1
env[0]:REL_CPL 4 129 0

send 0 DISC 3 130 1
env[0]:RELEASE 3 130 0
env[0]:DISC 2 17 1

send 0 REL_CPL 3 130 1
env[0]:SW_OFF 1 0 3 2 1 4

send 0 RELEASE 2 17 0
env[0]:REL_CPL 2 17 1

send 0 SETUP 1 14 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 14 1 2

send 0 ALERTING 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2
env[0]:ALERTING 1 14 1 2

send 0 SETUP 2 18 0 12 1 103
CID for new call: 2
env[0]:SETUP 3 130 0 2 1 103
env[0]:CALL_PROCEED 2 18 1 1
```


SETUP(001,013)
SETUP(004,129)
CALL_PROCEED(001,013)
ALERTING(004,129)
ALERTING(001,013)
SETUP(002,017)
SETUP(003,130)
CALL_PROCEED(002,017)
CONNECT(004,129)
CONNECT_ACK(004,129)
CONNECT(001,013)
CONNECT_ACK(001,013)
CONNECT(003,130)
CONNECT_ACK(003,130)
CONNECT(002,017)
CONNECT_ACK(002,017)
SETUP(001,015)
REL_CPL(001,015)
DISC(001,013)
RELEASE(001,013)
DISC(004,129)
REL_CPL(001,013)
RELEASE(004,129)
REL_CPL(004,129)
DISC(003,130)
RELEASE(003,130)
DISC(002,017)
REL_CPL(003,130)
RELEASE(002,017)
REL_CPL(002,017)
SETUP(001,014)

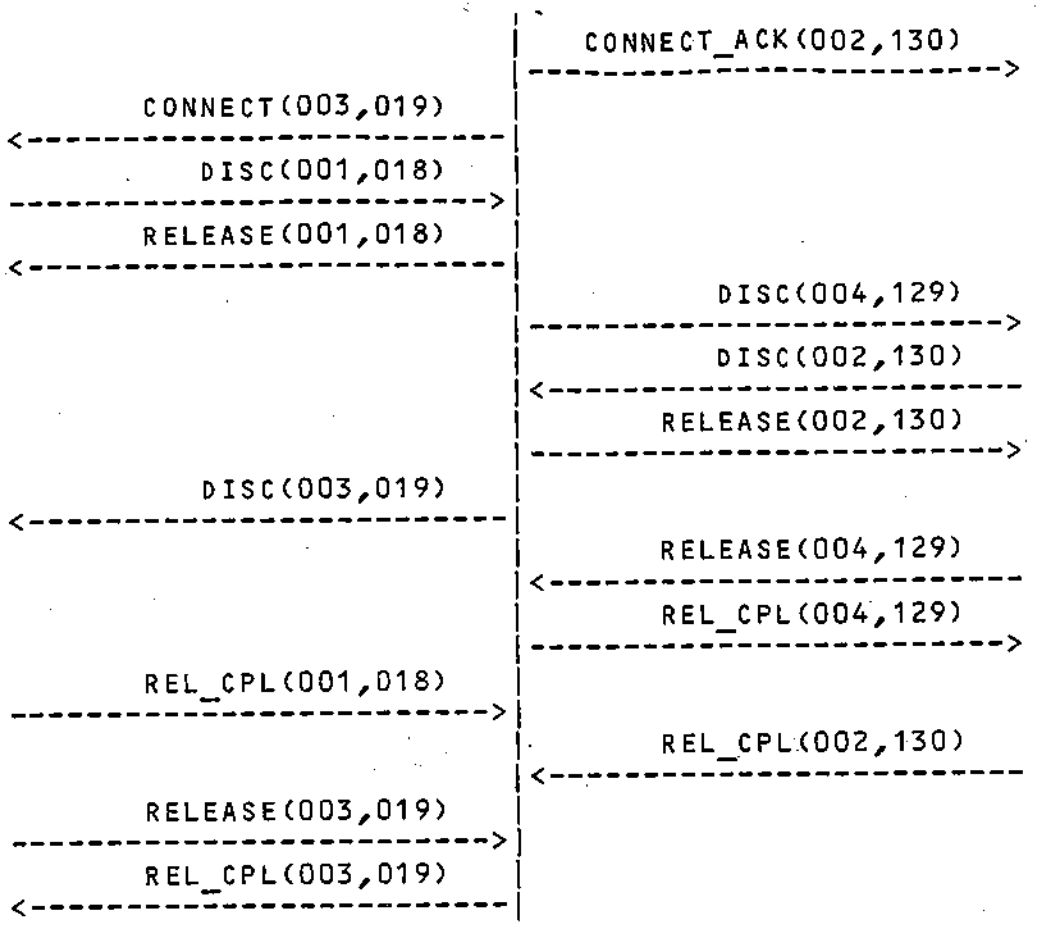
```
env[0]:CALL_PROCEED 2 18 1 1

send 0 CONNECT  4 129 1 1
env[0]:CONNECT_ACK 4 129 0
env[0]:CONNECT 1 14 1 2

send 0 CONNECT_ACK 1 14 0

send 0 CONNECT  3 130 1 2
env[0]:CONNECT_ACK 3 130 0
env[0]:SW_ON 1 0 3 2 1 4
env[0]:CONNECT 2 18 1 1

send 0 CONNECT_ACK 2 18 0

send 0 DISC 1 14 0
env[0]:RELEASE 1 14 1
env[0]:DISC 4 129 0

send 0 DISC 3 130 1
env[0]:RELEASE 3 130 0
env[0]:DISC 2 18 1

send 0 RELEASE 4 129 1
env[0]:REL_CPL 4 129 0

send 0 REL_CPL 3 130 1
env[0]:SW_OFF 1 0 3 2 1 4

send 0 REL_CPL 1 14 0
env[0]:SW_OFF 2 0 1 1 1 2

send 0 RELEASE 2 18 0
env[0]:REL_CPL 2 18 1

>>
```


SETUP(004,129)
CALL_PROCEED(001,014)
ALERTING(004,129)
ALERTING(001,014)
SETUP(002,018)
SETUP(003,130)
CALL_PROCEED(002,018)
CONNECT(004,129)
CONNECT_ACK(004,129)
CONNECT(001,014)
CONNECT_ACK(001,014)
CONNECT(003,130)
CONNECT_ACK(003,130)
CONNECT(002,018)
CONNECT_ACK(002,018)
DISC(001,014)
RELEASE(001,014)
DISC(004,129)
DISC(003,130)
RELEASE(003,130)
DISC(002,018)
RELEASE(004,129)
REL_CPL(004,129)
REL_CPL(003,130)
REL_CPL(001,014)
RELEASE(002,018)
REL_CPL(002,018)

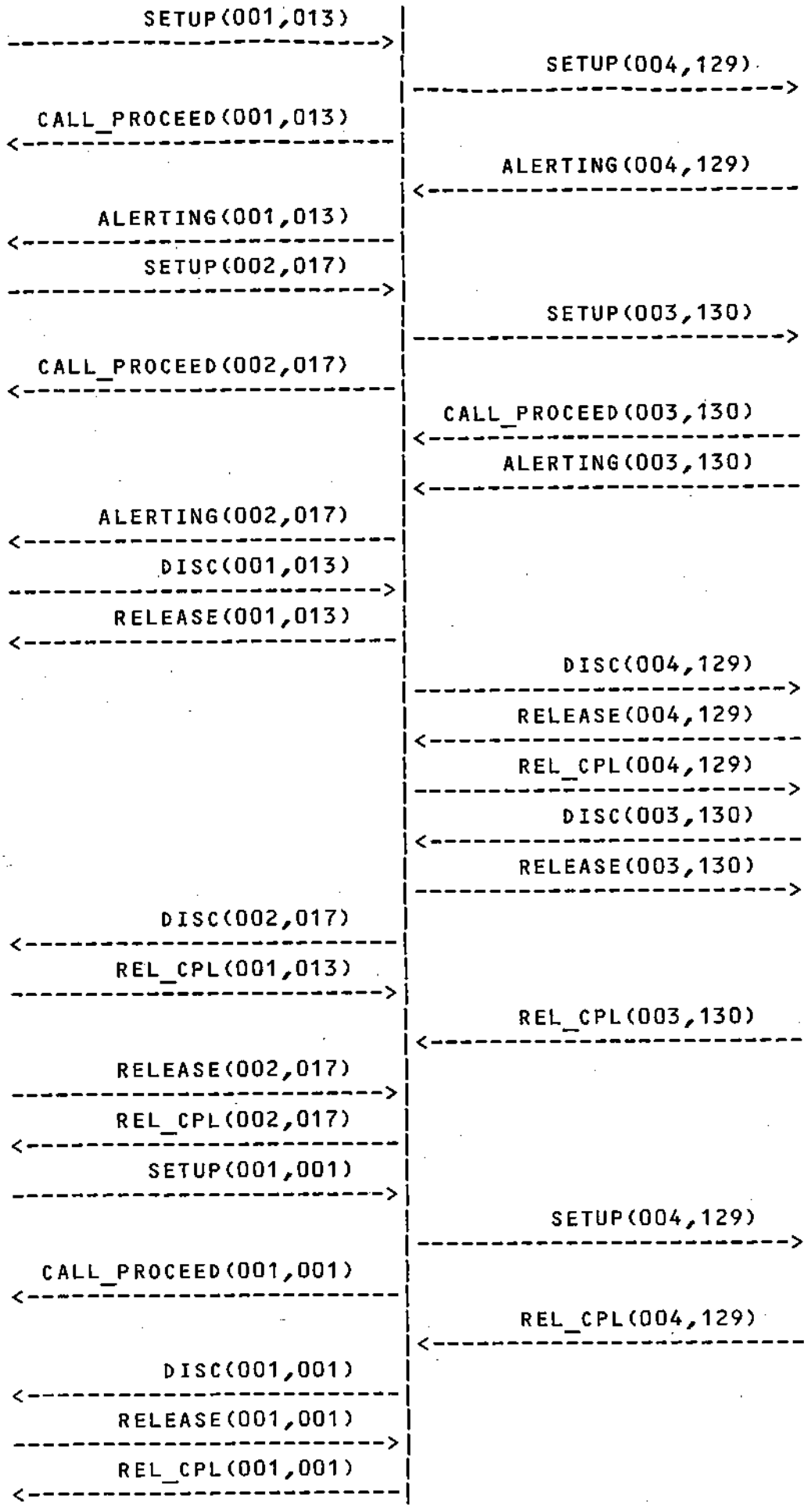
```
send 0 RELEASE 3 19 0

>>


  testgen 4
send 0 SETUP 1 13 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 13 1 2

send 0 ALERTING 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2
env[0]:ALERTING 1 13 1 2

send 0 SETUP 2 17 0 12 1 103
CID for new call: 2
env[0]:SETUP 3 130 0 2 1 103
env[0]:CALL_PROCEED 2 17 1 1

send 0 CALL_PROCEED 3 130 1 2
env[0]:SW_ON 1 0 3 2 1 4

send 0 ALERTING 3 130 1 2
env[0]:ALERTING 2 17 1 1

send 0 DISC 1 13 0
env[0]:RELEASE 1 13 1
env[0]:SW_OFF 2 0 1 1 1 2
env[0]:DISC 4 129 0

send 0 RELEASE 4 129 1
env[0]:REL_CPL 4 129 0

send 0 DISC 3 130 1
env[0]:RELEASE 3 130 0
env[0]:SW_OFF 1 0 3 2 1 4
env[0]:DISC 2 17 1

send 0 REL_CPL 1 13 0

send 0 REL_CPL 3 130 1

send 0 RELEASE 2 17 0
env[0]:REL_CPL 2 17 1

send 0 SETUP 1 18 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 18 1 2

send 0 SETUP 3 19 0 12 1 102
CID for new call: 2
env[0]:SETUP 2 130 0 1 1 102
env[0]:CALL_PROCEED 3 19 1 2

send 0 CONNECT 4 129 1 1
env[0]:CONNECT_ACK 4 129 0
env[0]:SW_ON 2 0 1 1 1 2
env[0]:CONNECT 1 18 1 2

send 0 CONNECT 2 130 1 1
env[0]:CONNECT_ACK 2 130 0
env[0]:SW_ON 2 1 3 1 0 4
env[0]:CONNECT 3 19 1 2
```


SETUP(001,013)
SETUP(004,129)
CALL_PROCEED(001,013)
ALERTING(004,129)
ALERTING(001,013)
SETUP(002,017)
SETUP(003,130)
CALL_PROCEED(002,017)
CALL_PROCEED(003,130)
ALERTING(003,130)
ALERTING(002,017)
DISC(001,013)
RELEASE(001,013)
DISC(004,129)
RELEASE(004,129)
REL_CPL(004,129)
DISC(003,130)
RELEASE(003,130)
DISC(002,017)
REL_CPL(001,013)
REL_CPL(003,130)
RELEASE(002,017)
REL_CPL(002,017)
SETUP(001,018)
SETUP(004,129)
CALL_PROCEED(001,018)
SETUP(003,019)
SETUP(002,130)
CALL_PROCEED(003,019)
CONNECT(004,129)
CONNECT_ACK(004,129)
CONNECT(001,018)
CONNECT(002,130)

```
send 0 DISC 1 18 0
env[0]:RELEASE 1 18 1
env[0]:DISC 4 129 0

send 0 DISC 2 130 1
env[0]:RELEASE 2 130 0
env[0]:DISC 3 19 1

send 0 RELEASE 4 129 1
env[0]:REL_CPL 4 129 0

send 0 REL_CPL 1 18 0
env[0]:SW_OFF 2 0 1 1 1 2

send 0 REL_CPL 2 130 1
env[0]:SW_OFF 2 1 3 1 0 4

send 0 RELEASE 3 19 0
env[0]:REL_CPL 3 19 1

>>
```


CONNECT_ACK(002,130)
CONNECT(003,019)
DISC(001,018)
RELEASE(001,018)
DISC(004,129)
DISC(002,130)
RELEASE(002,130)
DISC(003,019)
RELEASE(004,129)
REL_CPL(004,129)
REL_CPL(001,018)
REL_CPL(002,130)
RELEASE(003,019)
REL_CPL(003,019)

```
  testgen 5
send 0 SETUP 1 13 0 22 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 13 1 2

send 0 ALERTING 4 129 1 1
env[0]:SW_ON 2 0 1 1 1 2
env[0]:ALERTING 1 13 1 2

send 0 SETUP 2 17 0 12 1 103
CID for new call: 2
env[0]:SETUP 3 130 0 2 1 103
env[0]:CALL_PROCEED 2 17 1 1

send 0 CALL_PROCEED 3 130 1 2
env[0]:SW_ON 1 0 3 2 1 4

send 0 ALERTING 3 130 1 2
env[0]:ALERTING 2 17 1 1

send 0 DISC 1 13 0
env[0]:RELEASE 1 13 1
env[0]:SW_OFF 2 0 1 1 1 2
env[0]:DISC 4 129 0

send 0 RELEASE 4 129 1
env[0]:REL_CPL 4 129 0

send 0 DISC 3 130 1
env[0]:RELEASE 3 130 0
env[0]:SW_OFF 1 0 3 2 1 4
env[0]:DISC 2 17 1

send 0 REL_CPL 1 13 0

send 0 REL_CPL 3 130 1

send 0 RELEASE 2 17 0
env[0]:REL_CPL 2 17 1

send 0 SETUP 1 1 0 0 1 104
CID for new call: 1
env[0]:SETUP 4 129 0 1 1 104
env[0]:CALL_PROCEED 1 1 1 1

send 0 REL_CPL 4 129 1
env[0]:DISC 1 1 1

send 0 RELEASE 1 1 0
env[0]:REL_CPL 1 1 1

>>
```


SETUP(001,013)
SETUP(004,129)
CALL_PROCEED(001,013)
ALERTING(004,129)
ALERTING(001,013)
SETUP(002,017)
SETUP(003,130)
CALL_PROCEED(002,017)
CALL_PROCEED(003,130)
ALERTING(003,130)
ALERTING(002,017)
DISC(001,013)
RELEASE(001,013)
DISC(004,129)
RELEASE(004,129)
REL_CPL(004,129)
DISC(003,130)
RELEASE(003,130)
DISC(002,017)
REL_CPL(001,013)
REL_CPL(003,130)
RELEASE(002,017)
REL_CPL(002,017)
SETUP(001,001)
SETUP(004,129)
CALL_PROCEED(001,001)
REL_CPL(004,129)
DISC(001,001)
RELEASE(001,001)
REL_CPL(001,001)