เอกสารอ้างอิง

1. J. Bellamy, Digital Telephony , John Wiley & Sons INC., 1982.
2. S. Takamura, H. Kawashima, H. Nakajima, Software Designed for Electronic Switching Systems , M.T. Hills Editor, Peter Peragrinus LTD., 1979
3. MT. Hills, S. KANO ,Programing electronic Switching System , real-time aspects and their language implications. Peter Peregrinus LTD.
4. เจริญชัย เจริญทั้งเมือง, กฤษดา วิศวธีรานนท์,"การพัฒนาโปรแกรมควบคุมสำหรับตู้ชุมสาขาโทรศัพท์อัตโนมัติโดยใช้ภาษา STL",การประชุมทางวิชาการวิศวกรรมไฟฟ้าครั้งที่ 10 จุฬาลงกรณ์มหาวิทยาลัย 2530
5. ธเนส โชติรัตนพิทักษ์ สุรศักดิ์ อุทโยภาศ บัณฑิต โรจน์อารยานนท์, "การออกแบบและสร้างตู้สาขาโทรศัพท์อัตโนมัติระบบดิจิตอลขนาด 256 พอร์ต", การประชุมทางวิชาการวิศวกรรมไฟฟ้าครั้งที่ 12, มหาวิทยาลัยเกษตรศาสตร์ 2532
6. Brain W.Kernighan,Dennis M. Ritchie ,THE C PROGRAMMING LANGUAGE 2nd Editon ,Prentice-Hall of India Private Limited.,1990
7. เจริญชัย เจริญทั้งเมือง ,การพัฒนาโปรแกรมควบคุมสำหรับตู้ชุมสายโทรศัพท์อัตโนมัติขนาดเล็กที่ใช้ไมโครโปรเซสเซอร์ Z80, วิทยานิพนธ์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย 2531
8. อนุรักษ์ เถื่อนศิริ ,การสร้างชุมสายโทรศัพท์สาขาขนาดเล็กระบบอิเล็กโทรนิกส์ วิทยานิพนธ์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย 2528
9. T.J. Cieslak, L.M. Crorall, J.B. Roberts, M.W. Saad, J.M. Scanlon "Software Organization and Basic Call Handling", The Bell System Technical Journal VOL.56 NO.7 September 1977

10. M.N. Meyers, W.A. Routt, K.W. Yoder, "Maintenance Software" The Bell System Technical Journal VOL.56 NO.7 September 1977

11. J.P. Delatore, R.J. Frank, H. Oehring, L.C. Stecher, "Operational Software" , AT&T TECHNICAL JOURNAL, VOL.64 NO.6,Part 2 July-August 1985

12. ดวงแก้ว สวามิภักดิ์, การโปรแกรมภาษาC ,บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2531

13. ยืน ภู่วรวรรณ , วัฒนา เชียงกูล, ไมโครคอมพิวเตอร์ ไมโครคอมพิวเตอร์ [Z-80 MICROPROCESSOR]", บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2524

14. Borland , TURBO C Reference Guide Version 2.0 Borland International INC. ,USA 1989

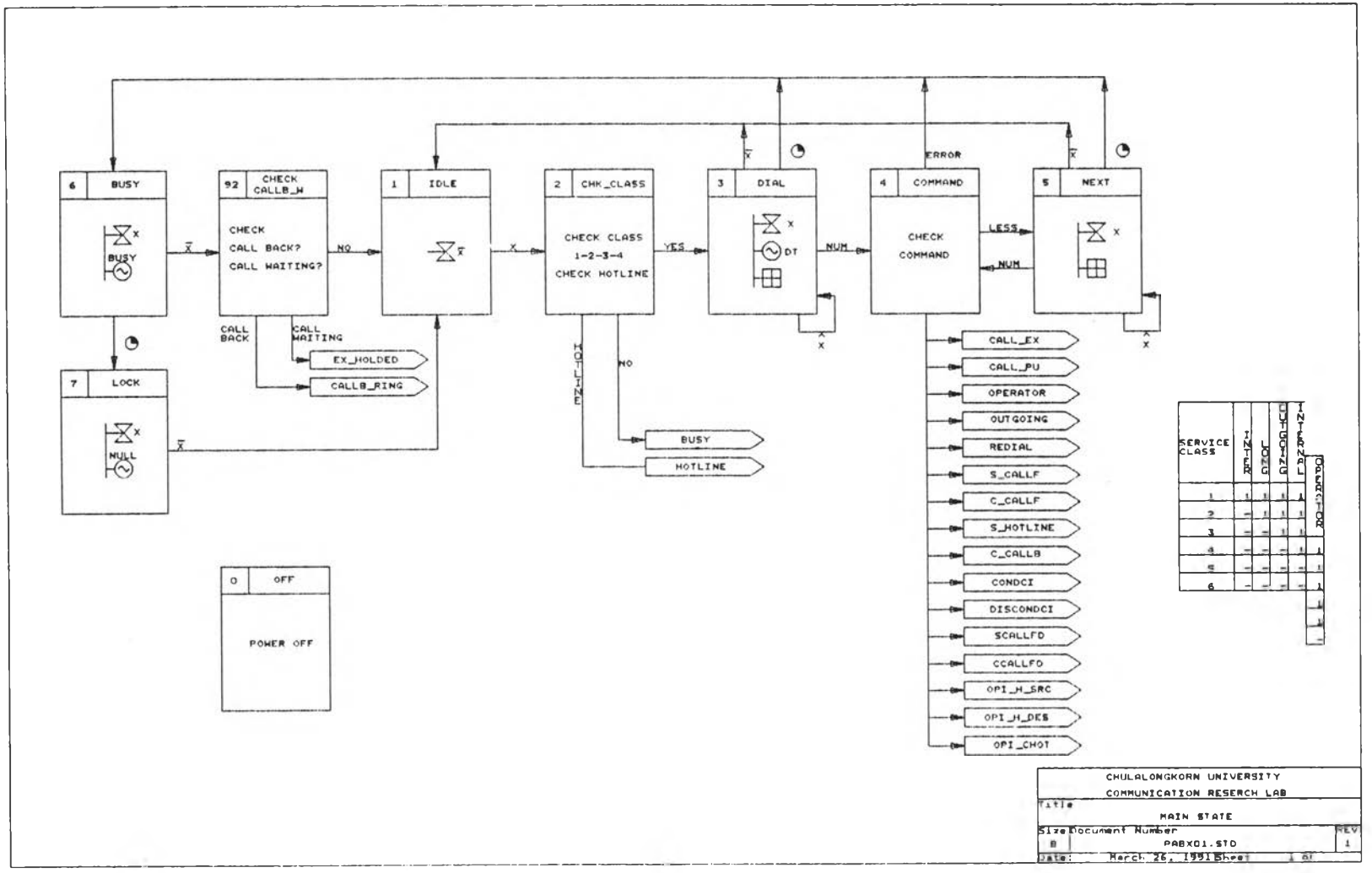
15. NEC, NEAX2400 IMS Features & Specifications (Voice service) NEC Corporation., January 1987.

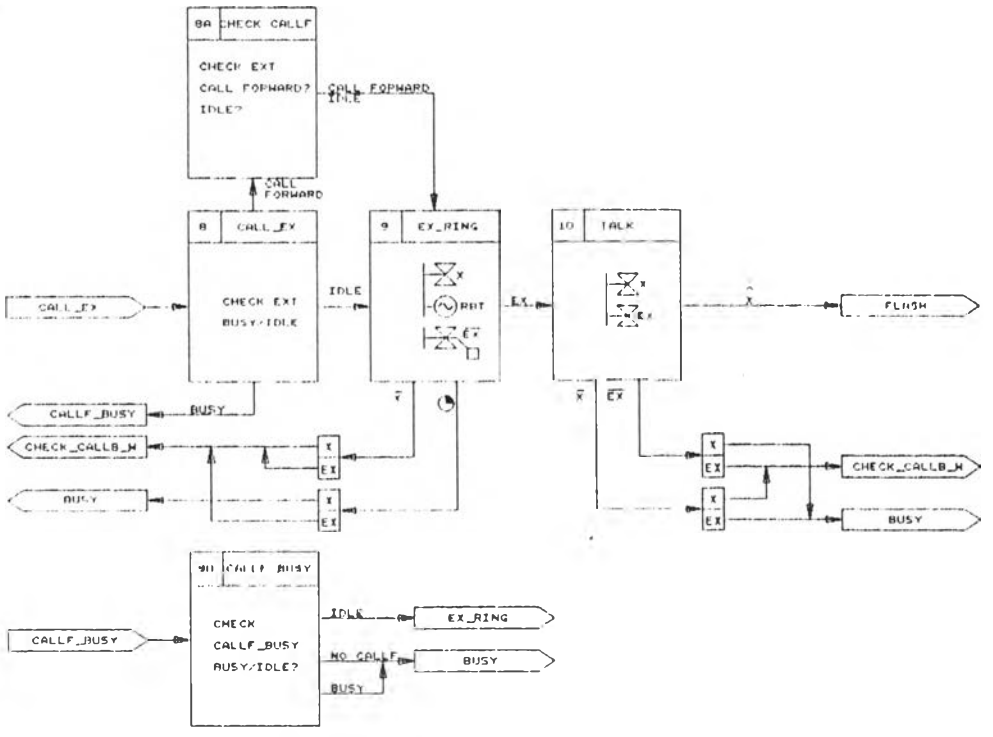
16. NEC, NEAX2400 IMS General Descrition, NEC Corporation. December 1986

ภาคผนวก

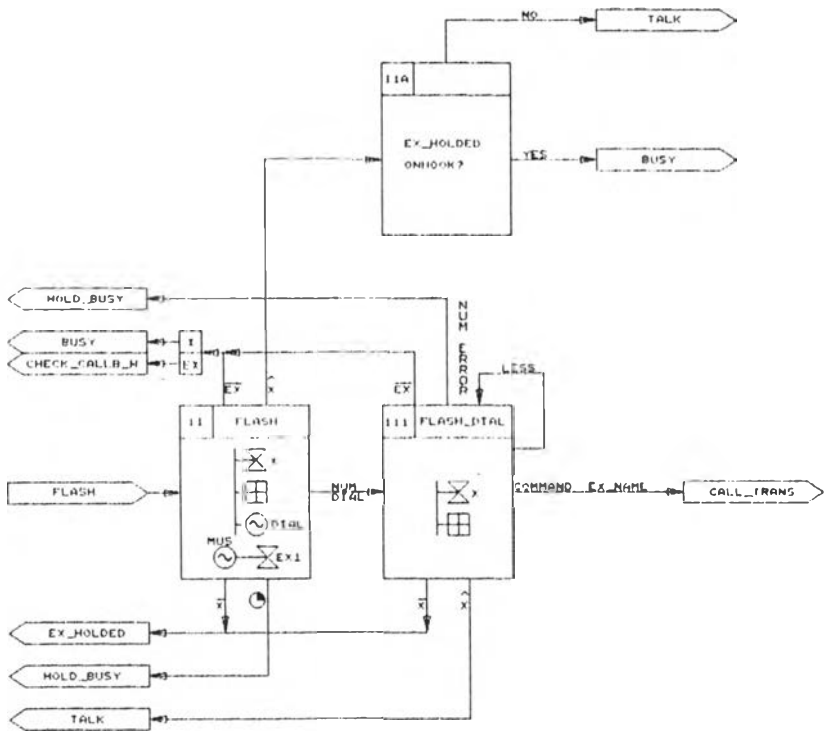
ภาคผนวก ก

รายละเอียดไดอะแกรมแสดงการเปลี่ยนสถานะ
ตู้ชุมสายโทรศัพท์อัตโนมัติระบบดิจิตอล 256 พอร์ต

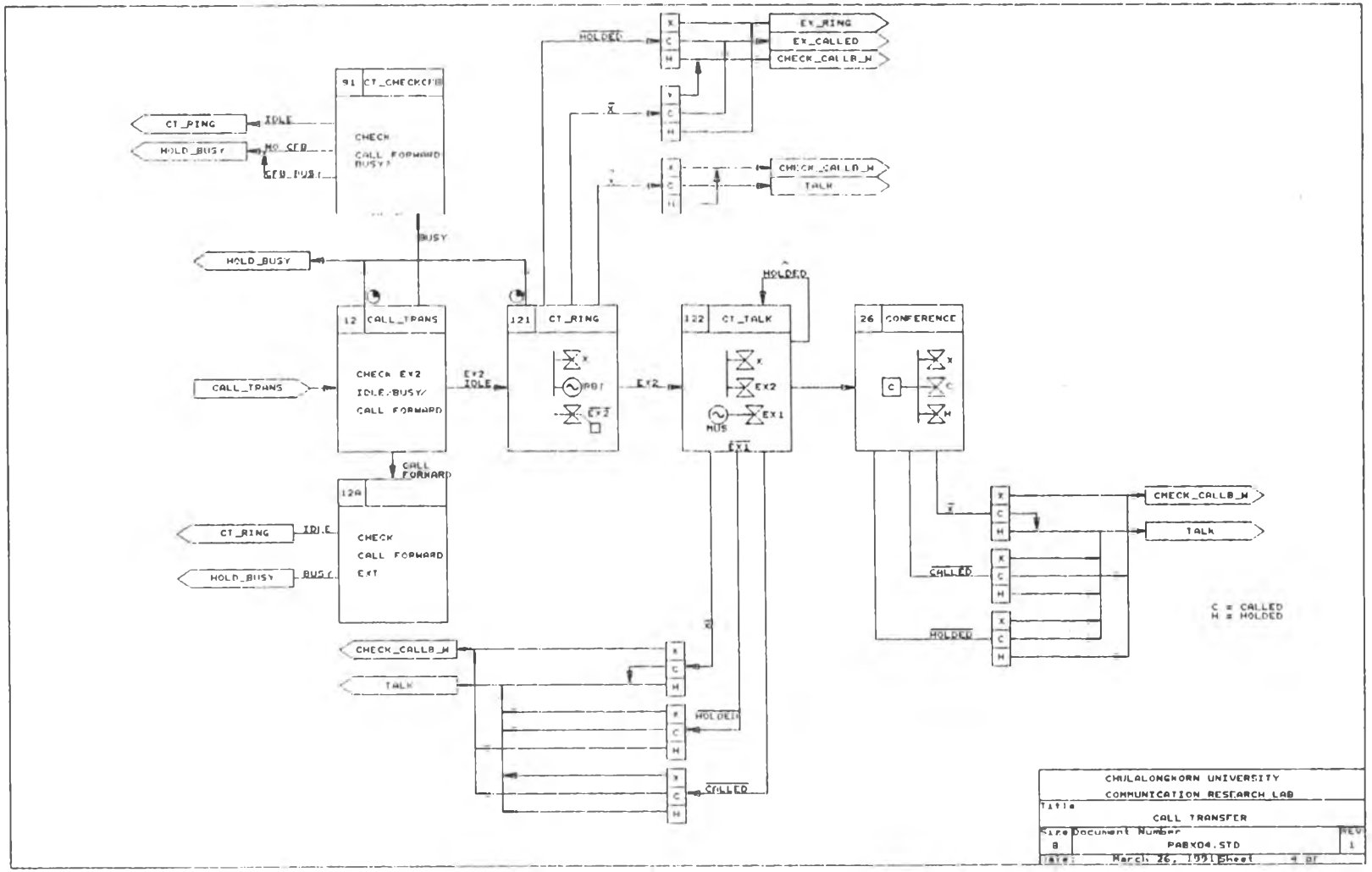

6 BUSY
BUSY
92 CHECK CALLB_W
CHECK
CALL BACK?
CALL WAITING?
1 IDLE
2 CHK_CLASS
CHECK CLASS
1-2-3-4
CHECK HOTLINE
3 DIAL
DT
4 COMMAND
CHECK
COMMAND
5 NEXT
7 LOCK
NULL
0 OFF
POWER OFF
ERROR
LESS
NUM
YES
NO
CALL BACK
CALL WAITING
EX_HOLDED
CALLB_RING
BUSY
HOTLINE
CALL_EX
CALL_PU
OPERATOR
OUTGOING
REDIAL
S_CALLF
C_CALLF
S_HOTLINE
C_CALLB
CONDCI
DISCONDCI
SCALLFD
CCALLFD
OPI_H_SRC
OPI_H_DES
OPI_CHOT
SERVICE CLASS
INTER
LOCAL
OUTGOING
INTERNAL
OPERATOR
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
MAIN STATE
Size
B
Document Number
PABX01.STD
REV
1
Date:
March 26, 1991
Sheet

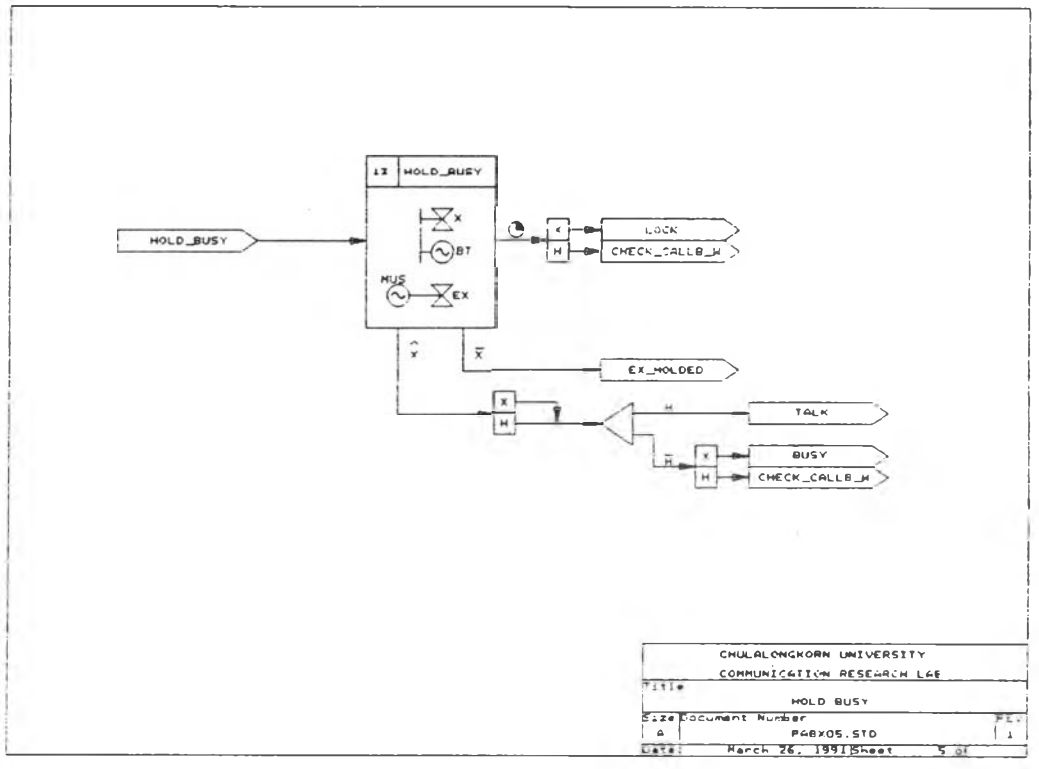


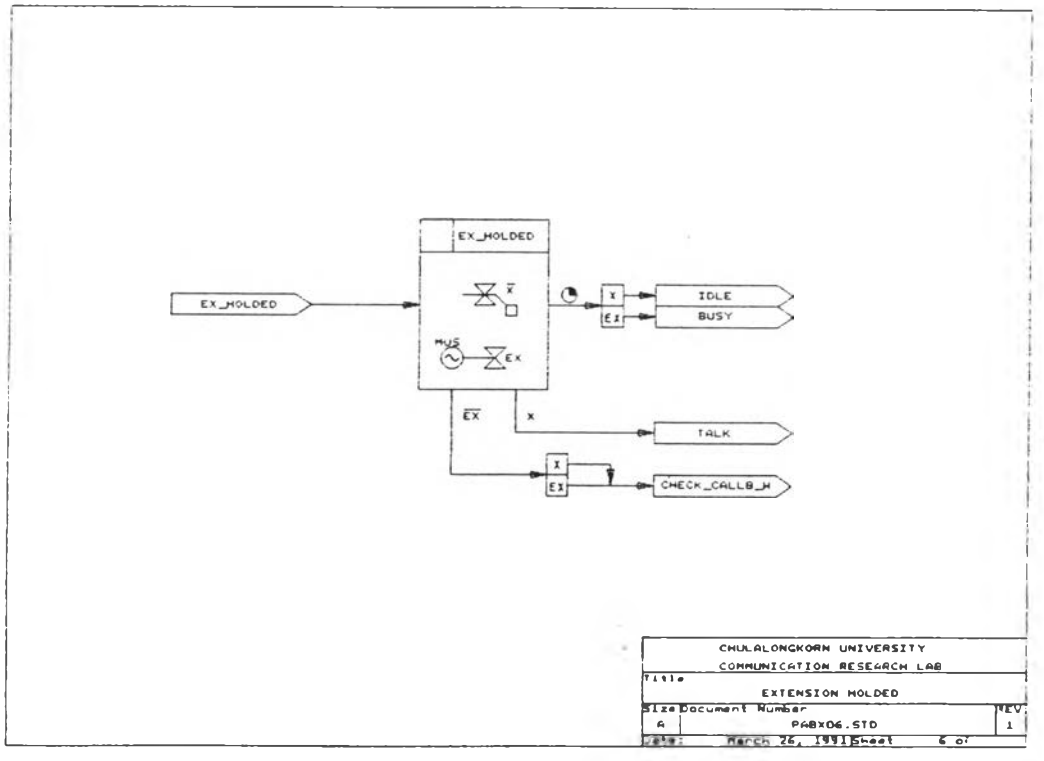
CHULALONGKORN UNIVERSITY		
COMMUNICATION RESEARCH LAB		
Title: CALL EXTENSION		
Size: B	Document Number: PABX02.STD	REV: 1
Date: March 26, 1991	Sheet of	

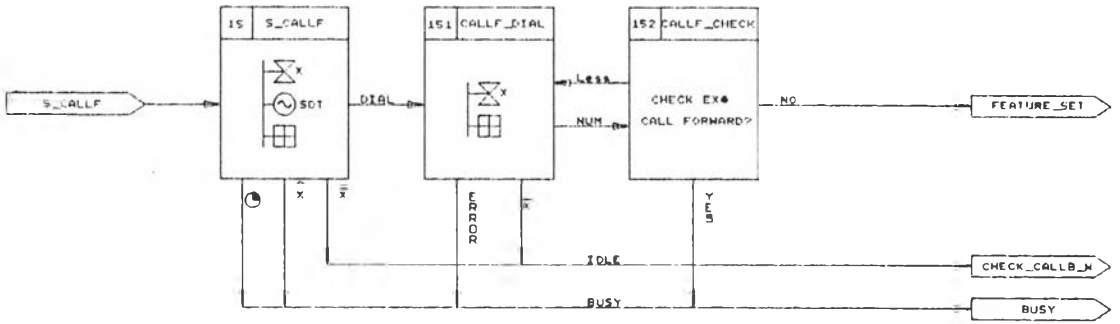


CHULALONGKORN UNIVERSITY COMMUNICATION RESEARCH LAB		
Title FLASH		
Size Document Number B	PABX03.STD	REV 1
Date: March 26, 1991 Sheet 3 of		

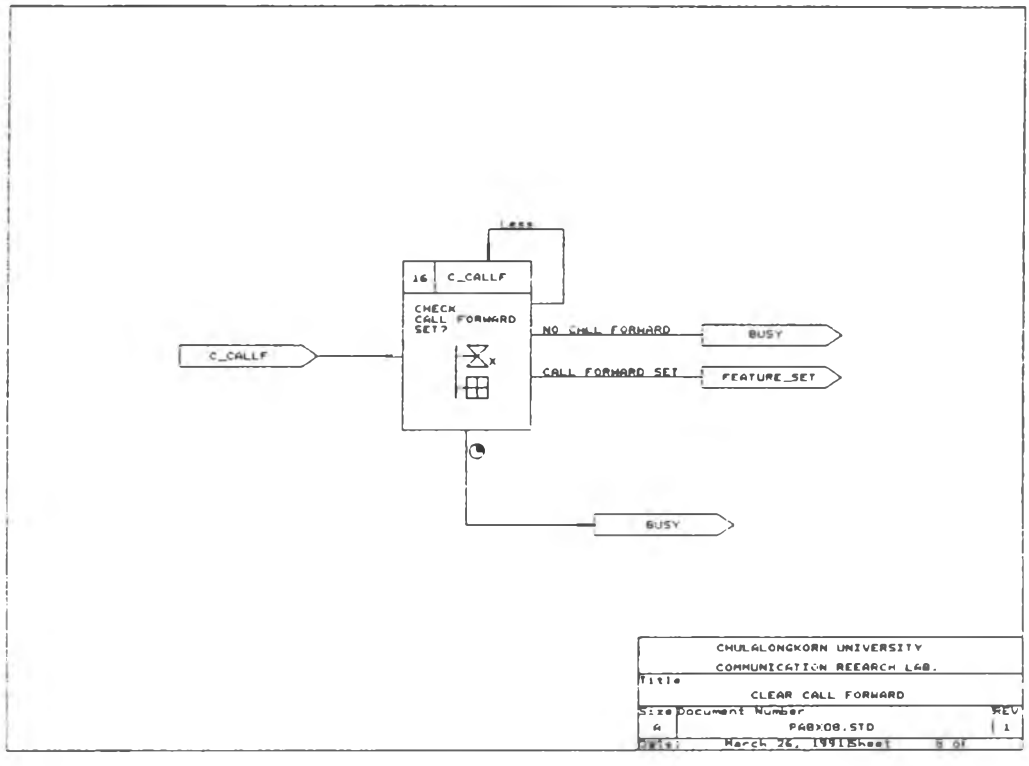

91 CT_CHECKCFB
CHECK
CALL FORWARD
BUSY?
CT_RING
HOLD_BUSY
IDLE
NO CFB
CFB BUSY
BUSY
HOLD_BUSY
12 CALL_TRANS
CHECK EXT
IDLE/BUSY/
CALL FORWARD
CALL_TRANS
EXT
IDLE
CALL FORWARD
12A
CHECK
CALL FORWARD
EXT
CT_RING
IDLE
HOLD_BUSY
BUSY
121 CT_RING
RBT
EXT
122 CT_TALK
EXT
EX1
MUS
26 CONFERENCE
HOLDED
EX_RING
EX_CALLED
CHECK_CALLB_W
TALK
CALLED
C = CALLED
H = HOLDED
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
CALL TRANSFER
Size
B
Document Number
PABX04.STD
REV
1
Date: March 26, 1991 Sheet 4 of

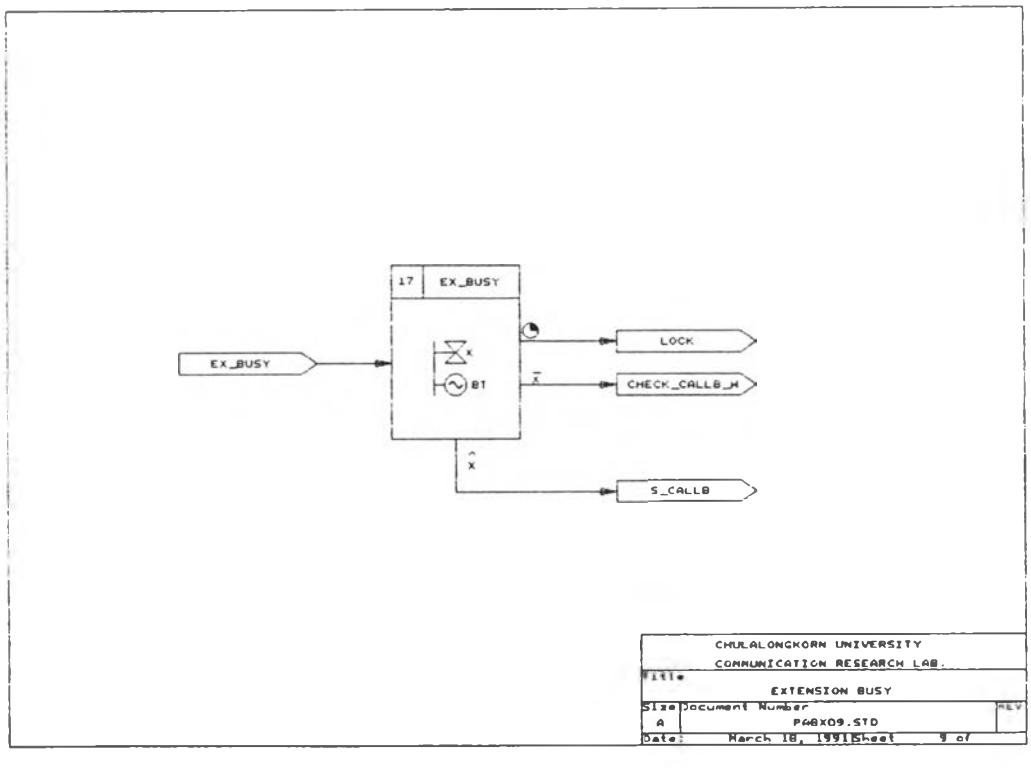

HOLD_BUSY
HOLD_BUSY
X
BT
MUS
EX
LOCK
CHECK_CALLB_H
EX_HOLDED
TALK
BUSY
CHECK_CALLB_H
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
HOLD BUSY
Size
A
Document Number
P48X05.STD
REV
1
Date: March 26, 1991 Sheet 5 of

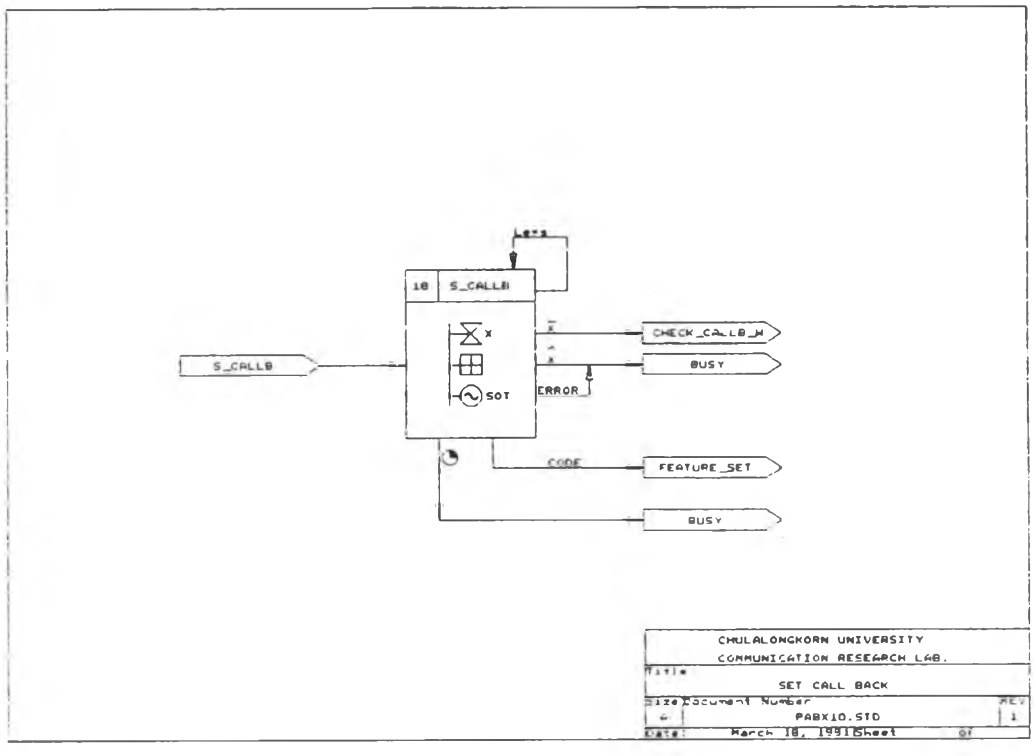

EX_HOLDED
EX_HOLDED
MUS
EX
IDLE
BUSY
TALK
CHECK_CALLB_H
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
EXTENSION HOLDED
Size
A
Document Number
P48X06.STD
REV
1
Date: March 26, 1991 Sheet 6 of

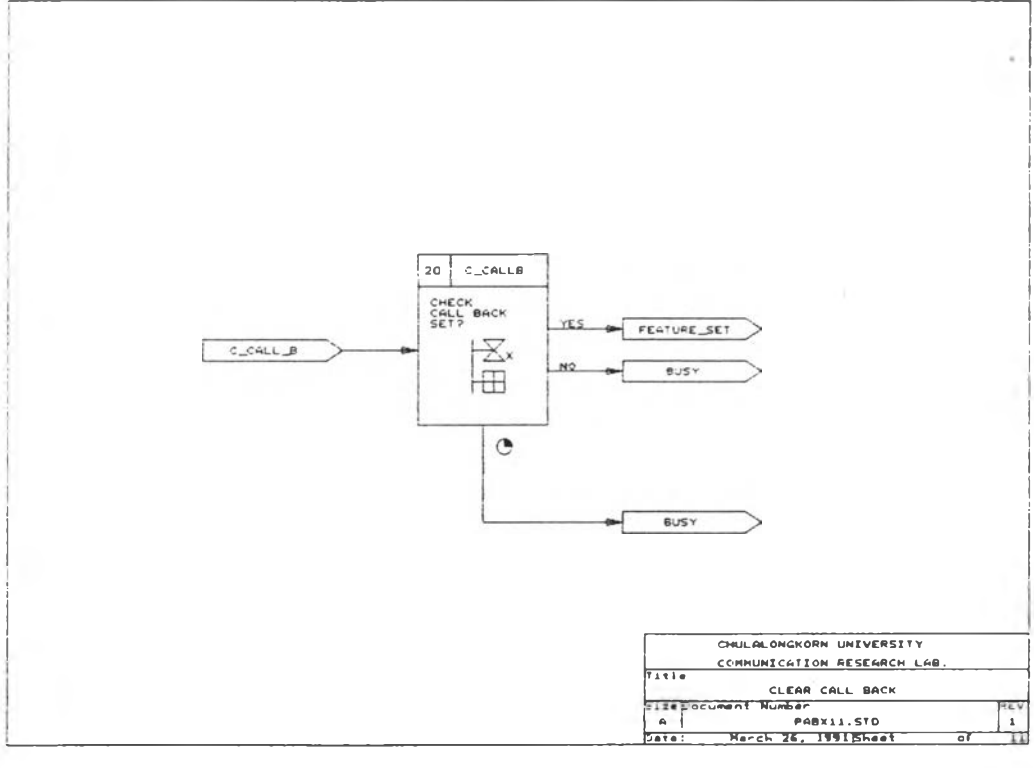


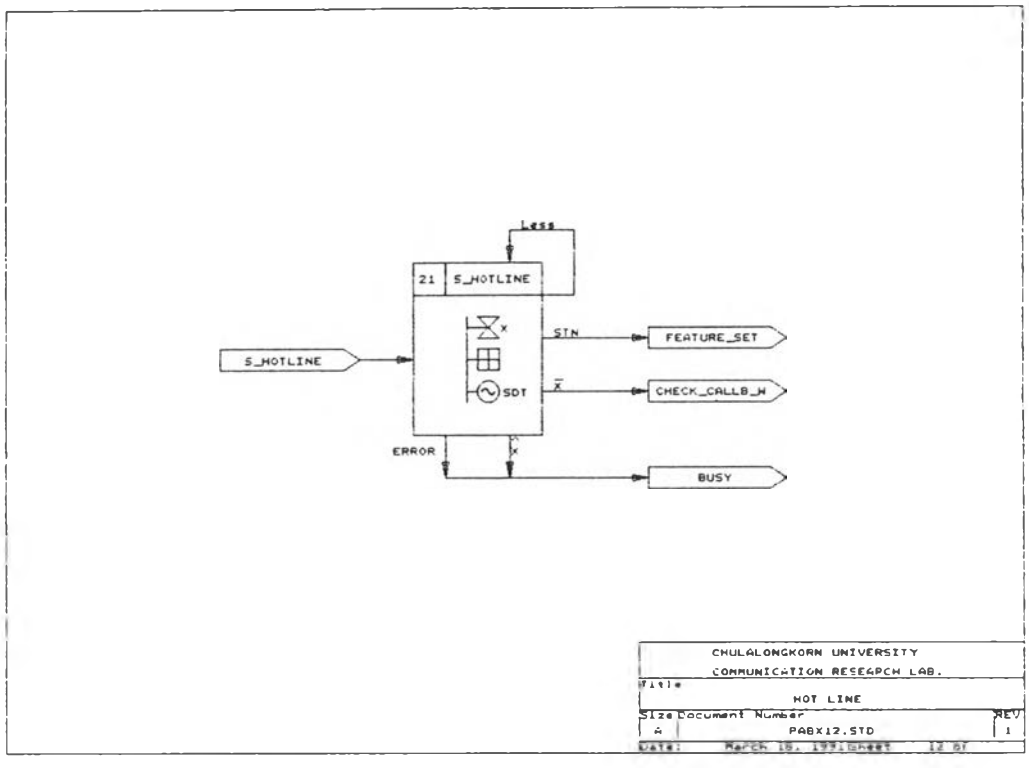
CHULALONGKORN UNIVERSITY		
COMMUNICATION RESEARCH LAB		
Title: SET CALL FORWARD		
Size: B	Document Number: PABX07.STD	REV: 1
Date: March 26, 1991	Sheet 7 of	

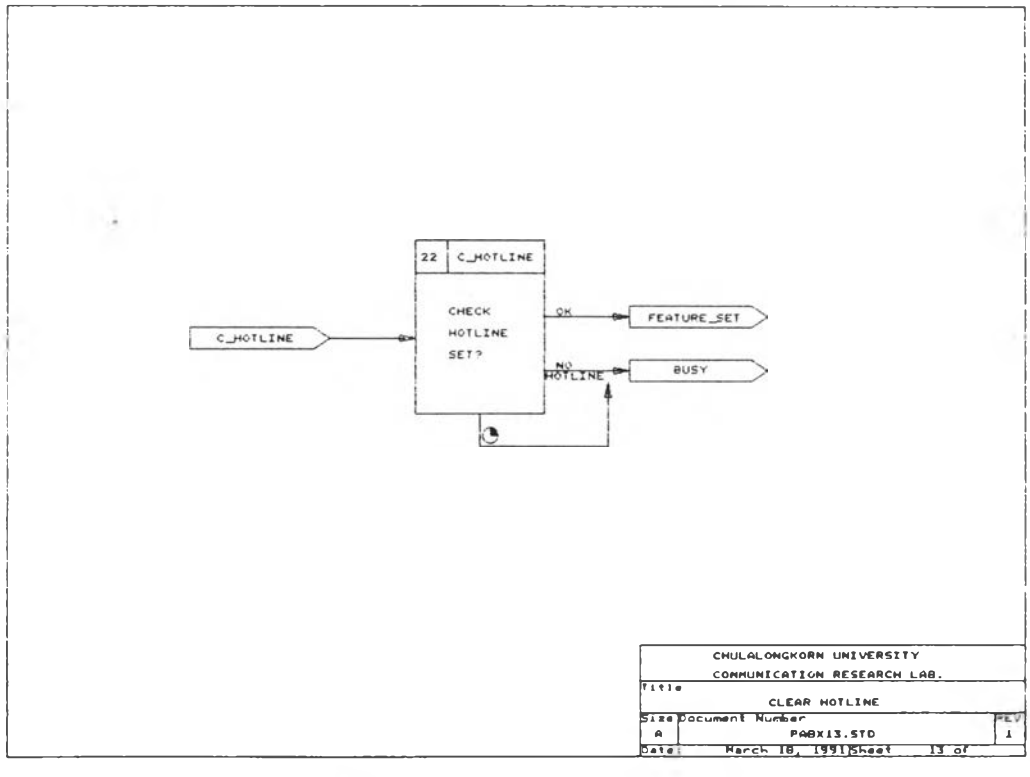

16
C_CALLF
CHECK
CALL FORWARD
SET?
C_CALLF
NO CALL FORWARD
BUSY
CALL FORWARD SET
FEATURE_SET
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
CLEAR CALL FORWARD
Size
A
Document Number
P48X08.STD
REV
1
Date:
March 26, 1991
Sheet
5 of

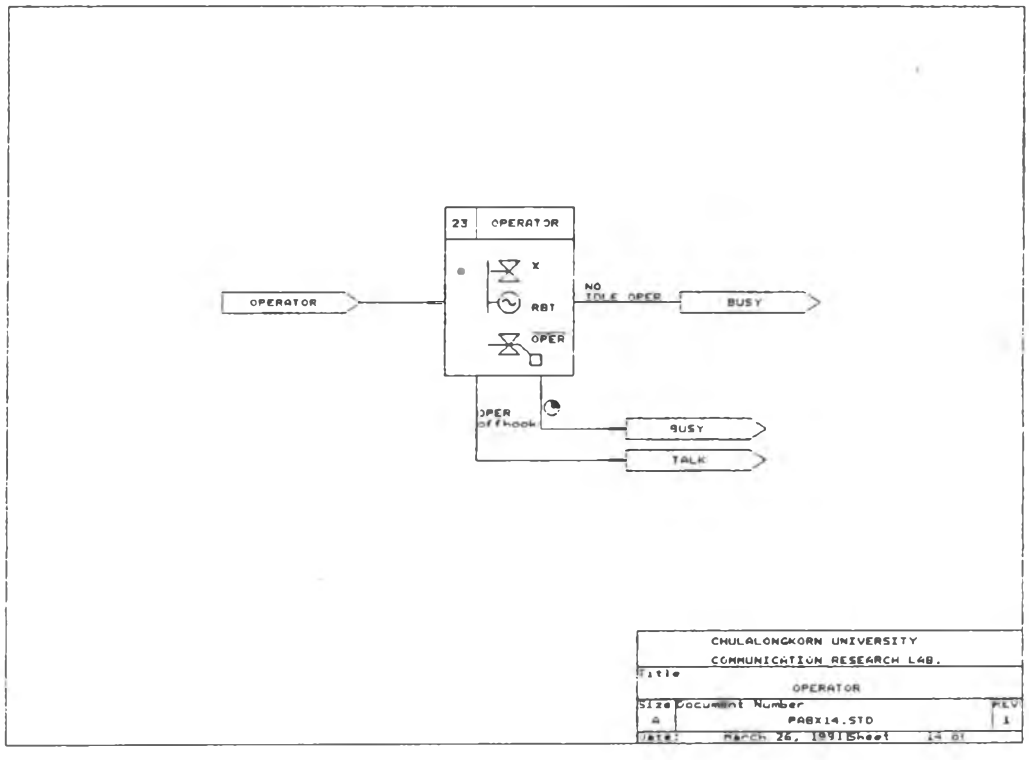

17
EX_BUSY
EX_BUSY
BT
LOCK
CHECK_CALLB_H
S_CALLB
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
EXTENSION BUSY
Size
A
Document Number
P48X09.STD
REV
Date:
March 18, 1991
Sheet
9 of

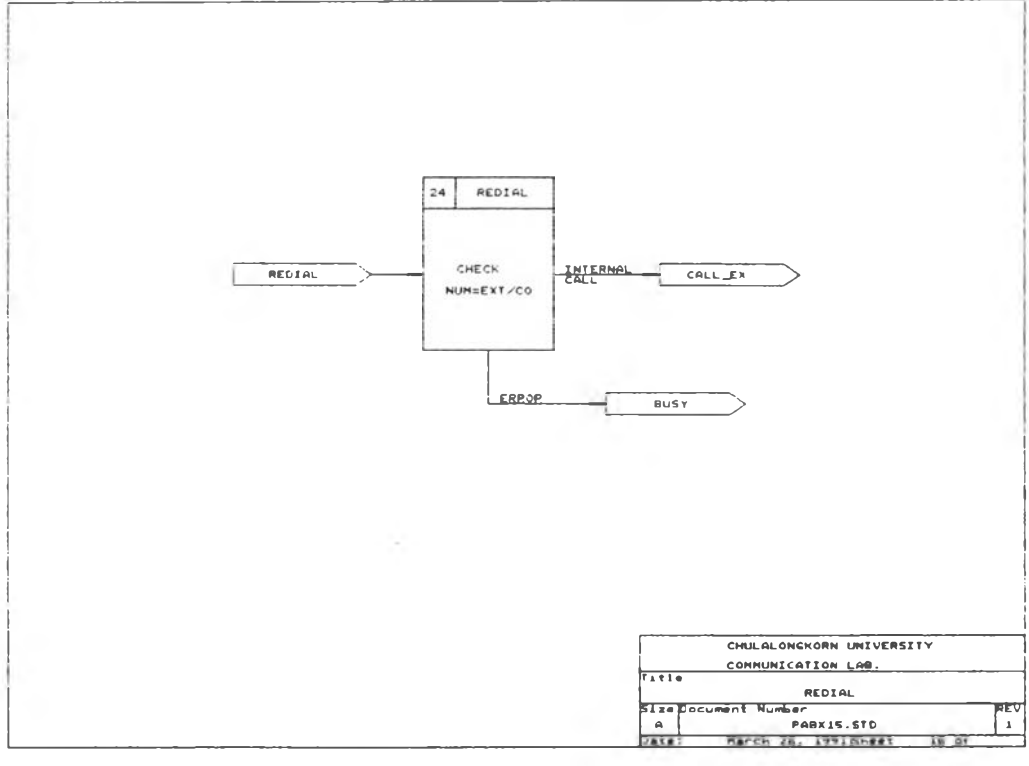

10
S_CALL8
S_CALL8
X
SOT
CHECK_CALL8_H
BUSY
ERROR
CODE
FEATURE_SET
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
SET CALL BACK
Size
A
Document Number
PABX10.STD
REV
1
Date: March 18, 1991
Sheet
of

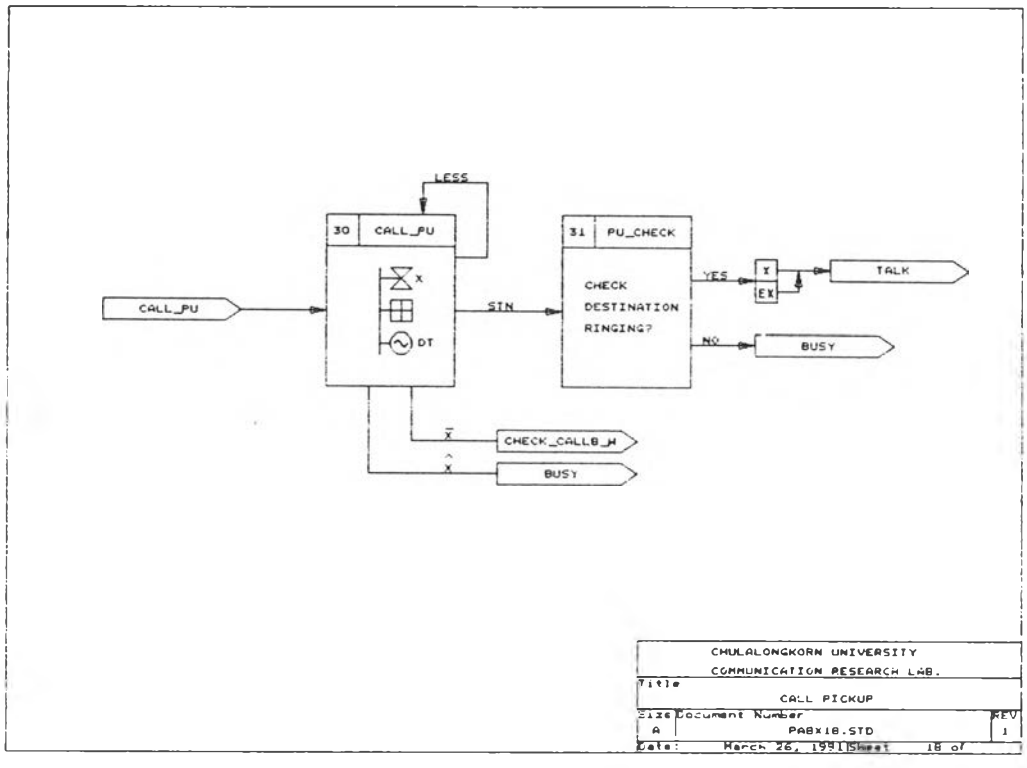

20
C_CALL8
CHECK
CALL BACK
SET?
C_CALL_B
X
YES
NO
FEATURE_SET
BUSY
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
CLEAR CALL BACK
Size
A
Document Number
PABX11.STD
REV
1
Date: March 26, 1991
Sheet
of

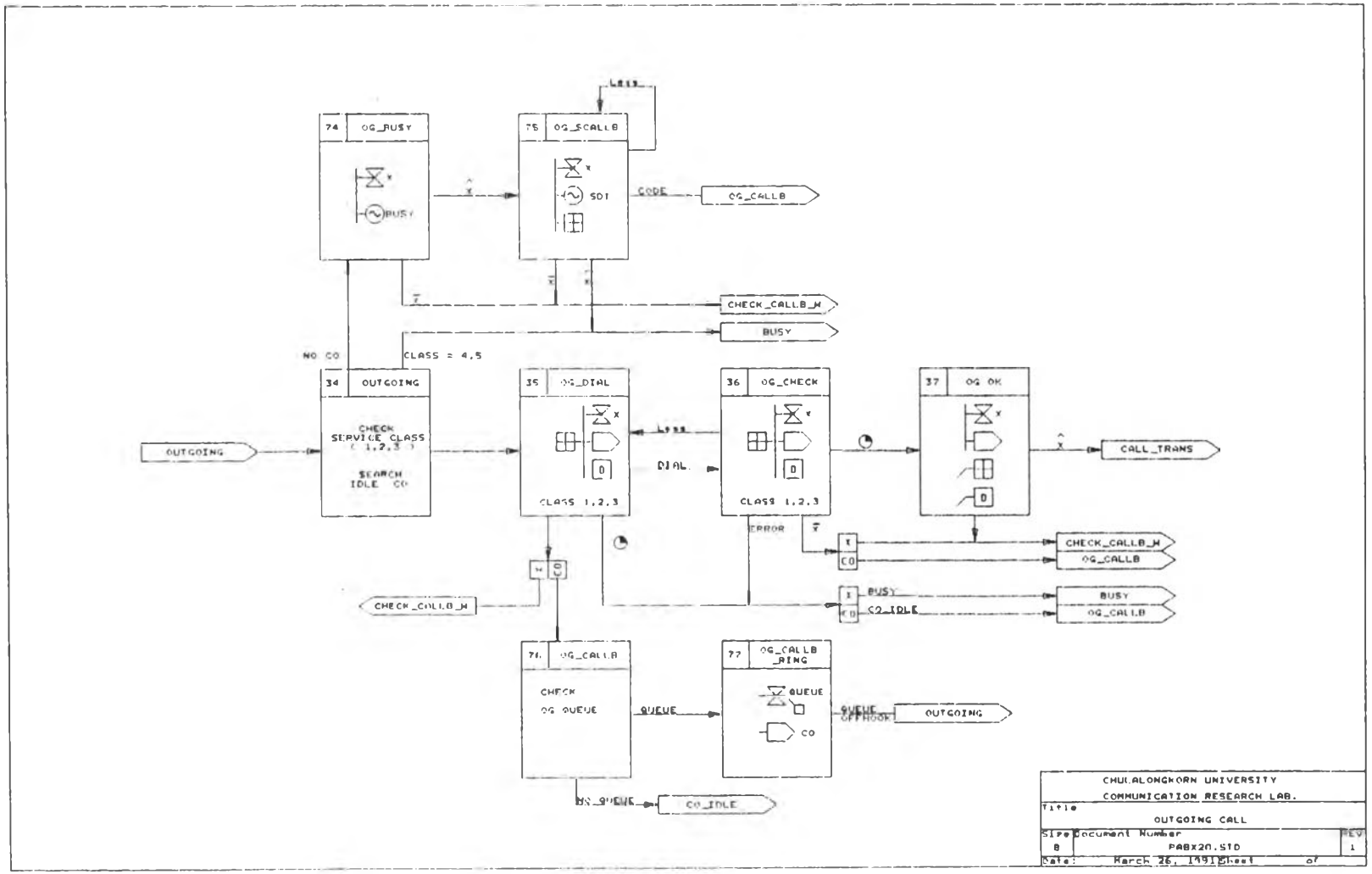

Less
21
S_HOTLINE
S_HOTLINE
SIN
FEATURE_SET
SDT
CHECK_CALLB_W
ERROR
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
HOT LINE
Size
A
Document Number
PABX12.STD
REV
1
Date: March 18, 1991 Sheet 12 of

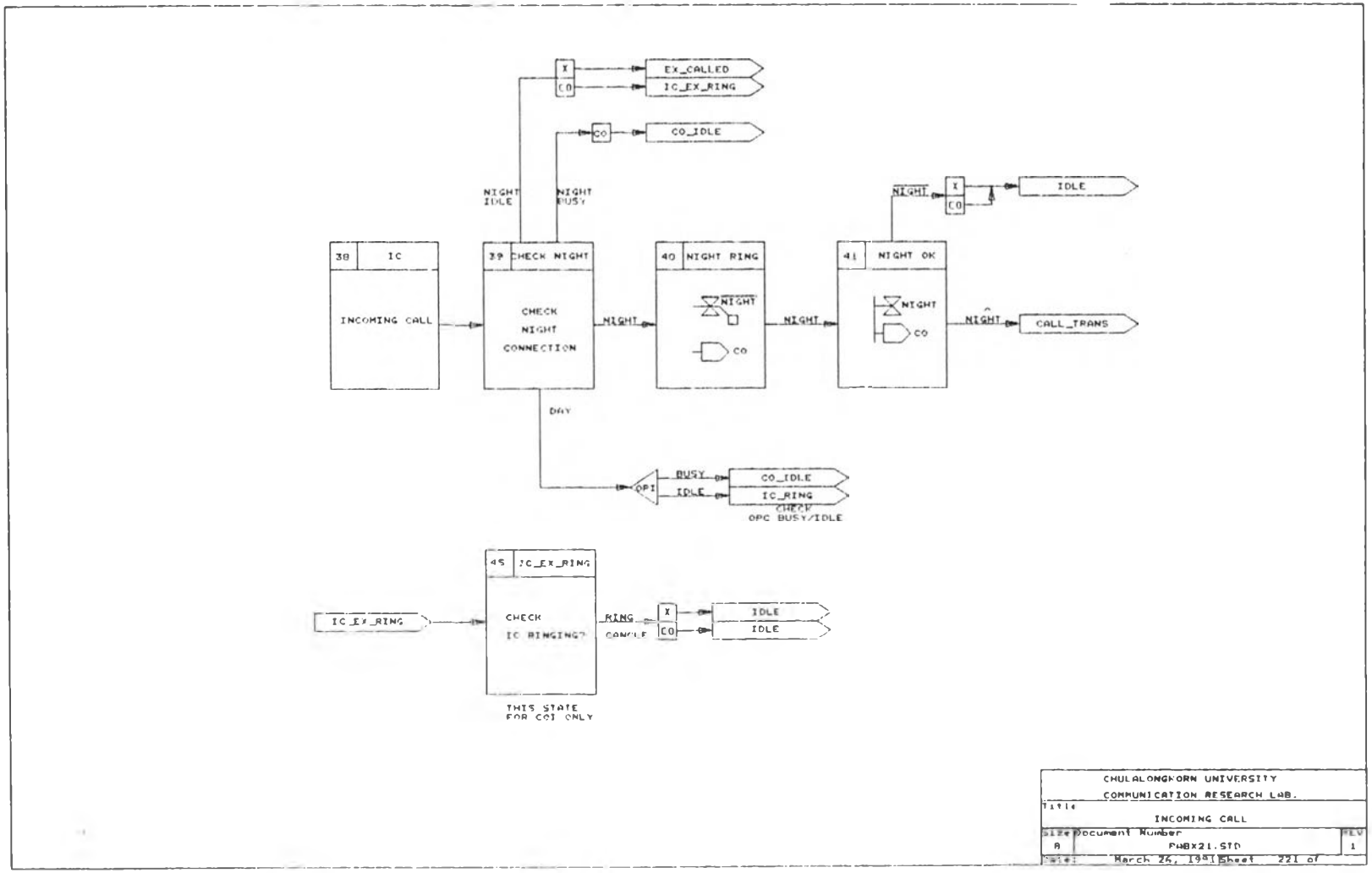

22
C_HOTLINE
C_HOTLINE
CHECK
HOTLINE
SET?
OK
FEATURE_SET
NO
HOTLINE
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
CLEAR HOTLINE
Size
A
Document Number
PABX13.STD
REV
1
Date: March 18, 1991 Sheet 13 of

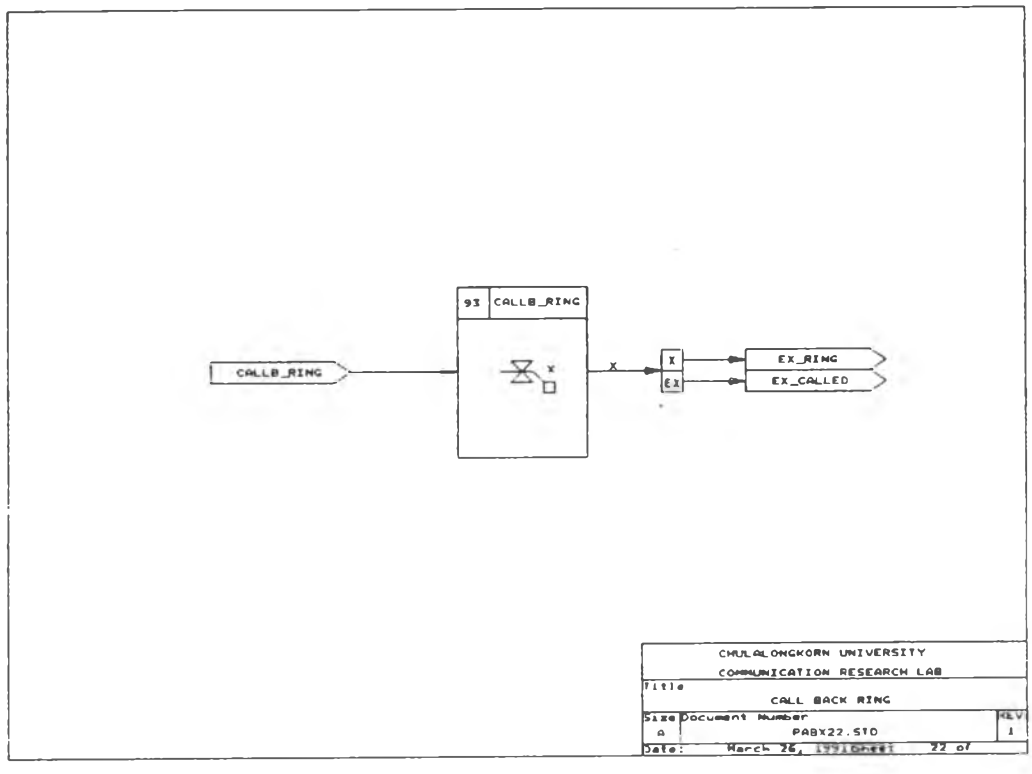

23
OPERATOR
X
RBT
OPER
NO
IDLE OPER
BUSY
OPER
offhook
BUSY
TALK
OPERATOR
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
OPERATOR
Size
A
Document Number
PABX14.STD
REV
1
Date:
March 26, 1991
Sheet
14 of

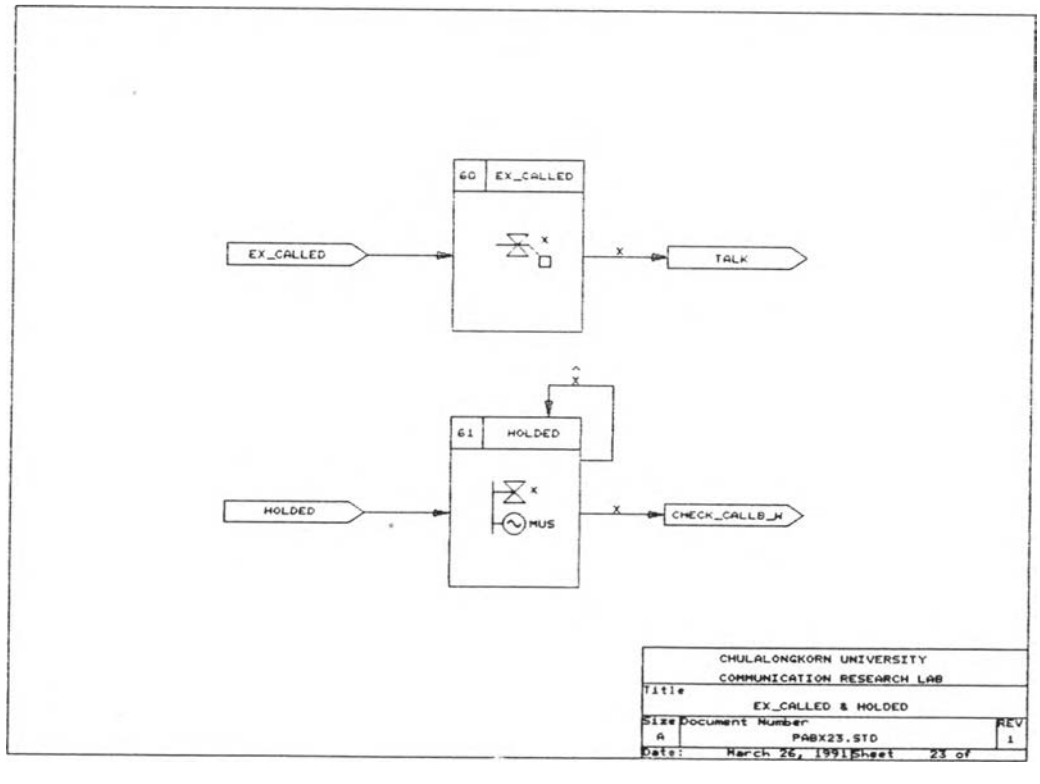

24
REDIAL
CHECK
NUM=EXT/CO
REDIAL
INTERNAL
CALL
CALL_EX
ERROP
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION LAB.
Title
REDIAL
Size
A
Document Number
PABX15.STD
REV
1
Date:
March 26, 1991
Sheet
15 of

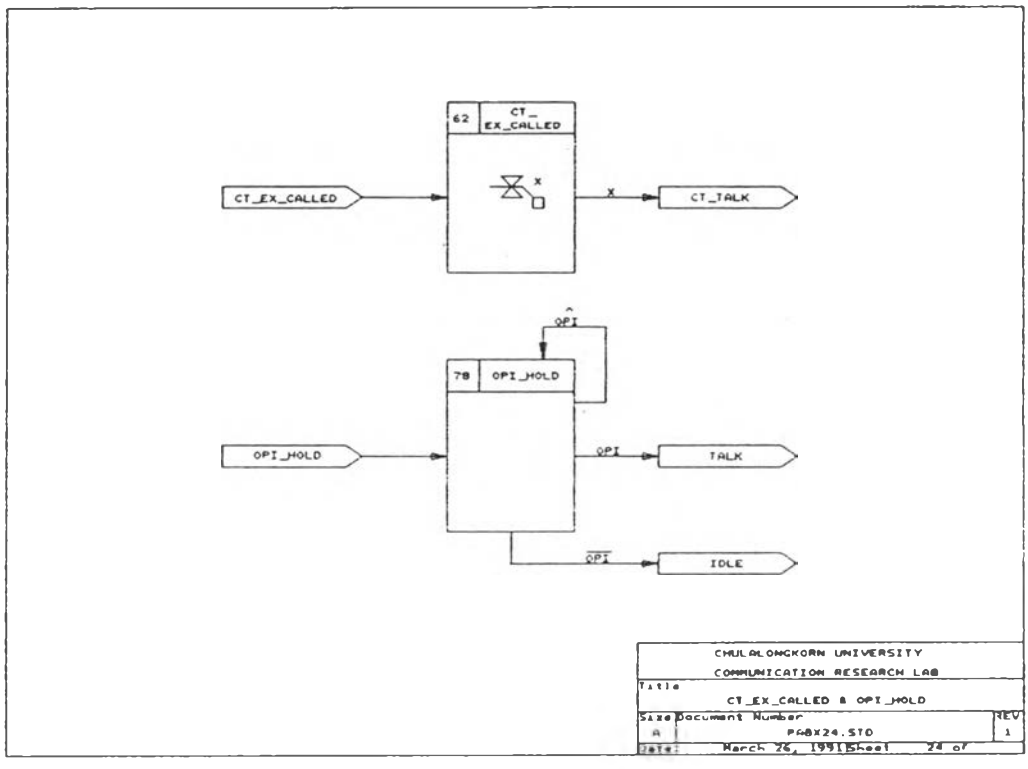

LESS
30
CALL_PU
X
DT
CALL_PU
SIN
31
PU_CHECK
CHECK
DESTINATION
RINGING?
YES
Y
EX
TALK
NO
BUSY
X
CHECK_CALLB_H
X
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
CALL PICKUP
Size
A
Document Number
PABX18.STD
REV
1
Date:
March 26, 1991
Sheet
18 of

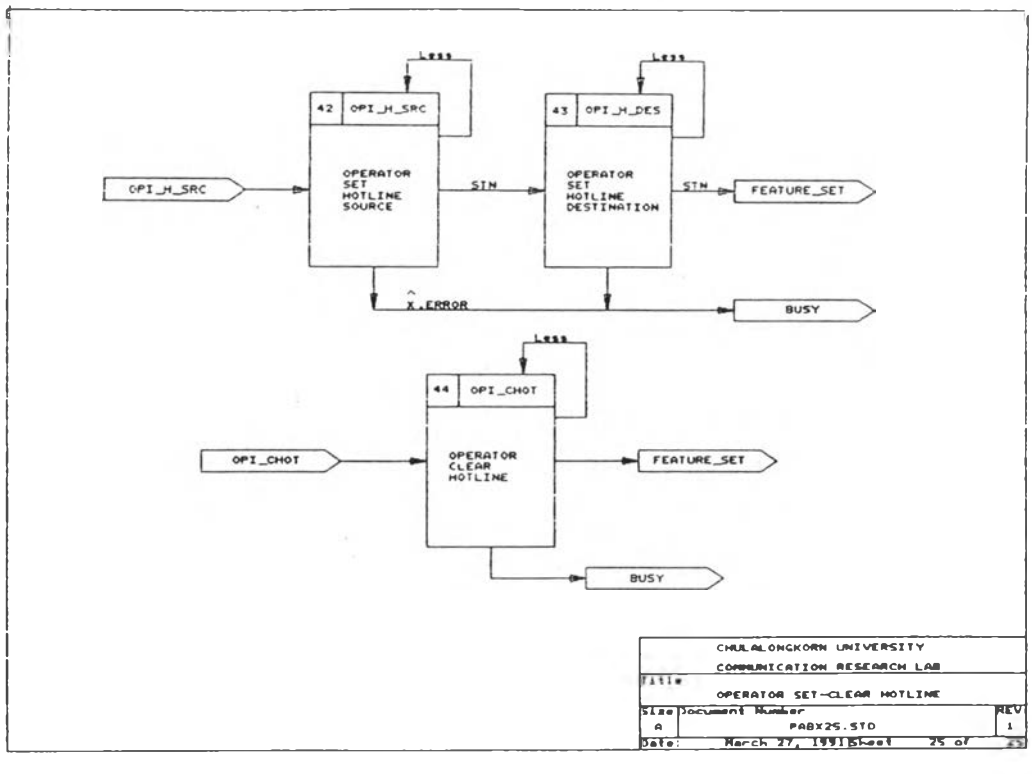

74 OG_BUSY
BUSY
75 OG_SCALLB
SDT
CODE
OG_CALLB
CHECK_CALLB_H
BUSY
NO CO
CLASS = 4,5
34 OUTGOING
CHECK SERVICE CLASS (1,2,3)
SEARCH IDLE CO
OUTGOING
35 OG_DIAL
CLASS 1,2,3
DIAL
36 OG_CHECK
CLASS 1,2,3
ERROR
37 OG OK
CALL_TRANS
CHECK_CALLB_H
OG_CALLB
BUSY
CO_IDLE
CHECK_CALLB_H
76 OG_CALLB
CHECK OG QUEUE
QUEUE
77 OG_CALLB _RING
QUEUE
CO
QUEUE OFF HOOK
OUTGOING
NO QUEUE
CO_IDLE
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
OUTGOING CALL
Size B
Document Number
PABX20.STD
REV 1
Date: March 26, 1991 Sheet of


X
CO
EX_CALLED
IC_EX_RING
CO
CO_IDLE
NIGHT
IDLE
NIGHT
BUSY
NIGHT
X
CO
IDLE
38 IC
INCOMING CALL
39 CHECK NIGHT
CHECK
NIGHT
CONNECTION
NIGHT
40 NIGHT RING
NIGHT
CO
NIGHT
41 NIGHT OK
NIGHT
CO
NIGHT
CALL_TRANS
DAY
OPI
BUSY
IDLE
CO_IDLE
IC_RING
CHECK
OPC BUSY/IDLE
45 IC_EX_RING
IC_EX_RING
CHECK
IC RINGING?
RING
CANCLE
X
CO
IDLE
IDLE
THIS STATE
FOR COI ONLY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB.
Title
INCOMING CALL
Size
A
Document Number
PMBX21.STD
REV
1
Date:
March 24, 1991
Sheet
221 of


93
CALLB_RING
CALLB_RING
X
X
X
EX
EX_RING
EX_CALLED
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
CALL BACK RING
Size
A
Document Number
PABX22.STD
REV
1
Date: March 26, 1991 Sheet 22 of


60
EX_CALLED
EX_CALLED
X
X
TALK
61
HOLDED
HOLDED
X
MUS
X
CHECK_CALLB_H
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
EX_CALLED & HOLDED
Size
A
Document Number
PABX23.STD
REV
1
Date: March 26, 1991 Sheet 23 of


62
CT_
EX_CALLED
CT_EX_CALLED
X
X
CT_TALK
OPI
78
OPI_HOLD
OPI_HOLD
OPI
TALK
OPI
IDLE
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
CT_EX_CALLED & OPI_HOLD
Size
A
Document Number
PABX24.STD
REV
1
Date: March 26, 1991 Sheet 24 of


Less
42
OPI_H_SRC
OPERATOR
SET
HOTLINE
SOURCE
Less
43
OPI_H_DES
OPERATOR
SET
HOTLINE
DESTINATION
OPI_H_SRC
SIN
SIN
FEATURE_SET
X.ERROR
BUSY
Less
44
OPI_CHOT
OPERATOR
CLEAR
HOTLINE
OPI_CHOT
FEATURE_SET
BUSY
CHULALONGKORN UNIVERSITY
COMMUNICATION RESEARCH LAB
Title
OPERATOR SET-CLEAR HOTLINE
Size
A
Document Number
PABX25.STD
REV
1
Date: March 27, 1991 Sheet 25 of

ภาคผนวก ข.

บริการพิเศษ

โปรแกรมควบคุมตู้ชุมสายโทรศัพท์อัตโนมัติที่พัฒนาขึ้นนี้ มีความสามารถให้บริการพิเศษต่างๆ แก่ผู้ใช้โทรศัพท์สายภายในทั้งหมด 8 บริการด้วยกัน คือ

1. บริการจองสายปลายทาง (Call back)
2. บริการฝากสายโดยการหมุนหมายเลข (Call forward dial)
3. บริการฝากสายเมื่อสายไม่ว่าง (Call forward all call)
4. บริการฝากสายเมื่อสายไม่ว่าง (Call forward busy)
5. บริการรอสายปลายทาง (Call waiting)
6. บริการรับสายแทน (Call pick up)
7. บริการสายด่วน (Hot line)
8. บริการติดต่อรับส่งข้อมูลแบบอะซิงโครนัส (DCI connection)

การให้บริการพิเศษทั้ง 8 บริการนี้มีรหัสการใช้งาน และรหัสการยกเลิกการใช้งานซึ่งจะถูกกำหนดหรือเปลี่ยนแปลงค่าโดยอาศัยโปรแกรมเมนทีแนนซ์และแอดมินนิสเตรชั่น ซึ่งในการทดสอบการใช้งานมีการตั้งค่ารหัสต่างๆ ดังแสดงในรูปที่ ข.1

Feature	Set	Cancel
Call Back	70	80
Call Forward Dial	71	81
Call Forward All Call	72	82
Call Forward Busy	73	83
Call Forward No Ans	74	84
Call Waiting	75	85
Call Pickup	76	0
Hot Line	77	87
Speed Dial SET	78	88
DCI Connection	79	89

รูปที่ ข.1 รหัสการใช้งานและการยกเลิกบริการพิเศษต่าง ๆ

รหัสการใช้งานและการยกเลิกการใช้บริการต่าง ๆ สามารถแก้ไขได้โดยอาศัยคำสั่ง ASPA (Assignment of Special Access Code) ในโปรแกรมเมนที่แนนท์และแอดมินนิสเตรชั่น รายละเอียดการใช้บริการพิเศษต่าง ๆ นี้จะกล่าวในหัวข้อต่อไป

ข.1 บริการจองสายปลายทาง (Call back)

การจองสายปลายทางจะมีการใช้งานเมื่อโทรศัพท์ต้นทางทำการหมุนหมายเลขปลายทางแล้ว พบว่าสายปลายทางไม่ว่าง การใช้งานบริการนี้ คือ เมื่อมีการหมุนหมายเลขปลายทางแล้ว ได้ยินสัญญาณไม่ว่าง ให้ทำการแฟลชแล้วหมุนหมายเลขรหัสการใช้บริการจองสาย " 70 " จากนั้นวางหูโทรศัพท์ลงเมื่อได้รับสัญญาณตั้งบริการพิเศษ (Service set tone) เมื่อโทรศัพท์ปลายทางเลิกใช้สายและวางหูลง โทรศัพท์ต้นทางที่ทำการจองสายไว้จะมีเสียงกระดิ่งดังขึ้น เมื่อโทรศัพท์ต้นทางยกหูขึ้น จะได้ยินสัญญาณการเรียกสาย (Ring back tone) ขณะเดียวกันจะมีสัญญาณกระดิ่งดังขึ้นที่ปลายทาง การติดต่อจะมีขึ้นเมื่อปลายทางทำการยกหูขึ้น

ข.2 บริการฝากสายโดยการหมุนหมายเลข (Call forward dial)

ระหว่างที่ผู้ใช้งานโทรศัพท์ไม่อยู่ ผู้ใช้สามารถทำการฝากสายไว้กับโทรศัพท์หมายเลขอื่น ๆ ได้ หากมีการเรียกสายมายังโทรศัพท์เครื่องที่ทำการฝากสายไว้ จะมีเสียงกระดิ่งสั้น ๆ หนึ่งครั้งดังที่โทรศัพท์นั้น จากนั้นสายจะถูกโอนไปยังหมายเลขปลายทางถูกฝากสายอยู่ โดยจะมีเสียงกระดิ่งดังขึ้นแทน ตัวอย่างการใช้งานบริการฝากสาย เช่น โทรศัพท์หมายเลข 2.1 จะทำการฝากสายไว้กับปลายทางหมายเลข 205 มีการทำงานตามขั้นตอนต่อไปนี้

1. โทรศัพท์เครื่องใดก็ได้ในระบบ ทำการยกหูขึ้น แล้วหมุนรหัส "71"

2. เมื่อได้ยินสัญญาณหมุนแบบพิเศษ (Special dial tone) ให้หมุนหมายเลขต้นทาง คือ 201

3. หลังจากได้ยินสัญญาณหมุนแบบพิเศษอีกครั้ง ให้ทำการหมุนหมายเลขปลายทางที่จะถูกฝากสายไว้

4. รอจนกระทั่งมีสัญญาณตั้งบริการพิเศษ (Service set tone) จึงวางสายลง

5. การยกเลิกการฝากสายชนิดนี้ ให้หมุนรหัสยกเลิก 81 แล้วตามด้วยหมายเลขปลายทาง

ข.3 บริการฝากสาย (Call forward all calls)

บริการฝากสายชนิดนี้คล้ายกับการฝากสายโดยหมุนหมายเลขปลายทาง แต่ต่างกันที่โทรศัพท์ที่จะฝากสายจะต้องหมุนหมายเลขรหัสต่าง ๆ เองเท่านั้น ตัวอย่างการฝากสายชนิดนี้ เช่น การฝากสายของโทรศัพท์หมายเลข 201 กับโทรศัพท์หมายเลข 205

1. ยกหูโทรศัพท์หมายเลข 201 ขึ้น แล้วหมุนรหัสการฝากสาย 72
2. หมุนหมายเลขปลายทางเมื่อได้ยินสัญญาณหมุนแบบพิเศษ
3. วางหูลงเมื่อได้ยินสัญญาณตั้งบริการพิเศษ
4. การยกเลิกกระทำโดยการหมุนรหัสยกเลิก 82 แล้วตามด้วยหมายเลขปลายทาง

ข.4 บริการฝากสายเมื่อสายไม่ว่าง (Call foward busy)

การฝากสายเมื่อสายไม่ว่างนี้ มีขั้นตอนการตั้งบริการเหมือนกับการฝากสายในหัวข้อที่แล้ว แต่การให้บริการจะต่างกันไป คือ ขณะที่โทรศัพท์ที่มีการใช้บริการนี้อยู่มีการเรียกสายจากโทรศัพท์เครื่องอื่น โดยที่สายโทรศัพท์ว่างไม่มีการใช้งาน จะมีกระดิ่งดังขึ้นและจะมีการติดต่อเกิดขึ้นได้ แต่ถ้าโทรศัพท์ไม่ว่าง จะมีการโอนสายปลายทางที่ถูกฝากสายไว้ทันที

ข.5 บริการรอสายปลายทาง (Call waiting)

บริการพิเศษรอสายปลายทาง จะถูกเรียกใช้ในกรณีที่ผู้ใช้หมุนหมายเลขเรียกปลายทางแล้วพบว่าสายไม่ว่าง และผู้ใช้ต้องการรอสายปลายทางจนกระทั่งปลายทางวางหู เมื่อผู้ใช้เรียกใช้บริการรอสายปลายทางแล้ว โทรศัพท์ที่เรียกใช้บริการจะถูกติดตั้งอยู่ในสถานะพักสายโดยจะมีเสียงดนตรีดังขึ้น จนกระทั่งปลายทางวางสายลงจะมีเสียงกระดิ่งดังขึ้นที่ปลายทาง เมื่อโทรศัพท์ปลายทางยกหูขึ้นก็จะมีการติดต่อขึ้นทันที ตัวอย่างขั้นตอนการใช้งานมีดังต่อไปนี้

1. โทรศัพท์ต้นทางทำการยกหูขึ้นหมุนหมายเลขปลายทาง เช่น หมุนหมายเลข 201 เป็นต้น แต่เมื่อสิ้นสุดการหมุนมีสัญญาณสายไม่ว่างดังขึ้น
2. ผู้ใช้โทรศัพท์ต้องการรอจนกระทั่งปลายทางวางสายลง โดยใช้บริการพิเศษรอสายหมุน ทำการแฟลชสายต้นทางแล้วหมุนรหัสการรอสายปลายทาง "75" จะมีเสียงดนตรีดังขึ้น

3. จนกระทั่งโทรศัพท์ปลายทางวางสายลง จะมีสัญญาณกระดิ่งดังขึ้นที่ปลายทาง และจะมีเสียงสัญญาณการเรียกสายดังขึ้นที่ต้นทาง

4. เมื่อโทรศัพท์ปลายทางทำการยกหูขึ้น ก็จะมีการติดต่อสื่อสารเกิดขึ้น

5. การยกเลิกการใช้บริการทำได้โดยการวางสายต้นทางลงเท่านั้น

ข.6 บริการรับสายแทน (Call Pickup)

บริการรับสายแทนใช้เมื่อมีโทรศัพท์เครื่องใดเครื่องหนึ่งมีเสียงกระดิ่งดังขึ้น ผู้ใช้โทรศัพท์เครื่องนั้นไม่อยู่ ผู้ใช้อื่นที่อยู่ในบริเวณใกล้เคียงสามารถทำการรับสายแทนได้ โดยหมุนรหัสการรับสายแทนเท่านั้น ตัวอย่างการใช้งานมีขั้นตอนดังต่อไปนี้

1. โทรศัพท์หมายเลข 201 มีเสียงกระดิ่งดังขึ้น โดยไม่มีผู้รับสาย

2. ผู้ใช้โทรศัพท์อยู่ใกล้เคียงสามารถทำการรับสายแทน โดยการยกหูโทรศัพท์ขึ้นแล้วหมุนรหัสการรับสายแทน "76"

3. เมื่อหมุนรหัสรับสายแทนแล้ว หมุนหมายเลขปลายทางของโทรศัพท์ที่มีเสียงกระดิ่งดังขึ้น การติดต่อสื่อสารจะเกิดขึ้นทันทีที่หมุนหมายเลขปลายทางจบ เสียงกระดิ่งจะหายไป

ข.7 บริการสายด่วน (Hotline)

บริการสายด่วนมีไว้ให้บริการในกรณีที่ผู้ใช้โทรศัพท์ต้องการติดต่อกับหมายเลขปลายทางหมายเลขใดหมายเลขหนึ่ง ได้ทันทีที่มีการยกหูโทรศัพท์ขึ้น โดยไม่ต้องมีการหมุนหมายเลขปลายทางเลย ตัวอย่างการใช้บริการพิเศษมีดังต่อไปนี้

1. ยกโทรศัพท์ต้นทางที่ต้องการตั้งบริการสายด่วนขึ้น หมุนรหัสการตั้งบริการสายด่วน "77" แล้วตามด้วยหมายเลขปลายทางที่ต้องการติดต่อด้วย

2. หลังจากตั้งบริการสายด่วนแล้ววางหูโทรศัพท์ลง เมื่อโทรศัพท์เครื่องนี้ยกหูโทรศัพท์ขึ้นจะมีการเรียกสายไปยังปลายทางที่ได้ตั้งไว้ทันที โดยจะมีสัญญาณการเรียกสายดังขึ้นที่ต้นทาง และมีเสียงการดังดังที่ปลายทาง

3. การยกเลิกการใช้บริการทำได้โดย ยกหูโทรศัพท์ขึ้นทำการแฟลชแล้วหมุนรหัสยกเลิกการใช้บริการสายด่วน "87" รอจนกระทั่งมีสัญญาณตั้งบริการพิเศษจึงวางสายลง

ข.8 บริการติดต่อรับส่งข้อมูลแบบอะซิงโครนัส

บริการติดต่อรับส่งข้อมูลแบบอะซิงโครนัสนี้ ใช้สำหรับการต่อช่องเวลาการใช้งานของหน่วยเชื่อมโยงอุปกรณ์สื่อสารแบบอะซิงโครนัสเข้าหากัน ตัวอย่างการใช้งานมีขั้นตอนดังต่อไปนี้

1. ยกหูโทรศัพท์เครื่องใดเครื่องหนึ่งขึ้น ทำการหมุนรหัสการใช้บริการติดต่อรับส่งข้อมูลแบบอะซิงโครนัส "79" รอจนกระทั่งมีสัญญาณหมุนแบบพิเศษ

2. หมุนหมายเลขต้นทางของหน่วยเชื่อมโยงอุปกรณ์สื่อสารแบบอะซิงโครนัส รอจนกระทั่งมีสัญญาณหมุนแบบพิเศษดังขึ้นอีกครั้ง

3. หมุนหมายเลขปลายทางของหน่วยเชื่อมโยงอุปกรณ์สื่อสารแบบอะซิงโครนัส เมื่อมีสัญญาณตั้งบริการพิเศษจึงวางสายลง

4. การยกเลิกการติดต่อสื่อสารกระทำได้โดยหมุนรหัสยกเลิกการติดต่อ "89" แล้วตามด้วยหมายเลขต้นทางของหน่วยเชื่อมโยงอุปกรณ์สื่อสารแบบอะซิงโครนัส

รหัสการใช้งานบริการพิเศษ และการยกเลิกบริการพิเศษที่ได้ยกตัวอย่างขึ้นมานี้เป็นรหัสการใช้งานที่ใช้ในการทดสอบการทำงานของโปรแกรมเท่านั้น ซึ่งรหัสเหล่านี้สามารถแก้ไขเปลี่ยนแปลงได้โดยอาศัยโปรแกรมเมนทีแนนซ์และแอดมินนิสเตรชัน

ภาคผนวก ค

โปรแกรมควบคุมตู้ชุมสายโทรศัพท์อัตโนมัติระบบดิจิตอล 256 พอร์ต

```
/*-------------------------------------------------
 *   MODULE:  MPUVAR.H
 *   REL    :  1.0
 *   AUTHOR:  Sunasak U.
 *   DATE   :  26 December 1990
 *   Description : variable header file
 *-------------------------------------------------
 */

#include "DOS.H"
#include <stdio.h>

#define status_line     22
#define MAX_DEV         256
#define MAX_PORT        255
#define MAX_BUSY        64
#define MAX_IO          32
#define MAX_TIMER       32
#define MAX_TIMER2      32
#define MAX_CONF        32
#define MAX_MESSAGE     32
#define MAX_INT         8
#define MAX_NIOINT      8
#define MAX_FEATURE     16
#define MAX_SIO         2
#define MAX_BUFFER      8192
#define MAX_LOG         5
#define MAX_QUEUE       256
#define max_command     45   /* Maximum command */
/* #define MAX_MODULE      64 */
#define DATA_FILENAME   "PROTOCOL.DAT"
#define IO_FILE1        "IO1.DAT"
#define IO_FILE2        "IO2.DAT"
#define NULL_ADDRESS    64
#define NULL_DATA       0xff

#define ON      1
#define OFF     0
#define YES     1
#define NO      0
#define TRUE    1
#define FALSE   0

#define byte unsigned char
#define uint unsigned int

extern unsigned unsigned char  module_table[max_port];

#define sio_rook    4
#define sio_flash   0
#define sio_digit   0
#define dev_digit   0
#define io1_rook    3
#define io1_ring    0
#define io1_sio     0
#define dual_free   8

struct mat_boi {
                    byte dev_no;
                    uint sio;
                    byte pad;
                    byte mod;
                    uint pos;
                    byte svc;
                    uint noc;
                    byte func;
                    byte io_type;
                    byte tmp;
                };

extern struct mat_boi Dev_table[MAX_PORT];

struct data_boi {
                    byte dev_no;
                    uint pos;
                    byte speed;
                    byte state;
                    byte status;
                    uint down_cnt;
                    byte a;
```

```
                byte b;
                byte c;
                byte dei_hold;
                byte *pdial;
                byte dial_cnt;
                byte dial[15];
                byte *presi;
                byte resi_cnt;
                byte resi[15];
                byte dev_called;
                byte dev_hold;
                byte dev_dtmf;
                byte dev_co;
                byte dev_callo;
                byte dev_calldes;
                byte dev_callf;
                byte dev_callfdes;
                byte dev_callfno;
                byte dev_callwait;
                byte not;
                byte hunt;
                byte svc;
                unsigned int svn;
                byte feat;
                byte src;
                byte two;
                struct time start;
              };

extern  struct data_tbl DEV[MAX_PORT+1];

extern unsigned char  Hot_tbl[MAX_SLOT];
extern unsigned char  Code_set[MAX_FEATURE];
extern unsigned char  Code_cnl[MAX_FEATURE];
extern unsigned char  Speed_tbl[MAX_SPEEDDIAL][15];
extern unsigned char  Tsi_add;
extern unsigned char  Tsi_data[MAX_PORT+1];
extern unsigned char  Co_cxo,Co_code;
extern unsigned int   night[MAX_NIGHT];
extern unsigned char  night[MAX_NIGHT] night_[illegible];
extern unsigned char  Isel,Ico,Idtmf,Itsi,Idev;
extern struct data_tbl *pdev,*pdevtmp,*pctmp,[illegible]

extern unsigned char  command[MAX_INFCOM];
extern unsigned char  *pcom;
extern unsigned char  n_in,n_out,address;
extern unsigned char  Dtmf_add[MAX_DTMF],Co_add[MAX_CO];
extern unsigned char  Dev_add[MAX_DEV],Cont_add[MAX_CONT];
extern unsigned char  num_dev,num_dtmf,num_co,[illegible]
extern unsigned char  Qdev[MAX_DEV],Qcnt,*pq;
extern unsigned char  Qdev2[MAX_DEV],Qcnt2,*pq2;
extern unsigned char  cod[MAX_COD],cod_cnt,[illegible]

struct og_data  {
                uint stn;
                struct date today;
                struct time now;
                unsigned char number[15];
              };
extern struct og_data OGbuff;
extern FILE *fpr,*prn;
extern unsigned char print;
```

```
/*-----------------------------------------
 *   MODULE:  MPUSTATE.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  23 December 1990
 *   DESCRIPTION : SSI State Definition
 * ----------------------------------------
 */

#define ONHOOK      0
#define OFFHOOK     1

#define OFF         0
#define IDLE        1
#define CHK_CLASS   2
#define DIAL        3
#define COMMAND     4
#define NEXT        5
#define BUSY        6
#define LOCK        7

#define CALL_EX     8
#define EX_RING     9
#define TALK        10
#define FLASH       11
#define FLASH_DIAL  111
#define CALL_TRANS  12
#define CT_RING     121
#define CT_TALK     122
#define CT_FLASH    123

#define HOLD_BUSY   13
#define EX_HOLDED   14

#define S_CALLF     15
#define CALLF_DIAL  151
#define CALLF_CHECK 152
#define CALLF_OK    153

#define C_CALLF     16

#define EX_BUSY     17

#define S_CALLB     18
#define S_CALLB2    19
#define CALLB_DIAL  181
#define CALLB_OK    182
#define C_CALLB     20

#define S_HOTLINE   21
#define HOT_DIAL    211
#define HOT_OK      212
#define C_HOTLINE   22

#define OPERATOR    23

#define REDIAL      24

#define HOLD        25
#define HOLD_DIAL   251
#define HOLD_CALLEX 252
#define HOLD_RING   253
#define HOLD_TALK   254

#define CONFERENCE  26
#define CONF_DIAL   27
#define CONF_CHECK  28
#define CONF_SET    29

#define CALL_PU     30
#define PU_CHECK    31

#define HOTLINE     32
#define HOTLINE_CNL 33

#define OUTGOING    34
#define OG_DIAL     35
#define OG_CHECK    36
#define OG_OK       37

#define IC          38
#define CHECK_NIGHT 39
#define NIGHT_RING  40
#define NIGHT_OK    41

#define OPI_H_SRC   42
#define OPI_H_DES   43
#define OPI_CHOT    44

#define IC_EX_RING  45
#define CAMP_ON     47
#define CO_TALK     46


#define COI_ONHOOK  50
#define COI_OFFHOOK 51
#define COI_FLASH   52
#define COI_DIAL    53

#define EX_CALLED   60
#define HOLDED      61
#define CT_EX_CALLED 62

#define DIAL_SPEED  63

#define SSPEED      64
#define SSPEED_DIAL 65
#define SSPEED_OK   66

#define CSPEED      67
#define CSPEED_DIAL 68
#define CSPEED_OK   69

#define CONOCI      70
#define CONOCI_SRC  71
#define CONOCI_DES  72

#define DISOCI      73

#define OG_BUSY     74
#define OG_SCALLB   75
#define OG_CALLB    76
#define OG_CALLB_RING 77

#define OPI_HOLD    78

#define FEATURE_SET 80

#define SCALLFD     81
#define SCALLFD_SRC 82
#define SCALLFD_DES 83
#define CCALLFD     84

#define CHECK_CALLFBUSY 90
#define CT_CHECKCFD     91
#define CHECK_CALLB_W   92
#define CALLB_RING      93
```

```
/*-------------------------------------
 *   MODULE:  MPUCMD.H
 *   REL    :  1.0
 *   by     :  Surasak Uthayopas
 *   DATE   :  28 December 1990
 *   DESCRIPTION: -MPU Command Control
 *                -Const of Tone timeslot
 *                -Device Number
 *-------------------------------------
 */

#define      chk_dev         0x20

/*
#define      _mpu_dvn        0x01
#define      _ssi_dvn        0x03
#define      _tsi_dvn        0x02
#define      _ssi_dvn        0x03
#define      _coi_dvn        0x04
#define      _dtmf_dvn       0x05
#define      _isdn_dvn       0x07
#define      _dci_dvn        0x08
#define      _opi_dvn        0x09
#define      _conf_dvn       0x0b
*/

#define      MPU             0x01
#define      TSI             0x02
#define      SSI             0x03
#define      COI             0x04
#define      DTMF            0x06
#define      ISDN            0x07
#define      DCI             0x08
#define      OPI             0x09
#define      CONF            0x0B
#define      NODEV           0x0ff

#define      _set_tsac       0x30
#define      _set_pcm        0x31

#define      _ssi_offh       0x32
#define      _ssi_onh        0x33
#define      _ssi_fl         0x34
#define      _ssi_dp         0x35
#define      _ssi_tr         0x36
#define      _ssi_enr        0x37
#define      _ssi_disr       0x38
#define      _ssi_test       0x39
#define      _ssi_click      0x27

#define      _tsi_clrtsi     0x21
#define      _tsi_ctone      0x23
#define      _tsi_exts       0x24
#define      _tsi_test       0x25

#define      tone_dial       0x00
#define      tone_ringb      0x01
#define      tone_busy       0x02
#define      tone_null       0x03
#define      tone_sdt        0x04
#define      tone_rot        0x05
#define      tone_sst        0x06
#define      tone_warn       0x07
#define      tone_hold       0x08
#define      tone_camp       0x09
#define      tone_music      0x0a
#define      tone_dtmf       0x0a

#define      _coi_onh        0x41
#define      _coi_offh       0x42
#define      _coi_fl         0x43
#define      _coi_dial       0x44
#define      _coi_ring       0x45
#define      _coi_drdy       0x46
#define      _coi_rcnl       0x047

#define      _opi_enr        _ssi_enr
#define      _opi_disr       _ssi_disr
#define      _opi_onh        _coi_onh
#define      _opi_offh       _coi_offh
```

```
#define     _opi_fl          _coi_fl
#define     _opi_dial        _coi_dial
#define     _opi_hold        0xa0
#define     _opi_cnl         _coi_rcnl

#define     _dtmf_digit      0x60
```

```
/*-----------------------------------------
 *   MODULE:  MPU001.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  27 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpucmd.h"
#include "mpuintf.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "dos.h"


void main(){

    unsigned char *p,flag,ch;

    clrscr();
    puts("\n\t\t\t==============================");
    puts("   \t\t\t Digital PABX Control Program ");
    puts("   \t\t\t==============================\n");
    printf("\nMEM %d\n",sizeof(DEV)+sizeof (Dev_tbl),sizeof(Slot_tbl)+
        sizeof(OGbuff)+sizeof(Qdev)+sizeof(Qdev2));

    init_var();
    reset_allcard();
    scan_card();
    Slot_tbl[0]=CONF;
/*  Slot_tbl[1]=OPI; */

    list_card();
    Num_dev=init_data();

    init_pcmtsac();
    init_TSI();
    init_data_MAT();

    Num_dtmf=init_DTMF();
    Num_co  =init_COI();
    Num_opi =init_OPI();
    Num_conf=init_CONF();
    INIT_NIGHT();
    remove(OG_FILE2);
    rename(OG_FILE1,OG_FILE2);
    fpt=fopen(OG_FILE1,"w+");

    Qcnt=0, Qcnt2=0, flag=0;
    ch=0;

    printf("\n\n> Print Outgoing Report : ");
    if ( (toupper(getch()))=='Y'){
        print=ON;
        prn=fopen("PRN","w");
        fprintf(prn,"\nOUTGOING CALL REPORT\n");
        fprintf(prn,"--------------------\n");
        fprintf(prn,"by SURASAK UTHAYOPAS\n");
        getdate(&OGbuff.today);
        gettime(&OGbuff.now);
        fprintf(prn,"START > DATE %2d/%2d/%2d  TIME %02d:%02d-%02d:%02d\n\n",
            OGbuff.today.da_mon,OGbuff.today.da_day,OGbuff.today.da_year,
            OGbuff.now.ti_hour,OGbuff.now.ti_min);

    }else print=OFF;
    printf("\n");
    beep(5);

    do{
        scan_status();
        if(Qcnt>0) flag=1;
            else flag=0;
        analyze_state(Qdev,Qcnt);
        analyze_state(Qdev2,Qcnt2);
        if (flag) show_status();
        if (kbhit()) {
```

```
            ch=getch();
            if (ch=='\r'){
                fclose(fpt);
                remove(OG_FILE2);
                rename(OG_FILE1,OG_FILE2);
                fpt=fopen(OG_FILE1,"w+");
            }
        }
    }while (ch!=32);

    fprintf(prn,"\n\n\n");

    fclose(fpt);
    fclose(prn);

    reset_allcard();
    show_co();
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU002.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

/* #define PRINT_SND */

#include <stdio.h>
#include <dos.h>
#include "mpu.h"
#include "mpuvar.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpu002.h"
#include "mpu004.h"

void beep(i)
unsigned char i;
{
    char i1;
    for (i1=1;i1<=i;i1++){
        sound(2000);
        delay(100);
        nosound();
        delay(200);
    }
}

void waitkey()
{
    sound(1000),delay(30),nosound();
    puts("\r****** Press a key *******");
    getch();
}

void call_slot(slot)        /* write pm address */
unsigned char slot;
{
    outp(INF_ADD,slot);
    #ifdef DEBUG_ADD
        printf("\n> Write address (%3d)\n",slot);
    #endif
}

void reset_cnt()            /* reset interface address counter */
{
    outp(INF_RST_CNT,NULL);
}

void  H_out(signal,on_off)  /* enable handshake output signal */
unsigned char signal,on_off;
{
   if (on_off) outp(INF_H_OUT,(h_out &=(~signal)));
    else    outp(INF_H_OUT,(h_out |=signal));
}

unsigned char data_rdy()    /* check data_rdy signal */
{
    return(inp(INF_PPU_ENI)==1);
}

unsigned char active(signal)    /* check data_rdy signal */
unsigned char signal;
{
    return( (inp(INF_H_IN) & signal)==0 );
}

void PPU_EN(on_off)             /* Set ppu enable signal on/off */
unsigned char on_off;
{
    if (on_off) outp(INF_PPU_ENO,0);
        else outp(INF_PPU_ENO,0xff);

}

void  clear_inf()               /* clear interface buffer */
{
```

```
    unsigned char i;

    PPU_EN(OFF);
    reset_cnt();
    for(i=1;i<=MAX_INFCOM ;i++)
         outp(INF_DATA,NULL);
    reset_cnt();
}

#ifdef LIST_INF_DATA

    void commwrite(unsigned char *c1)
    /* Write Command form buffer to interface */
    {
        unsigned char number,i;

        /*  s_disable_pm, reset counter ,
            write command ,reset counter     */

        cprintf("\r>> Write Command to interface <<\n");
        PPU_EN(OFF);
        reset_cnt();
        number=*c1++;
        outp(INF_DATA,number);
        cprintf("\rBYTE COUNT  =%3d\n",number);
        for (i=1;i<=number;i++){
            cprintf("\rCommand[%2d]=%3d\n",i,*c1);
            if ( (i%20)==0) waitkey();
            outp(INF_DATA,*c1++);
        }
        reset_cnt();
    }

    void  commread(unsigned char *c2)
    /* Read Command from Interface to buffer */
    {
        unsigned char number,i;
        /*  s_disable_pm, reset counter,
            read command, reset counter */
        cprintf("\r>> Read Command from interface <<\n");
        PPU_EN(OFF);
        reset_cnt();
        number=*c2=inp(INF_DATA);
        cprintf("\rRead BYTE COUNT=%3d\n",number);
        c2++;
        for (i=1;i<=number;i++){
            *c2=inp(INF_DATA);
            cprintf("\rCommand[%2d] =%3d\n",i,*c2++);
            if ( (i%20)==0) waitkey();
        }
        reset_cnt();
    }

#else

    void commwrite(unsigned char *c1)
    /* Write Command form buffer to interface */
    {
        /*  s_disable_pm, reset counter ,
            write command ,reset counter     */

        unsigned char number,i;

    /*  PPU_EN(OFF); */
        reset_cnt();
        number=*c1++;
        outp(INF_DATA,number);
        for (i=1;i<=number;i++){
            outp(INF_DATA,*c1++);
        }
        reset_cnt();
    }

    void  commread(unsigned char *c2)
    /* Read Command from Interface to buffer */
    {
        unsigned char number,i;

        /*  s_disable_pm, reset counter,
            read command, reset counter */
```

```
        /*  PPU_EN(OFF);  */
        reset_cnt();
        number=*c2++=inp(INF_DATA);
        number=20;
        for (i=1;i<=number;i++){
            *c2++=inp(INF_DATA);
        }
        reset_cnt();
    }
#endif

#ifdef USER_INTERRUPT

    void wait_data_rdy(){
        while (!data_rdy() && !kbhit()  );
        if (kbhit()){
            getch();
            puts(">> User Interupt\n\r");
        }
    }

#else

    void wait_data_rdy()
    {
    /* Wait data ready and count down */
        unsigned char count=20;

        while (!data_rdy() && (count>0) ){
            count--;
            delay(1);
        }
    }

#endif

void wait_data_rdy2()
{
    unsigned char count=100;

    while (!data_rdy() && (count>0) ){
        count--;
        delay(1);
    }
}

unsigned char check_dev(slot)
unsigned char slot;
{
/* Check Dev Number */
    unsigned char i;

    pcom=command;
    *pcom++=3,    *pcom++= chk_dev;
    *pcom++=NULL, *pcom++=NULL;
    commwrite(command);
    call_slot(slot);
    PPU_EN(ON);
    wait_data_rdy();
    PPU_EN(OFF);
    delay(10);
    if (active(REQ)){
        reset_cnt();
        H_out(REQ_ACK,ON);
        PPU_EN(ON);
        wait_data_rdy();
        PPU_EN(OFF);
        H_out(REQ_ACK,OFF);
        commread(command);
        i=command[1];
    }else i=NODEV;
    return(i);
}


void compare_card()
{
    unsigned char index,data;

    #ifdef DEBUG
```

```
        printf("\n>> COMPARE Card...\n");
    #endif
    for (index=0;index<MAX_SLOT;index++){
        call_slot(index);
        if ((data=check_dev(index))!=Slot_tbl[index] ) {
            printf("> ERROR.. Card Type mismatch! PAD$=%3d  MAT(%3d) != SCAN(%3d)\n",
               index,Slot_tbl[index],data);
        }
    }
}

void list_card()
{
    unsigned char i;

    printf("\n>> DISPLAY CARD DATA\n");
    for (i=0;i<MAX_SLOT;i++){
        if (Slot_tbl[i] !=NODEV) {
            printf("addersss# %3d-> %3d",i,Slot_tbl[i]);
            switch (Slot_tbl[i]){
                case MPU :puts(" > MPU  ");break;
                case TSI :puts(" > TSI  ");break;
                case SSI :puts(" > SSI  ");break;
                case DTMF:puts(" > DTMF ");break;
                case COI :puts(" > COI  ");break;
                case ISDN:puts(" > ISDN ");break;
                case OCI :puts(" > OCI  ");break;
                case CONF:puts(" > CONFERENCE");break;
                case OPI :puts(" > OPI   ");break;
                default:puts("ERROR");
            }
            if (i%20==0 && i!=0) { getch();};
        }
    }

}

void send_data(slot,a,b,c)
unsigned char slot,a,b,c;
{
    #ifdef PRINT_SND
        printf("\rSND>>SLOT%2d ( 3) : 0x%02x(%2d), 0x%02x(%2d), 0x%02x(%2d)\n",
            slot,a,a,b,b,c,c);
    #endif
    pcom=command;
    *pcom++ = 3 ,*pcom++ = a ,*pcom++ = b ,*pcom++ = c;
    commwrite(command);
    call_slot(slot);
    PPU_EN(ON);
    wait_data_rdy();
    PPU_EN(OFF);
    /* clear_inf() */
}

void send_command(slot,comm)
unsigned char slot,*comm;
{
    commwrite(comm);
    call_slot(slot);
    PPU_EN(ON);
    wait_data_rdy();
    PPU_EN(OFF);
}

void show_co()
{
    char ch='\r';

    printf("\n\nOUTGOING CALL REPORT\n");
    printf("    ==================\n");
    if ( (fpt=fopen(OG_FILE1,"r"))==NULL){
        printf("FILE ERROR !\n");
    }else{
        while(ch != EOF){
            putchar(ch);
            ch=getc(fpt);
        }
    }
    fclose(fpt);
}
```

```
//*-----------------------------------------
 *   MODULE:  MPU003.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include <dos.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "mpucmd.h"
#include "mpustate.h"

#define MAX_COI_MOD  3
#define MAX_SSI_MOD  8
#define MAX_DTMF_MOD 8
#define MAX_OPI_MOD  6
#define MAX_DCI_MOD  2
#define MAX_CONF_MOD 6

unsigned char search_dev(pos)
int pos;
{
    unsigned char i;

    pdev=DEV;
    for (i=0;i<MAX_PORT;i++,pdev++){
        if (pdev->pos==pos)
            return(i);
    }
    return(0xff);
}

void init_data_MAT()
/*  Initialize data
 *    - Read data from file
 */
{

    FILE *fpt;
    unsigned char i,tmp;
    unsigned int pos;
    struct mat_tbl *pmat;

    /* Intialize set_table */
    if((fpt=fopen(DATA_FILENAME,"rb+"))==NULL){
        puts (">ERROR! ... DATA File  missing");
        beep(1);
      }else{
        fread(Dev_tbl  ,sizeof(Dev_tbl)  ,1,fpt);
        fread(Code_set ,sizeof(Code_set) ,1,fpt);
        fread(Code_cnl ,sizeof(Code_cnl) ,1,fpt);
        fread(Speed_tbl,sizeof(Speed_tbl[0]),MAX_SPEEDDIAL,fpt);
        fread(&light    ,sizeof(&light)    ,1,fpt);
      }

      fclose(fpt);
      /* Convert data from Dev_tbl to DEV */
      printf("\nInitialize Data From Mat  \n");
      pmat=Dev_tbl;
      for (i=0;i<MAX_PORT;i++,pmat++){
        if (pmat->dev_typ==SSI){
            tmp=search_dev(pmat->pos);
            if (tmp==0xff) printf("\n > ERROR no POS found %d",pmat->pos);
            else{
                pdev=DEV+tmp;
                if (pdev->dev_typ==SSI){
                    pdev->stn=pmat->stn;
                    pdev->svc=pmat->svc;
                    printf("SSI(%02d)   ",pmat->pos>>3);
                }else{
                    printf("\n > ERROR TYPE MISMATCH SSI->%d",pdev->dev_typ);
                }
            }
        }
```

```
        if(pmat->dev_typ==COI){
            tmp=search_dev(pmat->pos);
            if (tmp==0xff) printf("\n > ERROR no POS found %d",pmat->pos);
            else{
                pdev=DEV+tmp;
                if (pdev->dev_typ==COI){
                    pdev->stn = pmat->stn;
                    pdev->redi_cnt = pmat->co_type;
                    printf("COI(%02d)   ",pmat->pos>>3);
                }else{
                    printf("\n > ERROR TYPE MISMATCH COI->%d",pdev->dev_typ);
                }
            }
        }

        if(pmat->dev_typ==DCI){
            tmp=search_dev(pmat->pos);
            if (tmp==0xff) printf("\n > ERROR no POS found %d",pmat->pos);
            else{
                pdev=DEV+tmp;
                if (pdev->dev_typ==DCI){
                    pdev->stn = pmat->stn;
                    printf("DCI(%02d)   ",pmat->pos>>3);
                }else{
                    printf("\n > ERROR TYPE MISMATCH DCI->%d",pdev->dev_typ);
                }
            }
        }
      }
      printf("\n");
}


void reset_card(slot)
unsigned char slot;
/* Reset PPU in each card */
{
    #ifdef DEBUG
        printf("\r >Reset PPU #%2d",slot);
    #endif
    call slot(slot);
    outp(INF_H_OUT,RST_PPU);
/*  H_out(RST_PPU,ON);  */
    PPU_EN(ON);
    delay(20);
    PPU_EN(OFF);
    outp(INF_H_OUT,h_out);
/*  H_out(RST_PPU,OFF); */
}

void reset_allcard()
/* Reset PPU in each card */
{
    unsigned char i;

    #ifdef DEBUG
        printf("\r\n>>Reset All Card\n");
    #endif
    for (i=0;i<MAX_SLOT;reset_card(i++));
}

void scan_card(){
    unsigned char slot;

     #ifdef DEBUG
        printf("\r\n>>Scan Card\n");
        for (slot=0;slot<MAX_SLOT;slot++){
    /*      printf("\r >Scan Slot #%02d",slot); */
            Slot_tbl[slot]=check_dev(slot);
        }
     #else
        for (slot=0;slot<MAX_SLOT;slot++)
            Slot_tbl[slot]=check_dev(slot);
     #endif
}



unsigned char init_data()
/* Program all TSAC in each card */
```

```
{
    unsigned char mod=0;
    unsigned int  slot,dev;

    #ifdef DEBUG
        printf("\r>>Init All DATA\n");
    #endif

    pdevtmp=DEV;
    for (dev=0;dev<MAX_DEV;dev++,pdevtmp++)
        pdevtmp->dev_typ=NODEV;
    dev=0;
    pdevtmp=&DEV[0];
    dev=0;
    for (slot=0;slot<MAX_SLOT;slot++){
        switch (Slot_tbl[slot]){
            case SSI:
                for (mod=0;mod<MAX_SSI_MOD;mod++,pdevtmp++,dev++){
                    /*----------
                    pdevtmp->svc=1;
                    pdevtmp->stn=500+dev;
                    ----------*/
                    pdevtmp->fea     =0xff;
                    pdevtmp->dev_typ=SSI;
                    pdevtmp->pos     = (slot << 3) | mod;
                    pdevtmp->tspcm   = 0xff;
                    pdevtmp->state   = IDLE;
                    pdevtmp->down_cnt  = 0;
                    pdevtmp->ssi_hook  = OFF;
                    pdevtmp->ssi_flash = OFF;
                    pdevtmp->ssi_digit = 0xff;
                    pdevtmp->dial_cnt  = 0;
                    pdevtmp->redi_cnt  = 0;
                    pdevtmp->dev_called=0xff;
                    pdevtmp->dev_hold  = 0xff;
                    pdevtmp->dev_dtmf  = 0xff;
                    pdevtmp->dev_co    = 0xff;
                    pdevtmp->dev_callf = 0xff;
                    pdevtmp->dev_callfbusy= 0xff;
                    pdevtmp->dev_callfno  = 0xff;
                    pdevtmp->dev_callb    = 0xff;
                    pdevtmp->dev_callbdes =0xff;
                    pdevtmp->dev_callwait =0xff;
                    pdevtmp->hot          = 0xff;
                    pdevtmp->hunt         = 0xff;
                    pdev->start.ti_hour   =0xff;
                }
              break;
            case COI:
                for (mod=0;mod<MAX_COI_MOD;mod++,pdevtmp++,dev++){
                    pdevtmp->dev_typ  =COI;
                    pdevtmp->pos      = (slot << 3) | mod;
                    pdevtmp->tspcm    = 0xff;
                    pdevtmp->state    = IDLE;
                    pdevtmp->down_cnt = 0;
                    pdevtmp->coi_hook = OFF;
                    pdevtmp->coi_ring = 0xff;
                    pdevtmp->coi_ssi  = 0;
                    pdevtmp->dial_cnt = 0;
                    pdev->start.ti_hour=0xff;
                }
              break;
            case DTMF:
                for (mod=0;mod<MAX_DTMF_MOD;mod++,pdevtmp++,dev++){
                    pdevtmp->dev_typ=DTMF;
                    pdevtmp->pos     = (slot << 3) | mod;
                    pdevtmp->tspcm   = 0xff;
                    pdevtmp->state   = IDLE;
                    pdevtmp->dtmf_free=IDLE;
                }
              break;
            case OPI:
                for (mod=0;mod<MAX_OPI_MOD;mod++,pdevtmp++,dev++){
                    pdevtmp->dev_typ  = OPI;
                    pdevtmp->pos      = (slot << 3) |mod;
                    pdevtmp->tspcm    = 0xff;
                    pdevtmp->state    = IDLE;
                    pdevtmp->down_cnt = 0;
                    pdevtmp->ssi_hook = OFF;
                    pdevtmp->ssi_flash= OFF;
```

```
                    pdevtmp->ssi_digit= 0xff;
                    pdevtmp->dial_cnt = 0;
                    pdevtmp->redi_cnt = 0;
                    pdevtmp->dev_hold = 0xff;
                    pdevtmp->dev_dtmf = 0xff;
                    pdevtmp->dev_co   = 0xff;
                    pdevtmp->dev_callf= 0xff;
                    pdevtmp->dev_callfbusy= 0xff;
                    pdevtmp->dev_callfno  = 0xff;
                    pdevtmp->dev_callb    = 0xff;
                    pdevtmp->dev_callbdes = 0xff;
                    pdevtmp->dev_callwait = 0xff;
                    pdevtmp->hot          = 0xff;
                    pdevtmp->hunt         = 0xff;
                    pdev->start.ti_hour   = 0xff;
                }
              break;
            case DCI:
                for (mod=0;mod<MAX_DCI_MOD;mod++,pdevtmp++,dev++){
                    pdevtmp->dev_typ    = DCI;
                    pdevtmp->pos        = (slot << 3) | mod;
                    pdevtmp->tspcm      = 0xff;
                    pdevtmp->state      = IDLE;
                    pdevtmp->dtmf_free =YES;
                    pdev->start.ti_hour=0xff;
                    /*  pdevtmp->stn=400+dev; */
                }
              break;
            case ISDN:
              break;
            case TSI:
              break;
            case CONF:
                for (mod=0;mod<MAX_CONF_MOD;mod++,pdevtmp++,dev++){
                    pdevtmp->dev_typ= CONF;
                    pdevtmp->a      = 0xff;
                    pdevtmp->b      = 0xff;
                    pdevtmp->c      = 0xff;
                    pdevtmp->pos    = (slot << 3) | mod;
                    pdevtmp->tspcm  = 0xff;
                    pdevtmp->state  = IDLE;
                    pdevtmp->dtmf_free=YES;
                }
              break;
            case NODEV:
                pdevtmp->dev_typ=NODEV;
              break;
        }
    }
    #ifdef DEBUG
        printf(" >MAX_DEVICE = %d\n",dev);
    wait_key();
    #endif
    return(dev);
}

unsigned char init_data_old()
/* Program all TSAC in each card */
{
    unsigned char mod=0;
    unsigned int  slot,dev;

    #ifdef DEBUG
        printf("\r>>Init All DATA\n");
    #endif

    pdevtmp=DEV;
    for (dev=0;dev<MAX_DEV;dev++,pdevtmp++)
        pdevtmp->dev_typ=NODEV;
    dev=0;
    pdevtmp=&DEV[0];
    dev=0;
    for (slot=0;slot<MAX_SLOT;slot++){
        switch (Slot_tbl[slot]){
            case SSI:
                for (mod=0;mod<MAX_SSI_MOD;mod++,pdevtmp++,dev++){
                    /*----------*/
                    pdevtmp->svc=1;
                    pdevtmp->stn=500+dev;
                    pdevtmp->fea=0xff;
```

```
        /*----------*/
        pdevtmp->dev_typ=SSI;
        pdevtmp->pos    = (slot << 3) | mod;
        pdevtmp->tspcm  = 0xff;
        pdevtmp->state  = IDLE;
        pdevtmp->down_cnt = 0;
        pdevtmp->ssi_hook = OFF;
        pdevtmp->ssi_flash= OFF;
        pdevtmp->ssi_digit= 0xff;
        pdevtmp->dial_cnt = 0;
        pdevtmp->redi_cnt = 0;
        pdevtmp->dev_called=0xff;
        pdevtmp->dev_hold = 0xff;
        pdevtmp->dev_dtmf = 0xff;
        pdevtmp->dev_co   = 0xff;
        pdevtmp->dev_callf= 0xff;
        pdevtmp->dev_callfbusy= 0xff;
        pdevtmp->dev_callfno= 0xff;
        pdevtmp->dev_callb= 0xff;
        pdevtmp->dev_callbdes=0xff;
        pdevtmp->dev_callwait=0xff;
        pdevtmp->hot      = 0xff;
        pdevtmp->hunt     = 0xff;
    }
  break;
case COI:
    for (mod=0;mod<MAX_COI_MOD;mod++,pdevtmp++,dev++){
        pdevtmp->dev_typ=COI;
        pdevtmp->pos    = (slot << 3) | mod;
        pdevtmp->tspcm  = 0xff;
        pdevtmp->state  = IDLE;
        pdevtmp->down_cnt = 0;
        pdevtmp->coi_hook = OFF;
        pdevtmp->coi_ring = 0xff;
        pdevtmp->coi_ssi  = 0;
        pdevtmp->dial_cnt =0;
        /* pdevtmp->redi_cnt = COI_TYPE; */
    }
  break;
case DTMF:
    for (mod=0;mod<MAX_DTMF_MOD;mod++,pdevtmp++,dev++){
        pdevtmp->dev_typ=DTMF;
        pdevtmp->pos    = (slot << 3) | mod;
        pdevtmp->tspcm  = 0xff;
        pdevtmp->state  = IDLE;
        pdevtmp->dtmf_free=IDLE;
    }
  break;
case OPI:
    for (mod=0;mod<MAX_OPI_MOD;mod++,pdevtmp++,dev++){
        pdevtmp->dev_typ= OPI;
        pdevtmp->pos    = (slot << 3) |mod;
        pdevtmp->tspcm  = 0xff;
        pdevtmp->state  = IDLE;
        pdevtmp->down_cnt = 0;
        pdevtmp->ssi_hook = OFF;
        pdevtmp->ssi_flash= OFF;
        pdevtmp->ssi_digit= 0xff;
        pdevtmp->dial_cnt = 0;
        pdevtmp->redi_cnt = 0;
        pdevtmp->dev_hold = 0xff;
        pdevtmp->dev_dtmf = 0xff;
        pdevtmp->dev_co   = 0xff;
        pdevtmp->dev_callf= 0xff;
        pdevtmp->dev_callfbusy= 0xff;
        pdevtmp->dev_callfno= 0xff;
        pdevtmp->dev_callb= 0xff;
        pdevtmp->dev_callbdes=0xff;
        pdevtmp->dev_callwait=0xff;
        pdevtmp->hot      = 0xff;
        pdevtmp->hunt     = 0xff;

    }
  break;
case DCI:
    for (mod=0;mod<MAX_DCI_MOD;mod++,pdevtmp++,dev++){
        pdevtmp->dev_typ= DCI;
        pdevtmp->pos    = (slot << 3) | mod;
        pdevtmp->tspcm  = 0xff;
        pdevtmp->state  = IDLE;
```

```
                    pdevtmp->dtmf_free=YES;
                    pdevtmp->stn=400+dev;
                }
              break;
            case ISDN:
              break;
            case TSI:
              break;
            case CONF:
                for (mod=0;mod<MAX_CONF_MOD;mod++,pdevtmp++,dev++){
                    pdevtmp->dev_typ= CONF;
                    pdevtmp->pos    = (slot << 3) | mod;
                    pdevtmp->tspcm  = 0xff;
                    pdevtmp->state  = IDLE;
                    pdevtmp->dtmf_free=YES;
                    pdevtmp->stn=400+dev;
                }
              break;
            case NODEV:
                pdevtmp->dev_typ=NODEV;
              break;
        }
    }
    #ifdef DEBUG
        printf(" >MAX_DEVICE = %d\n",dev);
/*      waitkey();  */
    #endif
    return(dev);
}

void init_tspcm_data()
/* Select TS/PCM  for each DEV */
{
#define max 6

    int data[max]={ SSI,COI,DTMF,DCI,OPI,CONF};
    int dev;
    unsigned char i,ts=0,pcm=0,opi=0,conf=0;
    struct data_tbl *pdata;

    for (i=0;i<max;i++){
        pdata=&DEV[0];
        for (dev=0;dev<MAX_DEV;dev++,pdata++){
            if (pdata->dev_typ==data[i]){
                #ifdef DEBUG
                    printf("DEV(%03d) SLOT(%2d) MOD(%1d) TYPE(%2d) TS=%2d,PCM=%2d\n",
                        dev,(pdata->pos)>>3,(pdata->pos)&7, pdata->dev_typ,ts,pcm);
                #endif
                switch(pdata->dev_typ){
                    case OPI:
                        opi++;
                        if (opi>MAX_OPI_MOD) opi=1;
                        if (opi==1){
                            if (ts+MAX_OPI_MOD>31)
                                pcm++,ts=0;
                        }
                      break;
                    case CONF:
                        conf++;
                        if (conf>MAX_CONF_MOD) conf=1;
                        if (conf==1){
                            if (ts+MAX_CONF_MOD>31)
                                pcm++,ts=0;
                        }
                      break;
                } /* end switch */

                pdata->tspcm=(ts<<3) | pcm;

                ts++;
                if (ts>31)
                    ts=0,pcm++;
            }
        }
    }
#undef max
}

void init_pcmtsac()
/* Program all TSAC in each card */
```

```
{
    unsigned char dev,inc;

    #ifdef DEBUG
        printf("\r>>Init All TSAC\n");
    #endif

    init_tspcm_data();
    for (dev=0;dev<MAX_DEV;dev++){
        if (DEV[dev].dev_typ != NODEV){
            STSAC(dev);
        /*  SPCM(dev);  */
        }
    }

    for (dev=0;dev<MAX_DEV;dev+=inc){
        switch (DEV[dev].dev_typ){
            case SSI:
                SPCM(dev);
                inc=MAX_SSI_MOD;
              break;
            case COI:
                SPCM(dev);
                inc=MAX_COI_MOD;
              break;
            case DTMF:
                SPCM(dev);
                inc=MAX_DTMF_MOD;
              break;
            case OPI:
                SPCM(dev);
                inc=MAX_OPI_MOD;
              break;
            case DCI:
                SPCM(dev);
                inc=MAX_DCI_MOD;
              break;
            case ISDN:
                inc=1;
              break;
            default :
                inc=1;
              break;
        }
    }
}

void init_TSI(){

    unsigned char ts,*pcomm;
    int     dev;

    #ifdef DEBUG
        printf(">>Initialize TSI \n");
    #endif
    /* Search TSI card */
    Tsi_add=0xff;
    for (dev=0;dev<MAX_SLOT;dev++)
        if (Slot_tbl[dev]==TSI) Tsi_add=dev;
    /* Reset and clear Timeslot */
    if  (Tsi_add!=0xff){
        reset_card(Tsi_add);
        #ifdef DEBUG
            printf("\n");
        #endif
        for (dev=0;dev<=255;dev++)
            send_data(Tsi_add,_tsi_ctone,tone_null,dev);
    }
}

unsigned char init_DTMF()
{
    unsigned char dev,dtmf=0;

    for (dev=0;dev<MAX_DTMF;dev++)
        Dtmf_add[dev]=0xff;
    pdevtmp=DEV;
    for (dev=0;dev<MAX_DEV;dev++,pdevtmp++)
        if (pdevtmp->dev_typ == DTMF) Dtmf_add[dtmf++]=dev;
```

```
    #ifdef DEBUG
        printf(">>Search DTMF \n");
        printf(" >DTMF found (%2d) : ",dtmf);
        for (dev=0;dev<dtmf;dev++)
            printf("%2d ",Dtmf_add[dev]);
        printf("\n");
    #endif
    return(dtmf);
}

unsigned char init_CONF()
{
    unsigned char dev,conf=0;

    for (dev=0;dev<MAX_CONF;dev++)
        Conf_add[dev]=0xff;
    pdevtmp=DEV;
    for (dev=0;dev<MAX_DEV;dev++,pdevtmp++)
        if (pdevtmp->dev_typ == CONF) Conf_add[conf++]=dev;
    #ifdef DEBUG
        printf(">>Search CONF \n");
        printf(" >CONF found (%2d) : ",conf);
        for (dev=0;dev<dtmf;dev++)
            printf("%2d ",Conf_add[dev]);
        printf("\n");
    #endif
    return(conf);
}


unsigned char init_COI()
{
    unsigned char dev,co=0;

    for (dev=0;dev<MAX_CO;dev++)
        Co_add[dev]=0xff;
    pdevtmp=DEV;
    for (dev=0;dev<MAX_DEV;dev++,pdevtmp++)
        if (pdevtmp->dev_typ == COI) Co_add[co++]=dev;

    #ifdef DEBUG
        printf(">>Search CO \n");
        printf(" >CO   found (%2d) : ",co);
        for (dev=0;dev<co;dev++)
            printf("%2d ",Co_add[dev]);
        printf("\n");
    #endif
    return(co);
}


unsigned char init_OPI()
{
    unsigned char dev,opi=0;

    for (dev=0;dev<MAX_OPI;dev++)
        Opi_add[dev]=0xff;
    pdevtmp=DEV;
    for (dev=0;dev<MAX_DEV;dev++,pdevtmp++)
        if (pdevtmp->dev_typ == OPI) Opi_add[opi++]=dev;

        #ifdef DEBUG
            printf(">>Search OPI \n");
            printf(" >OPI  found (%2d) : ",opi);
            for (dev=0;dev<opi;dev++)
                printf("%2d ",Opi_add[dev]);
            printf("\n");
        #endif
    return(opi);
}

void init_var()
{
    unsigned char i;

    h_out=0xff;h_in=0xff;
    Qcnt=0;
    coQ_cnt=0;
    pcoQ=coQ;
    for (i=0;i<MAX_COQ;i++) coQ[i]=0xff;
```

```
    Night_flag=OFF;

/*  (USE IN TEST PROCESS) */
/*
    p=Code_set , *p++=70, *p++=71 ,*p++=72 ,*p++=73 ,*p++=74;
                 *p++=75, *p++=76 ,*p++=77 ,*p++=78 ,*p++=79;
                 *p++=4 , *p++=6 ;
    p=Code_cnl , *p++=80, *p++=81 ,*p++=82 ,*p++=83 ,*p++=84;
                 *p++=85, *p++=86 ,*p++=87 ,*p++=88 ,*p++=89;
*/

/*  p=Slot_tbl;

    *p++=SSI; *p++=TSI; *p++=COI ;
    *p++=COI; *p++=OCI; *p++=COI ;
    *p++=SSI; *p++=OCI; *p++=SSI ;
    *p++=SSI; *p++=SSI; *p++=DTMF;
    *p++=OCI; *p++=OCI; *p++=OCI ;
    *p++=SSI ;*p++=OPI; *p++=DTMF;
    *p++=OCI;
*/
/*  *p++=NODEV; *p++=NODEV; *p++=OPI; *p++=NODEV;
    *p++=SSI  ; *p++=NODEV; *p++=TSI; *p++=COI;
    *p++=DTMF ;
*/
/*
    p=Night;
    *p++=7   ,*p++=8   ,*p++=0xff   ,
    *p++=0xff,*p++=0xff,*p++=5;
*/
}
```

```
void init_night()
{
    unsigned char i,j;

    for (i=0;i<=MAX_NIGHT;i++)
        pdev=DEV;
        night[i]=0xff;
        if (night[i]!=0xff)
            for (j=0;j<=MAX_DEV;j++,pdev++)
                if (pdev->srn==night[i])
                    night[i]=j;
        }
    }
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU004.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "mpu004.h"
#include "mpustate.h"

void SPCM(dev)
/* Set PCM */
unsigned char dev;
{
    pdevtmp=&DEV[dev];
/*  slot = (pdevtmp->pos) >> 3; */
    send_data((pdevtmp->pos) >> 3, set_pcm,( (pdevtmp->tspcm) & 0x07),0);
}

void STSAC(dev)
/* Set TSAC */
unsigned char dev;
{
    pdevtmp=&DEV[dev];
/*
    slot = (pdevtmp->pos) >> 3;
    mod  = (pdevtmp->pos) & 0x07;
*/
    send_data( pdevtmp->pos >> 3, set_tsac,(pdevtmp->pos) & 0x07,( (pdevtmp->tspcm) >> 3));
}

void ENTONE(dev,tone)
/* Enable Tone */
unsigned char dev,tone;
{
    #ifdef DEBUG
        printf("ENTONE : %2d ->%2d\n",dev,tone);
    #endif
    pdevtmp=&DEV[dev];
    send_data(Tsi_add,_tsi_ctone,tone,pdevtmp->tspcm);
/*  if (on_off)
     else
        send_data(Tsi_add,_tsi_ctone,tone_null,pdevtmp->tspcm);
*/
}

void ENMUS(dev)
/* Enable Music */
unsigned char dev;
{
    pdevtmp=DEV+dev;
    send_data(Tsi_add,_tsi_ctone,tone_music,pdevtmp->tspcm);
}

void SEND_DTMF(dev,num)
/* Send dtmf tone to time slot */
unsigned char dev,num;
{
    if (num==0) num=10;
    pdevtmp=DEV+dev;
    send_data(Tsi_add,_tsi_ctone,tone_dtmf+num,pdevtmp->tspcm);
}

void CONNECT_TS(source,des)
/* Connect 2 Timslot */
unsigned char source,des;
{
    send_data(Tsi_add,_tsi_exts,source,des);
}

void CONNECT_TSDEV(source,des)
/* Connect 2 DEV only send */
unsigned char source,des;
```

```
{
    send_data(Tsi_add,_tsi_exts,DEV[source].tspcm,DEV[des].tspcm);
}

void CONNECT_DEV(source,des)
/* Connect DEV both send & Receive */
unsigned char source,des;
{
    unsigned char *pcom;

    pcom=command;
    *pcom++=6;
    *pcom++=_tsi_exts , *pcom++=DEV[source].tspcm , *pcom++ =DEV[des].tspcm;
    *pcom++=_tsi_exts , *pcom++=DEV[des].tspcm ,    *pcom++ =DEV[source].tspcm;
    send_command(Tsi_add,command);

    /* send_data(Tsi_add,_tsi_exts,DEV[source].tspcm,DEV[des].tspcm); */
}

unsigned char SEARCH_OPI()
/* Search Idle OPI */
{
    unsigned char i;

    for (i=0;i<Num_opi;i++){
        if ( DEV[Opi_add[i]].state==IDLE)
            return(Opi_add[i]);
    }
    return(0xff);

}

unsigned char IS_OPI(dev)
/* Is dev OPI ? */
unsigned char dev;
{
    return(DEV[dev].dev_typ==OPI);
/*
    if (DEV[dev].dev_typ==OPI) return(TRUE);
        else return(FALSE);
*/
}

void SOPI_DIAL(dev)
/* Set OPI dial */
unsigned char dev;
{
    unsigned char *pcom,mod;
    int tmp,tmp2;

    pdevtmp=DEV+dev;
    tmp=ASSI_STN(pdevtmp->dev_called);
    mod=pdevtmp->pos&7;
    pcom=command;
    *pcom++=9;
    *pcom++=_opi_dial ,*pcom++=mod, *pcom++=(tmp2= (tmp-(tmp%100))) /100;
    tmp-=tmp2;
    *pcom++=_opi_dial ,*pcom++=mod, *pcom++=(tmp2= (tmp-(tmp%10 ))) /10;
    tmp-=tmp2;
    *pcom++=_opi_dial ,*pcom++=mod, *pcom++=tmp;
    send_command(pdevtmp->pos >>3,command);
}

unsigned char SEARCH_DTMF()
/* Search Idle DTMF */
{
    unsigned char i;

    printf("Search DTMF : ");

    for (i=0;i<Num_dtmf;i++){
        printf("%2d(%2d) ",Dtmf_add[i],DEV[Dtmf_add[i]].state);
        if ( DEV[Dtmf_add[i]].state==IDLE){
            printf("\n");
            return(Dtmf_add[i]);
        }
    }
    return (0xff);
}
```

```
unsigned char SEARCH_CONF( )
/* Search Idle Conference  */
{
    unsigned char i;

    printf("Search CONF : ");

    for (i=0;i<Num_dtmf;i++){
        printf("%2d(%2d) ",Conf_add[i],DEV[Dtmf_add[i]].state);
        if ( DEV[Conf_add[i]].state==IDLE){
            printf("\n");
            return(Conf_add[i]);
        }
    }
    return (0xff);
}

void RELEASE_DTMF( ) /* Release DTMF from SSI# dev */
                     /* Before use - Set value -> pdev,pdtmf */
{
    pdev-> dev_dtmf=0xff;
    pdtmf->state= IDLE;
}

unsigned char ADTMF_DIGIT(dev)
/* Get dial digit from dtmf */
unsigned char dev;
{
    return(DEV[dev].dev_digit);
}

unsigned char GET_DIGIT(dev)
/* Get Dial digit from DTMF or SSI module */
unsigned char dev;
{
    unsigned char digit;

    pdevtmp=DEV+dev;
    if ( (digit=pdevtmp->dev_digit)!=0xff){
        pdevtmp->dev_digit=0xff;
/*      if (digit==0) digit=10; */
        #ifdef DEBUG
            printf("GET DIGIT %d\n",digit);
        #endif
        return(digit);
    }
      else {
        digit=DEV[pdevtmp->dev_dtmf].dev_digit;
        #ifdef DEBUG
            printf("GET DIGIT %d\n",digit);
        #endif
        return (digit);
      }
}

unsigned char ADEV(slot,mod)
/* Search POS */
int slot;
unsigned char mod;
{
    unsigned char dev;
    int pos;

    pos=(slot <<3) + mod;
    pdevtmp=DEV;
    for (dev=0;dev<Num_dev;dev++,pdevtmp++)
        if (pdevtmp->pos==pos) return(dev);
    return(0xff);
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU005.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "mpu004.h"
#include "mpustate.h"

unsigned char ASSI_IDLE(dev)
/* Ask SSI Busy/Idle */
unsigned char dev;
{
    return( DEV[dev].state==IDLE);
/*
    if (DEV[dev].state==IDLE) return(TRUE);
        else return(FALSE);
*/
}

unsigned int ASSI_STN(dev)
/* Find DEV# from STN# */
unsigned char  dev;
{
    return(DEV[dev].stn);
}

unsigned char ASSI_SVC(dev)
/* Ask SVC */
unsigned char dev;
{
    return(DEV[dev].svc);
}

unsigned char ASSI_CALLB(dev)
/* Ask dev Call Back ? */
unsigned char dev;
{
    return(DEV[dev].dev_callb);
}

unsigned char ASSI_HOT(dev)
/* Ask dev set Hot line ? */
unsigned char dev;
{
    return(DEV[dev].hot);
}

unsigned char ASSI_CALLF(dev)
/* Ask dev set Call Forward ? */
unsigned char dev;
{
    return(DEV[dev].dev_callf);
}

unsigned char ASSI_POS(dev)
/* Find Pos from DEV# */
unsigned char dev;
{
    return(DEV[dev].pos);
}

unsigned char ASSI_CALLED(dev)
/* Ask Dev connected to what dev */
unsigned char dev;
{
    return(DEV[dev].dev_called);
}

unsigned char ASSI_HOLD(dev)
/* Is dev hold another dev */
unsigned char dev;
```

```
{
    return(DEV[dev].dev_hold);
}


unsigned char ASSI_DTMF(dev)
/* Ask dev connected to what dev */
unsigned char dev;
{
    return(DEV[dev].dev_dtmf);
}


unsigned char ASSI_CO(dev)
/* Ask dev connected to what dev */
unsigned char dev;
{
    return(DEV[dev].dev_co);
}

unsigned char ASSI_HUNT(dev)
/* Ask Hunting group number of dev */
unsigned char dev;
{
    return(DEV[dev].hunt);
}

unsigned char SEARCH_SSI(stn)
/* Fine dev# from STN# */
unsigned int stn;
{
    unsigned char dev;

    pdevtmp=DEV;
    for(dev=0;dev<Num_dev;dev++,pdevtmp++)
        if ( (pdevtmp->stn==stn) /* && (pdev->dev_typ==SSI) */) return(dev);
    return(0xff);
}

void SSSI_RING(dev,on_off)
/* Set SSI -> ringing ON/OFF */
unsigned char dev,on_off;
{
    pdevtmp=DEV+dev;
    if (on_off)
        send_data(pdevtmp->pos >>3, ssi_enr,pdevtmp->pos & 7,0);
      else
        send_data(pdevtmp->pos >>3, ssi_disr,pdevtmp->pos & 7,0);
}

void SSSI_CLICK(dev)
/* ENABLE  SSI Sent a short Ring */
unsigned char dev;
{
    pdevtmp=DEV+dev;
    send_data(pdevtmp->pos >>3, ssi_click,pdevtmp->pos & 7,0);
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU006.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "mpu004.h"
#include "mpustate.h"

unsigned char SEARCH_CO()
/* Search Idle CO */
{
    unsigned char i;

    for (i=0;i<Num_co;i++){
        if ( DEV[Co_add[i]].state==IDLE){
            printf(">> COFOUND %2d\n",Co_add[i]);
            return(Co_add[i]);
        }
    }
    return(0xff);
}

unsigned char ACO_TYPE(dev)
/* Ask CO_TYPE ->DTMF/PLUSE */
unsigned char dev;
{
    return(DEV[dev].redi_cnt);          /
}

unsigned char ACO_IDLE(dev)
/* Ask CO idle/busy */
unsigned char dev;
{
    return(DEV[dev].state==IDLE);
}

unsigned char ACO_POS(dev)
/* Find  CO POS */
unsigned char dev;
{
    return(DEV[dev].pos);
}

unsigned char ACO_RING(dev)
/* Ask CO Ringing or not */
unsigned char dev;
{
    return(DEV[dev].coi_ring==1);
}

void SCO_HOOK(dev,on_off)
/* SET CO ONHOOK/OFFHOOK */
unsigned char dev,on_off;
{
    pdevtmp=DEV+dev;
    if (on_off){
        send_data(pdevtmp->pos >> 3,_coi_onh,pdevtmp->pos & 7,0);
/*      pdevtmp->state=COI_ONHOOK;  */
      }
      else{
        send_data(pdevtmp->pos >> 3,_coi_offh,pdevtmp->pos & 7,0);
/*      pdevtmp->state=COI_OFFHOOK; */
      }
}

void SCO_FLASH(dev)
/* SET CO_FLASH */
unsigned char dev;
{
    pdevtmp=DEV+dev;
    send_data(pdevtmp->pos >> 3,_coi_fl,pdevtmp->pos & 7,0);
```

```
    pdevtmp->state=COI_FLASH;
}

void SCO_DIAL(dev,num)
/* SEND DIAL PULSE DIGIT */
unsigned char dev,num;
{
    printf("..........................DIAL %2d\n",num);

    pdevtmp=DEV+dev;
/*  if (num==0) num=10; */
    send_data(pdevtmp->pos >> 3,_coi_dial,pdevtmp->pos & 7,num);
    pdevtmp->state=COI_DIAL;
}

unsigned char SEARCH_NIGHT()
/* Search Night Connection */
{
    unsigned char i,tmp;

    for (i=0;i<MAX_NIGHT;i++){
        if ( (tmp=Night[i])!=0xff)
            if ( DEV[tmp].state==IDLE){
                printf(">> NIGHT FOUND %2d\n",Night[i]);
                return(Night[i]);
            }
    }
    return(0xff);
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU007.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ---------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "mpu004.h"
#include "mpustate.h"

void test()
/* Hardware test Routine */
{
    puts("RING SSI1");
    SSSI_RING(6,ON);  getch();
    SSSI_RING(6,OFF);

    puts("RING SSI2");
    SSSI_RING(7,ON);  getch();
    SSSI_RING(7,OFF);


    puts("RING OPI1");
    SSSI_RING(1,ON);  getch();
    SSSI_RING(1,OFF);

    puts("DIAL OPI");
    pdevtmp=DEV;
    pdevtmp->redi[0]=1, pdevtmp->redi[1]=2, pdevtmp->redi[2]=3;
    pdevtmp->redi[3]=4, pdevtmp->redi[4]=5, pdevtmp->redi[5]=6;

    SOPI_DIAL(0);

    puts("DIAL COI");
    puts("  OFF HOOK"); SCO_HOOK(14,OFF); getch();
    puts("  ON  HOOK"); SCO_HOOK(14,ON ); getch();
    puts("  ON  HOOK"); SCO_HOOK(14,OFF); getch();
    puts("  FLASH    "); SCO_FLASH(14)   ; getch();
    puts("  DIAL     "); SCO_DIAL (14,9 ); getch();

    puts("CONNECT DIAL TONE "), ENTONE(6,tone_dial), getch();
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU006.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  4 January 1991
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "mpu004.h"
#include "mpustate.h"

void rx_data(slot)
/* Receive data form SLOT */
unsigned char slot;
{
    call_slot(slot);
    if (active(REQ)){
        reset_cnt();
        H_out(REQ_ACK,ON);
        PPU_EN(ON);
        wait_data_rdy();
        PPU_EN(OFF);
        H_out(REQ_ACK,OFF);
        commread(command);
    }
}

void analyze_status(slot,pcom)
/* Analyze DEV Status Change */
unsigned char *pcom,slot;
{
    unsigned char i,dev,num;

    num=*pcom;
    if (num!=255){
        *pcom++ =0;
        switch (Slot_tbl[slot]){
        case SSI:
            for (i=1;i<=num;i+=3){
                dev=ADEV(slot,*(pcom+1));
                pdev=DEV+dev;
                *pQ++=dev ,Qcnt++;
                switch (*pcom++){
                    case _ssi_offh:
                        pdev->ssi_hook=OFFHOOK;
                        pcom+=2;
                        printf("SSI_OFFHOOK %02d\n",dev);
                      break;
                    case _ssi_onh:
                        pdev->ssi_hook=ONHOOK;
                        pcom+=2;
                        printf("SSI_ONHOOK  %02d\n",dev);
                      break;
                    case _ssi_fl:
                        pdev->ssi_flash=ON;
                        pcom+=2;
                        printf("SSI_FLASH   %02d\n",dev);
                      break;
                    case _ssi_dp:
                        pcom++;
                        if (*pcom!=10) pdev->ssi_digit=*pcom;
                            else pdev->ssi_digit=0;
                        pcom++;
                        printf("SSI_DIAL    %02d ->num %2d\n",dev,pdev->ssi_digit);
                      break;
                    default:
                        pQ--,Qcnt--;
                      break;
                }
            }
          break;
        case COI:
            for (i=1;i<=num;i+=3){
                dev=ADEV(slot,*(pcom+1));
```

```
            pdev=DEV+dev;
            *pQ++=dev ,Qcnt++;
            switch (*pcom++){
                case _coi_ring:
                    pdev->coi_ring=ON;
                    pdev->state=IC;
                    pcom+=2;
                    printf("COI_RING    %02d\n",dev);
                  break;
                case _coi_rcnl:
                    pdev->coi_ring=OFF;
                    pcom+=2;
                    printf("COI_RING Cancel %02d\n",dev);
                  break;
                case _coi_drdy:              /* Dial ready */
                    pdev->dev_dtmf=ON;       /* set dial_rdy flag */
                    pcom+=2;
                    printf("COI_DIAL RDY    %02d\n",dev);
                  break;
                default:
                    pQ--,Qcnt--;
                  break;
            }
        }
      break;
    case OPI:
        for (i=1;i<=num;i+=3){
            dev=ADEV(slot,*(pcom+1));
            pdev=DEV+dev;
            *pQ++=dev ,Qcnt++;
            switch (*pcom++){
                case _opi_offh:
                    pdev->ssi_hook=OFFHOOK;
                    pcom+=2;
                    printf("OPI_OFFHOOK %02d\n",dev);
                  break;
                case _opi_onh:
                    pdev->ssi_hook=ONHOOK;
                    pcom+=2;
                    printf("OPI_ONHOOK  %02d\n",dev);
                  break;
                case _opi_fl:
                    pdev->ssi_flash=ON;
                    pcom+=2;
                    printf("OPI_FLASH   %02d\n",dev);
                  break;
                case _opi_dial:
                    pcom++;
                    if (*pcom<10) pdev->ssi_digit=*pcom;
                        else pdev->ssi_digit=1+*pcom;
                    pcom++;
                    printf("OPI_DIAL    %02d ->num %2d\n",dev,pdev->ssi_digit);
                  break;
                case _opi_hold:
                    if (pdev->state !=BUSY) {
                        pdev->ostate=pdev->state;
                        pdev->state=OPI_HOLD;
                    }
                    pdev->opi_hold=ON;
                    pcom+=2;
                    printf("OPI_HOLD %02d\n",dev);
                  break;
                case _opi_cnl:
                    pcom+=2;
                  break;
                default:
                    pQ--,Qcnt--;
                  break;
            }
        }
      break;
    case DCI:
      break;
    case DTMF:
        for (i=1;i<=num;i+=3){

            dev=ADEV(slot,*(pcom+1));;
            pdev=DEV+dev;
            *pQ++=pdev->dev_dtmf;
            Qcnt++;
```

```
                switch (*pcom++){
                    case _dtmf_digit:
                        pcom++;
                        if (*pcom!=10) pdev->dev_digit=*pcom;
                            else pdev->dev_digit=0;
                        printf("DTMF_DIGIT  %02d  NUM-> %d\n",dev,pdev->dev_digit);
                        pcom++;
                      break;
                    default:
                        pQ--,Qcnt--;
                      break;
                }
              }
            break;
        case TSI:
          break;
        }
    }
}

void scan_status()
/* Scan All slot and Get Status Change */
{
    unsigned char slot;

    Qcnt=0;pQ=Qdev;
    for (slot=0;slot<MAX_SLOT;slot++){
        #ifdef DEBUG
            /*  printf("\r> SCAN  STATUS -> Slot %2d",slot); */
        #endif
        rx_data(slot);
        analyze_status(slot,command);
    }
    #ifdef DEBUG
    if (Qcnt!=0){
        printf(">>Q count = %2d\n > Q : ",Qcnt);
        for (slot=0;slot<Qcnt;slot++){
            printf("%2d ",Qdev[slot]);
        }
        printf("\n");
    }
    #endif
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU009.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  4 January 1991
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpu002.h"
#include "mpu004.h"
#include "mpustate.h"
#include "conio.h"

/* #define FULL */

void analyze_state(qdev,qcnt)
/* Analyze DEV State Change */
/* Get data from Q */
unsigned char qdev[MAX_DEV],qcnt;
{
    unsigned char i;

    pQ=qdev;
    for (i=0;i<qcnt;i++,pQ++){
        pdev =DEV+ (*pQ);
        pdtmf=DEV+pdev->dev_dtmf;
        pco  =DEV+pdev->dev_co;
            pssi =DEV+pdev->dev_called;
        phold=DEV+pdev->dev_hold;
        switch (pdev->state){
            case OFF :
                STATE0(); break;
            case IDLE:
                STATE1(); break;
            case CHK_CLASS:
                STATE2(); break;
            case DIAL:
                STATE3(); break;
            case COMMAND:
                STATE4(); break;
            case NEXT:
                STATE5(); break;
            case BUSY:
                STATE6(); break;
            case LOCK:
                STATE7(); break;

            case CALL_EX:
                STATE8(); break;
            case EX_RING:
                STATE9(); break;
            case TALK :
                STATE10(); break;
            case FLASH:
                STATE11(); break;
            case FLASH_DIAL:
                STATE111(); break;
            case CALL_TRANS:
                STATE12(); break;
            case CT_RING:
                STATE121(); break;
            case CT_TALK:
                STATE122(); break;
            case CT_FLASH:
                STATE123(); break;

            case HOLD_BUSY:
                STATE13(); break;
            case EX_HOLDED:
                STATE14(); break;

            case S_CALLF:
                STATE15(); break;
            case CALLF_DIAL:
                STATE151(); break;
```

```
case CALLF_CHECK:
    STATE152();  break;
case CALLF_OK:
    STATE153();  break;

case C_CALLF:
    STATE16();  break;

case EX_BUSY:
    STATE17();  break;

case S_CALLB:
    STATE18();  break;
case CALLB_DIAL:
    STATE181();  break;
case CALLB_OK:
    STATE182();  break;
case C_CALLB:
    STATE20();  break;

case S_HOTLINE:
    STATE21();  break;
case HOT_DIAL:
    STATE211();  break;
case HOT_OK:
    STATE212();  break;
case C_HOTLINE:
    STATE22();  break;

case REDIAL:
    STATE24();  break;

case HOLD:
    STATE25();  break;
case HOLD_DIAL:
    STATE251();  break;
case HOLD_CALLEX:
    STATE252();  break;
case HOLD_RING:
    STATE253();  break;
case HOLD_TALK:
    STATE254();  break;

case CONFERENCE:
    STATE26();  break;
case CONF_DIAL:
    STATE27();  break;
case CONF_CHECK:
    STATE28();  break;
case CONF_SET:
    STATE29();  break;

case CALL_PU:
    STATE30();  break;
case PU_CHECK:
    STATE31();  break;

case HOTLINE:
    STATE32();  break;
case HOTLINE_CNL:
    STATE33();  break;

case OUTGOING:
    STATE34();  break;
case OG_DIAL:
    STATE35();  break;
case OG_CHECK:
    STATE36();  break;
case OG_OK:
    STATE37();  break;

case IC:
    STATE38();  break;
case CHECK_NIGHT:
    STATE39();  break;
case NIGHT_RING:
    STATE40();  break;
case NIGHT_OK:
    STATE41();  break;
case OPI_H_SRC:
```

```
                STATE42();  break;
            case OPI_H_DES:
                STATE43();  break;
            case OPI_CHOT:
                STATE44();  break;
            case IC_EX_RING:
                STATE45();  break;
            case CAMP_ON:
                STATE47();  break;
            case CO_TALK:
                STATE46();  break;

            case COI_ONHOOK:
                STATE50();  break;
            case COI_OFFHOOK:
                STATE51();  break;
            case COI_FLASH:
                STATE52();  break;
            case COI_DIAL:
                STATE53();  break;
            case EX_CALLED:
                STATE60();  break;
            case HOLDED:
                STATE61();  break;
            case CT_EX_CALLED:
                STATE62();  break;
           case DIAL_SPEED:
                STATE63();  break;
           case SSPEED:
                STATE64();  break;
           case SSPEED_DIAL:
                STATE65();  break;
           case SSPEED_OK:
                STATE66();  break;
           case CSPEED:
                STATE67();  break;
           case CSPEED_DIAL:
                STATE68();  break;
           case CSPEED_OK:
                STATE69();  break;
           case CONDCI:
                STATE70();  break;
           case CONDCI_SRC:
                STATE71();  break;
           case CONDCI_DES:
                STATE72();  break;
           case DISDCI:
                STATE73();  break;
           case OG_BUSY:
                STATE74();  break;

           case OG_SCALLB:
                STATE75();  break;
           case OG_CALLB:
                STATE76();  break;
           case OG_CALLB_RING:
                STATE77();  break;

           case OPI_HOLD:
                STATE78();  break;

           case FEATURE_SET:
                STATE80();  break;

           case SCALLFD:
                STATE81();  break;
           case SCALLFD_SRC:
                STATE82();  break;
           case SCALLFD_DES:
                STATE83();  break;
           case CCALLFD:
                STATE84();  break;
           case CALLB_RING:
                STATE93();  break;
        }
    }
}

void time_out()
/* Time Count down , Analyze State change */
```

```
/* Not used */
{
    unsigned char dev;
/*
    pdev=DEV;
    pQ2=Qdev2; Qcnt2=0;

    for (dev=0;dev<Num_dev;dev++,pdev++){
      if (pdev->down_cnt!=0)
      {
        pdev->down_cnt--;
        if (pdev->down_cnt==0){
            pQ2++=dev; Qcnt2++;
            switch (pdev->state){
                case BUSY:
                  break;
                case NEXT:
                 break;
                case EX_RING:
                 break;
                case FLASH:
                 break;
                case CALL_TRANS:
                 break;
                case CT_RING:
                 break;
                case HOLD_BUSY:
                 break;
                case EX_HOLDED:
                 break;
                case S_CALLF:
                 break;
                case CALLF_OK:
                 break;
                case C_CALLF:
                 break;
                case EX_BUSY:
                 break;
                case S_CALLB:
                 break;
                case CALLB_DIAL:
                 break;
                case CALLB_OK:
                 break;
                case S_CALLB2:
                 break;
                case C_CALLB:
                 break;
                case S_HOTLINE:
                 break;
                case HOT_DIAL:
                 break;
                case HOT_OK:
                 break;
                case C_HOTLINE:
                 break;
                case OPERATOR:
                 break;
                case HOLD:
                 break;
                case HOLD_DIAL:
                 break;
                case HOLD_RING:
                 break;
                case CONFERENCE:
                 break;
                case CALL_PU:
                 break;
                case OG_DIAL:
                 break;
                case IC_EX_RING:
                 break;
                case CAMP_ON:
                 break;
            }
        }

    }
*/
}
```

```
void show_status()
/* Show Status of Dev for DEBUG Process */
{
    unsigned char xo,yo,x,y,dev;
    struct data_tbl *p;

    xo=wherex() , yo=wherey();
    gotoxy(1,1);

    cprintf("\rSSI       : ");
    p=DEV;
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",dev);
        }
    }
    cprintf("                              ");
    p=DEV;
    cprintf("\r\nSSI STATE : ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",p->state);
        }
    }
    p=DEV;
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==OPI) {
            cprintf("%3d ",p->state);
        }
    }

    p=DEV;
    cprintf("\r\nSSI CALLED: ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",p->dev_called);
        }
    }
    p=DEV;

    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==OPI) {
            cprintf("%3d ",p->dev_called);
        }
    }

#ifdef FULL
    p=DEV;
    cprintf("\r\nSSI HOLD  : ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",p->dev_hold);
        }
    }

    p=DEV;
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==OPI) {
            cprintf("%3d ",p->dev_hold);
        }
    }

    p=DEV;
    cprintf("\r\nSSI CALLF : ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",p->dev_callf);
        }
    }

    p=DEV;
    cprintf("\r\nCALLF BUSY: ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",p->dev_callfbusy);
        }
    }
```

```
    p=DEV;
    cprintf("\r\nCALLF NO  : ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",p->dev_callfno);
        }
    }

    p=DEV;
    cprintf("\r\nHOTLINE   : ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI) {
            cprintf("%3d ",p->hot);
        }
    }
#endif

    p=DEV;
    cprintf("\r\nSSI->DTMF : ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==SSI){
            cprintf("%3d ",p->dev_dtmf);
        }
    }

    p=DEV;
    cprintf("\r\nDTMF->DEV : ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==DTMF){
            cprintf("%3d ",p->dev_dtmf);
        }
    }

    p=DEV;
    cprintf("\r\nCOI  STATE: ");
    for (dev=0;dev<50;dev++,p++){
        if (p->dev_typ==COI){
            cprintf("%3d ",p->state);
        }
    }

    cprintf("\r\n-----------------------------------------\n");
    cprintf("\r                                         ");
    gotoxy(xo,yo);
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU010.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  3 January 1991
 * -----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpustate.h"
#include "mpufn.h"

void STATE2();
void STATE4();
void STATE8();
void STATE32();

void STATE0()
/* OFF */
{
}

void STATE1()
/* IDLE */
{
    if (pdev->dev_typ==OPI) {
        send_data(pdev->pos >>3 , set_tsac,0,pdev->tspcm>>3);
    }

    STSAC(*pQ);
    pdev->dial_cnt=0;
    pdev->pdial=pdev->dial;
    pdev->tmp=pdev->dev_called;
    pdev->dev_hold=0xff;
    pdev->dev_called=0xff;
    pdev->ssi_flash=OFF;

    if (pdev->ssi_hook==OFFHOOK){
        if (pdev->hot==0xff){
            /* NO HOTLINE SET */
            pdev->state=CHK_CLASS;
            #ifdef DEBUG
                printf("DEV(%2d) IDLE       -> CHK_CLASS\n",*pQ);
            #endif
            STATE2();
        }else{
            /* HOTLINE ? */
            pdev->dev_called=pdev->hot;
            /* Check Hotline_des */
            if (DEV[pdev->hot].state==IDLE){
                pdev->state=HOTLINE;
                pssi=DEV+pdev->hot;
                pssi->state=EX_CALLED;
                pssi->dev_called=*pQ;
                ENTONE(*pQ,tone_ringb);
                SSSI_RING(pdev->dev_called,ON);
            }else{
                /* Des BUSY */
                #ifdef DEBUG
                    printf("DEV(%2d) IDLE       -> BUSY (HOTLINE BUSY)\n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
                pdev->state=EX_BUSY;
            }

            #ifdef DEBUG
                printf("DEV(%2d) IDLE       -> HOTLINE\n",*pQ);
            #endif
        }
    }
}

void STATE2()
/* CHECK CLASS */
{
    unsigned char tmp;
```

```
    if (pdev->svc<6){
        pdev->pdial = pdev->dial;
        pdev->dial_cnt=0;
        if ( (tmp=SEARCH_DTMF())==0xff){
            /* NO DTMF */
            pdev->state=BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) CHK_CLASS -> BUSY     \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }else{
            STSAC(tmp);
            if (pdev->dev_typ !=OPI){
                /* connect ssi to dtmf */
                pdtmf=DEV+tmp;
                pdev ->dev_dtmf=tmp;
                pdtmf->dev_dtmf=*pQ;
                pdtmf->state=BUSY;
                pdtmf->dev_digit=0xff;
                CONNECT_TSDEV(*pQ,tmp);
            }
        pdev ->state=DIAL;
        #ifdef DEBUG
            printf("DEV(%2d) CHK_CLASS -> DIAL     \n",*pQ);
        #endif
        if (pdev->dev_typ==OPI && Night_flag==ON)
            ENTONE(*pQ,tone_warn);
         else ENTONE(*pQ,tone_dial);
        }
    }else{
        /* not allow to make call */
        pdev->state = BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CHK_CLASS -> BUSY     \n",*pQ);
        #endif

        ENTONE(*pQ,tone_busy);
    }
}

void STATE3()
/*  DIAL  */
{
    unsigned char digit;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) DIAL      -> IDLE     \n",*pQ);
        #endif
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);

        /* Check Call Back */
        STATE92();
    }

    /* FLASH */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
    }

    /* DIAL DIGIT */
    if (pdev->state==DIAL && (digit=GET_DIGIT(*pQ))!=0xff) {
        *(pdev->pdial++)=digit;     /* save digit to dial array */
        (pdev->dial_cnt)++;
        ENTONE(*pQ,tone_null);
        STATE4();
    }
}

void STATE90(tmp)
/* CHECK_CALLFBUSY */
unsigned char tmp;
{

        pssi=DEV+(tmp=pssi->dev_callfbusy);
```

```
        if (pssi->state==IDLE){
            /* CALLF EXT IDLE */
            pdev->dev_called=tmp;
            pssi->dev_called=*pQ;
            pdev->state=EX_RING;
            #ifdef DEBUG
                printf("DEV(%2d) COMMAND   -> CALL_EX  (CALLF BUSY)\n",*pQ);
            #endif
            pssi->state=EX_CALLED;
            ENTONE(*pQ,tone_ringb);
            SSSI_RING(pdev->dev_called,ON);
            /* STATE8();  Check callF all */
        }else{
            /* CALLF EXT BUSY */
            pdev->state=EX_BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) COMMAND   -> EX_BUSY(CALLF BUSY) \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }
}

void check_des()
{
    unsigned char *p,tmp,i,number;
    int STN;

    pdtmf->state=IDLE;
    ENTONE(pdev->dev_dtmf,tone_null);
    pdev->dev_dtmf=0xff;
    p=pdev->dial; STN=0;
    for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
        STN=STN*10 + *p;

    if ( pdev->dev_typ==OPI && (STN==777 || STN==838)){
        if (STN==777)
                Night_flag=ON;
            else
                Night_flag=OFF;
        pdev->state=FEATURE_SET;
        ENTONE(*pQ,tone_warn);
    }else {
        tmp=SEARCH_SSI(STN);
        if (tmp==0xff || tmp==*pQ){
            /* NO EXT */
            pdev->state=BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) COMMAND   -> BUSY     \n",*pQ);
            #endif
            pdev->dev_dtmf=0xff;
            ENTONE(*pQ,tone_busy);
        }else{
            /* EXT FINDED */
            pssi=DEV+tmp;
            /* save des-ext (may be callback )*/
            pdev->dev_called=tmp;
            /* Check EX BUSY/IDLE */
            if (pssi->state!=IDLE){
                /* EX BUSY */
                if (pssi->dev_callfbusy==0xff){
                    /* NO CALLF BUSY */
                    pdev->state=EX_BUSY;
                    #ifdef DEBUG
                        printf("DEV(%2d) COMMAND   -> EX_BUSY \n",*pQ);
                    #endif
                    ENTONE(*pQ,tone_busy);
                }else{
                    /* CHECK CALLF BUSY */
                    STATE90(tmp);
                }
            }else{
                /* EX IDLE */
                pssi->dev_called=*pQ;
                pdev->state=CALL_EX;
                #ifdef DEBUG
                    printf("DEV(%2d) COMMAND   -> CALL_EX  \n",*pQ);
                #endif
                pssi->state=EX_CALLED;
                ENTONE(*pQ,tone_ringb);
                SSSI_RING(tmp,ON);
```

```
if (pdev->svc>4){
    /* not allow to make call */
    pdev->state = BUSY;
    #ifdef DEBUG
        printf("DEV(%2d) COMMAND -> BUSY  Call 5-6 no Internal call   \n",*pQ);
    #endif
    ENTONE(*pQ,tone_busy);
    break;
}
============================*/

number=pdev->dial[0]*10+pdev->dial[1];
/* Feature Set/Cnl */
p=Code_set ,tmp=0xff;
for (i=0;i<MAX_FEATURE;i++,p++)
    if (number==*p) {
        tmp=i; break;
    }
number=tmp;

if (pdev->dev_typ ==OPI){
    /* If is OPI -> use only some feature */
    if (number==7) {
        ENTONE(*pQ,tone_sdt);
        pdev->dial_cnt=0;
        pdev->pdial=pdev->dial;
        pdev->state=OPI_H_SRC;
        break;
    }else number=0xff; /* Goto Check Code cnl */
}

/* Feature Set */
if (number!=0xff){
  /*pdtmf->state  =IDLE;
    pdev->dev_dtmf=0xff; */
    pdev->dial_cnt=0;
    pdev->pdial=pdev->dial;
    /* CODE SET FOUND */
    switch (number){
        case 0: /* CALL BACK */
            /* ERROR */
            pdev->state=BUSY;
            ENTONE(*pQ,tone_busy);
          break;
        case 1: /* CALL_FORWARD DIAL SRC */
            if (pdev->dev_callf!=0xff) {
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else {
                pdev->state=SCALLFD_SRC;
                ENTONE(*pQ,tone_sdt);
            }
          break;
        case 2: /* CALL_FORWARD ALL */
            if (pdev->dev_callf!=0xff) {
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else {
                pdev->state=S_CALLF;
                pdev->tmp=1;
                ENTONE(*pQ,tone_sdt);
            }
          break;
        case 3: /* CALL_FORWARD BUSY */
            if (pdev->dev_callfbusy!=0xff) {
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else {
                pdev->state=S_CALLF;
                pdev->tmp=2;
                ENTONE(*pQ,tone_sdt);
            }
          break;

        case 4: /* CALL_FORWARD NO ANSWER */
            if (pdev->dev_callfno!=0xff) {
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else {
                pdev->state=S_CALLF;
```

```
                STATE8();
            }
        }
    }
}

void STATE4()
/*  COMMAND */
{
    unsigned char *p,tmp,i,number;
    int STN;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) COMMAND   -> IDLE      \n",*pQ);
        #endif
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);
        /* Check Call Back */
        STATE92();
    }

    /* FLASH */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
    }

    /* Check COMMAND */
    switch(pdev->dial_cnt){
        case 1: /* 1 NUMBER */
            if ( (tmp=pdev->dial[0])== Code_set[11]){
                /* SPEED DIAL */
                pdev->dial[0]=1;
            }

            switch(pdev->dial[0]){
                case 1: /* SPEED DIAL */
                pdev->state=DIAL_SPEED;
                ENTONE(*pQ,tone_sdt);
                  break;
                case 9: /* OUTGOING CALL */
                    pdev->state=OUTGOING;
                    tmp=SEARCH_CO();
                    if (pdev->svc>3 || tmp==0xff){
                        /* SVC = 4,5 or NO COI */
                        pdev->state=OG_BUSY;
                        ENTONE(*pQ,tone_busy);
                    }else{
                        pdev->dev_called=tmp;
                        pdevtmp=DEV+tmp;
                        pdevtmp->dev_called=*pQ;
                        pdevtmp->state=BUSY;
                        pdev->state=OUTGOING;
                        /* Send tone from CO to SSI */
                        CONNECT_DEV(tmp,*pQ);
                        SCO_HOOK(tmp,OFF);
                    }
                  break;
                case 0:
                    /* OPERATOR */
                    STATE23();
                  break;
                case 11: /*  '*' -> REDIAL */
                    STATE24();
                  break;

                default:
                    #ifdef DEBUG
                        printf("DEV(%2d) COMMAND   -> NEXT  \n",*pQ);
                    #endif
                    pdev->state=NEXT;
                  break;
            }
          break;
        case 2: /* 2 NUMBER */

            /*=========================
```

```
                pdev->tmp=3;
                ENTONE(*pQ,tone_sdt);
            }
          break;
        case 5:
            /* ERROR no CAll waiting */
            pdev->state=BUSY;
            ENTONE(*pQ,tone_busy);
          break;
        case 6: /* CALL PICKUP */
            pdev->state=CALL_PU;
            ENTONE(*pQ,tone_sdt);
          break;
        case 7: /* HOTLINE */
            if (pdev->hot !=0xff) {
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else {
                pdev->state=S_HOTLINE;
                ENTONE(*pQ,tone_sdt);
            }
          break;
        case 8: /* SPEED SET */
            pdev->state=SSPEED;
            ENTONE(*pQ,tone_sdt);
          break;
        case 9:/* DCI CONNECT */
            pdev->state=CONDCI;
            ENTONE(*pQ,tone_sdt);
          break;
    }
}else{
    number=pdev->dial[0]*10+pdev->dial[1];
    /* Feature Cancel */
    p=Code_cnl ,tmp=0xff;
    for (i=0;i<MAX_FEATURE;i++,p++)
        if (number==*p) {
            tmp=i;
            break;
        }
    number=tmp;

    if (pdev->dev_typ ==OPI){
        /* If is OPI -> use only some feature */
        if (number==7) {
            ENTONE(*pQ,tone_sdt);
            pdev->dial_cnt=0;
            pdev->pdial=pdev->dial;

            pdev->state=OPI_CHOT;
            break;
        }else number=0xff; /* Goto Check Code cnl */
    }
    if (number!=0xff){
        pdev->dial_cnt=0;
        pdev->pdial=pdev->dial;
        /* CODE CNL FOUND */
        switch (number){
            case 0: /* CNL CALL BACK */
                pdtmf->state  =IDLE;
                pdev->dev_dtmf=0xff;
                pdev->state=C_CALLB;
                STATE20();
              break;
            case 1: /* CNL CALL_FORWARD DIAL SRC */
                pdev->state=CCALLFD;
                ENTONE(*pQ,tone_sdt);
                pdev->dial_cnt=0;
              break;
            case 2: /* CNL CALL_FORWARD ALL */
            case 3: /* CNL CALL_FORWARD BUSY */
            case 4: /* CNL CALL_FORWARD NO ANS*/
                pdtmf->state  =IDLE;
                pdev->dev_dtmf=0xff;
                pdev->tmp=number-1;
                pdev->state=C_CALLF;
                STATE16();
              break;
            case 5:
            case 6:
```

```
                                /* No cnl callwait, no cnl callpickup */
                                pdev->state=BUSY;
                                ENTONE(*pQ,tone_busy);
                                break;
                            case 7: /* CNL HOTLINE */
                                pdtmf->state  =IDLE;
                                pdev->dev_dtmf=0xff;
                                pdev->state=C_HOTLINE;
                                STATE22();
                              break;
                            case 6: /* CNL SPEED SET */
                                pdev->state=CSPEED;
                                ENTONE(*pQ,tone_sdt);
                                pdev->dial_cnt=0;
                              break;
                            case 9: /* DCI DISCONNECT */
                                pdev->state=DISDCI;
                                ENTONE(*pQ,tone_sdt);
                                pdev->dial_cnt=0;
                                pdev->pdial=pdev->dial;
                              break;
                        } /* End Switch number */
                    } /* END CODE CNL FOUND */
                }   /* END FEATURE CHECK */
              break;
            case 3: /* 3 NUMBER = EX CALL */
            case 4:
                check_des();
              break;
        }
}

void STATES()
/*  NEXT  */
{
    unsigned char digit;

    if (pdev->ssi_hook==ONHOOK) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) NEXT       -> IDLE      \n",*pQ);
        #endif
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);
       /* Check Call Back */
        STATE92();
    }

    /* DIAL DIGIT */
    if ( (digit=GET_DIGIT(*pQ))!=0xff) {
        *(pdev->pdial++)=digit;     /* save digit to dial array */
        (pdev->dial_cnt)++;
        ENTONE(*pQ,tone_null);
        STATE4();
    /*---- ???????????????? -----*/
    }
}

void STATES()
/*  BUSY  */
{
    pdtmf->state  =IDLE;
    pdev->dev_dtmf=0xff;
    ENTONE(*pQ,tone_busy);
    if (pdev->dev_typ==OPI)
        send_data(pdev->pos >>3 , set_tsac,0,pdev->tspcm>>3);

    if (pdev->ssi_hook==ONHOOK) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) BUSY       -> IDLE      \n",*pQ);
        #endif
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);

        /* Check Call Back */
        STATE92();
    }
```

```
}

void STATE7()
/*  LOCK  */
{
    ENTONE(*pQ,tone_null);
    if (pdev->ssi_hook==ONHOOK) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) LOCK       -> IDLE     \n",*pQ);
        #endif

        ENTONE(*pQ,tone_null);
    }
}

void STATE8()
/*  CALL_EX - Check CALL Forward */
{
    unsigned char tmp;

    if ( ((tmp=pssi->dev_callf) == 0xff) || tmp==*pQ){
        /* NO CALL FORWARD or Callf is X */
        pdev->state=EX_RING;
        #ifdef DEBUG
            printf("DEV(%2d) CALL_EX    -> EX_RING  \n",*pQ);
        #endif
    }else{
        /* CALL FORWARD SET an CALLF is not X */

        pdevtmp=DEV+(tmp=pssi->dev_callf);
        if ( pdevtmp->state==IDLE ){
            /* CALL FORWARD -> Idle and no callf set */
            pssi->state=IDLE;
            SSSI_RING(pdev->dev_called,OFF);
            SSSI_CLICK(pdev->dev_called); /* send one short tone */
            pssi=DEV+tmp;
            pdev->dev_called=tmp;
            pdev->state=EX_RING;
            #ifdef DEBUG
                printf("DEV(%2d) CALL_EX    -> CALL FORWARD \n",*pQ);
            #endif
            pssi->state=EX_CALLED;
            pssi->dev_called=*pQ;
            pdev->dev_dtmf=0xff;
            pdtmf->state=IDLE;
            ENTONE(*pQ,tone_null);
            SSSI_RING(tmp,ON);
        }else {
            /* CALL FORWARD BUSY */
            /* if CALLF_EX is set CALLF -> NO CALLB */
            pssi->state=IDLE;
            pdev->dev_called=0xff;
            pdev->state=EX_BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) CALL_EX    -> BUSY  \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }
    }
}

void STATE9()
/*  EX_RING */
{
    if (pdev->ssi_hook==ONHOOK) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) EX_RING    -> IDLE  \n",*pQ);
        #endif
        pssi->state = IDLE;

        SSSI_RING(pdev->dev_called,OFF);
        ENTONE(*pQ,tone_null);

        /* Check Call Back */
        STATE92();
    }

    if (pdev->ssi_flash==ON)
```

```
        pdev->ssi_flash=OFF;
}

void STATE10()
/*  TALK  */
{
    unsigned char tmp;

    /* ON HOOK */
    if (pdev->ssi_hook==ONHOOK) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) TALK      -> IDLE  \n",*pQ);
        #endif
        if (pssi->dev_typ==COI){
            SCO_HOOK(pdev->dev_called,ON);
            pssi->state=IDLE;
            STATE76();
        }else{
            pssi->state = BUSY;
            ENTONE(pdev->dev_called,tone_busy);
        }

        /* Check Call Back */
        STATE92();
    }

    /* FLASH */
    if (pdev->ssi_flash==ON){
        /* FLASH HOOK */
        pdev->ssi_flash=OFF;
        pdev->dev_hold=pdev->dev_called;
        ENMUS(pdev->dev_hold);
        if ( (tmp=SEARCH_DTMF())==0x.ff){
            /* No DTMF */
            pdev->state=HOLD_BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) TALK      -> HOLD BUSY  \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }else{
            /* DTMF found */
            if (pdev->dev_typ!=OPI){
                STSAC(tmp);
                pdev->dev_dtmf = tmp;
                pdtmf=DEV+tmp;
                pdtmf->state=BUSY;
                pdtmf->dev_dtmf=*pQ;
                pdtmf->dev_digit=0x.ff;
                CONNECT_TSDEV(*pQ,tmp);
            }
            pdev->state=FLASH;
            #ifdef DEBUG
                printf("DEV(%2d) TALK      -> FLASH      \n",*pQ);
            #endif
            pssi->state=HOLDED;
            ENTONE(*pQ,tone_dial);
            pdev->dial_cnt=0;
            pdev->pdial=pdev->dial;
        }
    }
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU011.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * -----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpustate.h"
#include "mpufn.h"

void STATE111();
void STATE121();
void STATE12();

void STATE11()
/*  FLASH  */
{
    unsigned char digit;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        if (phold->state==HOLDED && phold->dev_called==*pQ){
            /* HAVE A EX_HOLED */
            pdev->state = EX_HOLDED;
            #ifdef DEBUG
                printf("DEV(%2d) FLASH      -> EX_HOLD     \n",*pQ);
            #endif
            SSSI_RING(*pQ,ON);
        }else{
            /* EX_HOLDED ONHOOK */
            pdev->state = IDLE;
            #ifdef DEBUG
                printf("DEV(%2d) FLASH      -> IDLE        \n",*pQ);
            #endif
            pdev->dev_hold=0xff;
            /* Check Call Back */
            STATE92();
        }
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;

        if (phold->state==HOLDED && phold->dev_called==*pQ){
            /* If Flash ->ex_hold state is holded by old dev */
            pdev->state=TALK;
            #ifdef DEBUG
                printf("DEV(%2d) FLASH      -> TALK     \n",*pQ);
            #endif
            pssi=phold;
            pssi->state=TALK;
            pdev->dev_called=pdev->dev_hold;
            pdev->dev_hold=0xff;
            CONNECT_DEV(*pQ,pdev->dev_called);
        }else {
            /* EX_HOLDED ONHOOK */
            pdev->state=BUSY;
            ENTONE(*pQ,tone_busy);
        }
    }

    /* DIAL DIGIT */

    if ( pdev->state==FLASH ){
        /*
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
           *(pdev->pdial++)=digit;      /* save digit to dial array */
            (pdev->dial_cnt)++;
```

```
            ENTONE(*pQ,tone_null);
        }
        */
        pdev->state=FLASH_DIAL;
        STATE111();

    }

}

void STATE91(tmp)
/*  CT_CHECKCFB */
unsigned char tmp;
{
    pssi=DEV+(tmp=pssi->dev_callfbusy);
    if (pssi->state==IDLE){
        /* CALLF EXT IDLE */
        pdev->dev_called=tmp;
        pssi->dev_called=*pQ;
        pdev->state=CT_RING;
        #ifdef DEBUG
            printf("DEV(%2d) COMMAND   -> CALL_EX  (CALLF BUSY)\n",*pQ);
        #endif
        pssi->state=CT_EX_CALLED;
        ENTONE(*pQ,tone_ringb);
        SSSI_RING(pdev->dev_called,ON);
    /*      STATE8();  Check callF all */
    }else{
        /* CALLF EXT BUSY */
        pdev->state=HOLD_BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) COMMAND   -> EX_BUSY(CALLF BUSY) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }
}

void STATE111()
/*  FLASH_DIAL  */
{
    unsigned char *p,tmp;
    int STN,digit;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        if (phold->state==HOLDED && phold->dev_called==*pQ){
        /* HAVE A EX_HOLED */
            pdev->state = EX_HOLDED;
            #ifdef DEBUG
                printf("DEV(%2d) FLASH      -> EX_HOLD     \n",*pQ);
            #endif

            SSSI_RING(*pQ,ON);
        }else{
            /* EX_HOLDED ONHOOK */
            pdev->state = IDLE;
            #ifdef DEBUG
                printf("DEV(%2d) FLASH_DIAL-> IDLE        \n",*pQ);
            #endif
            /* Check Call Back */
            STATE92();
        }
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        if (phold->state==HOLDED && phold->dev_called==*pQ){
            pdev->state=TALK;
            #ifdef DEBUG
                printf("DEV(%2d) FLASH      -> TALK     \n",*pQ);
            #endif
            pssi=phold;
            pssi->state=TALK;
            pdev->dev_called=pdev->dev_hold;
```

```
            pdev->dev_hold=0xff;
            CONNECT_DEV(*pQ,pdev->dev_called);
          /*CONNECT_TSDEV(pdev->dev_called,*pQ);  */
        }else {
            /* HOLDED ONHOOK */
            pdev->state=BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) FLASH_DIAL-> IDLE\n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }
    }

    /* DIAL DIGIT */

    if (pdev->state==FLASH_DIAL){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;      /* save digit to dial array */
            (pdev->dial_cnt)++;
            ENTONE(*pQ,tone_null);
        }

        /* Check COMMAND */
        switch(pdev->dial_cnt){
          case 1: /* 1 NUMBER */
            switch(pdev->dial[0]){
                case 9: /* OUTGOING CALL */
                  break;
                case 0:
                    /* CALL OPERATOR */
                    pdtmf->state  =IDLE;
                    pdev->dev_dtmf=0xff;
                    tmp=SEARCH_OPI();
                    if (pdev->dev_typ==OPI) tmp=0xff; /* if dev=OPI not connect*/
                    if (tmp==0xff){
                        /* NO IDLE OPI */
                        pdev->state=HOLD_BUSY;
                        #ifdef DEBUG
                            printf("DEV(%2d) FLASH_DIAL-> HOLD_BUSY  \n",*pQ);
                        #endif
                        ENTONE(*pQ,tone_busy);
                    }else{
                        pssi=DEV+tmp;
                        pdev->state=CT_RING;
                        #ifdef DEBUG
                            printf("DEV(%2d) FLASH_DIAL-> CT_RING  \n",*pQ);
                        #endif
                        pssi->state=CT_EX_CALLED;
                        pdev->dev_called=tmp;;
                        pssi->dev_called=*pQ;
                        SSSI_RING(tmp,ON);
                        ENTONE(*pQ,tone_ringb);
                    }
                  break;
                default:
                    #ifdef DEBUG
                        printf("DEV(%2d) FLASH_DIAL-> FLASH_DIAL  \n",*pQ);
                    #endif
                    pdev->state=FLASH_DIAL;
                  break;
            }
          break;
          case 2: /* 2 NUMBER */
            /* Feature called */
            /* HOLD */
            /* CONFERENCE */
            break;
          case 3: /* 3 NUMBER = EX CALL */
          case 4:
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;

            tmp=SEARCH_SSI(STN);
            if (tmp==0xff || tmp ==pdev->dev_hold){
                /* NO EXT */
                pdev->state=HOLD_BUSY;
                #ifdef DEBUG
```

```
                    printf("DEV(%2d) FLASH_DIAL-> HOLD_BUSY   \n",*pQ);
                #endif
                pdev->dev_dtmf=0xff;
                ENTONE(*pQ,tone_busy);
            }else{
                /* EXT FINDED */
                pssi=DEV+tmp;
                pdev->dev_called=tmp;
                /* Check EXT IDLE/BUSY */
                if (pssi->state!=IDLE){
                    /* EX BUSY */
                    if (pssi->dev_callfbusy==0xff){
                        /* NO CALLF BUSY */
                        pdev->state=HOLD_BUSY;
                        #ifdef DEBUG
                            printf("DEV(%2d) FLASH_DIAL-> HOLD_BUSY  (CALLFB->BUSY) \n",*pQ);
                        #endif
                        ENTONE(*pQ,tone_busy);
                    }else{
                        STATE31(tmp);
                    }
                }else{
                    pssi->dev_called=*pQ;
                    pdev->state=CALL_TRANS;
                    #ifdef DEBUG
                        printf("DEV(%2d) FLASH_DIAL-> CALL_TRANS \n",*pQ);
                    #endif
                    pssi->state=CT_EX_CALLED;
                    ENTONE(*pQ,tone_ringb);
                    SSSI_RING(tmp,ON);
                    STATE12();
                }
            }
          break;
        }
    }
}

void STATE12()
/*  CALL_TRANS  */
{
    /* Check CALL FORWARD */
    unsigned char tmp;

    if (pssi->dev_callf == 0xff){
        /* NO CALL FORWARD */
        pdev->state=CT_RING;
        #ifdef DEBUG
            printf("DEV(%2d) CALL TRANS-> CT_RING \n",*pQ);
        #endif
    }else{
        /* CALL FORWARD SET */
        pdevtmp=DEV+(tmp=pssi->dev_callf);
        if ( (pdevtmp->state==IDLE) && (pdevtmp->dev_callf==0xff)){
            /* CALLF EX -> Idle and no callf set */

                pssi->state=IDLE;
                SSSI_RING(pdev->dev_called,OFF);
                pssi=DEV+tmp;
                pdev->dev_called=tmp;
                pdev->state=CT_RING;
                #ifdef DEBUG
                    printf("DEV(%2d) CALL_TRANS-> CT_RING \n",*pQ);
                #endif
                pssi->state=CT_EX_CALLED;
                pssi->dev_called=*pQ;

                pdev->dev_dtmf=0xff;
                pdtmf->state=IDLE;

                ENTONE(*pQ,tone_ringb);
                SSSI_RING(tmp,ON);

        }else {
            pdev->dev_called=0xff;
            pdev->state=HOLD_BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) CALL_TRANS-> HOLD_BUSY  \n",*pQ);
            #endif
            pssi->state=IDLE;
```

```
            ENTONE(*pQ,tone_busy);
        }
    }

}

void STATE121_OLD()
/*  CT_RING  */
{
    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        SSSI_RING(pdev->dev_called,OFF);
        pssi->state=IDLE;
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdevtmp=DEV+pdev->dev_hold;
        if (pdevtmp->state==HOLDED && pdevtmp->dev_called==*pQ){
        /* HAVE A EX_HOLED */
            pdev->state = EX_HOLDED;
            #ifdef DEBUG
                printf("DEV(%2d) CT_RING   -> EX_HOLD     \n",*pQ);
            #endif

            /*  pdev->dev_called=pdev->dev_hold;
                pdev->dev_hold=0xff;
            */
            SSSI_RING(*pQ,ON);
        }else{
            /* EX_HOLDED ONHOOK */
            pdev->state = BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) CT_RING   -> IDLE        \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
            pdev->dev_hold=0xff;
            /* Check Call Back */
            STATE92();
        }
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        if (phold->state==HOLDED && phold->dev_called==*pQ){
            pdev->state=TALK;
            #ifdef DEBUG
                printf("DEV(%2d) CT_RING   -> TALK        \n",*pQ);
            #endif
            pssi=phold;
            pssi->state=TALK;
            pdev->dev_called=pdev->dev_hold;
            pdev->dev_hold=0xff;
            CONNECT_DEV(*pQ,pdev->dev_called);
         /* CONNECT_TSDEV(pdev->dev_called,*pQ); */
        }else {
            pdev->state=BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) CT_RING   -> IDLE        \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }
    }
}

void STATE121()
/*  CT_RING  */
{
    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        pdev->state=IDLE;
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        if (phold->state==HOLDED && phold->dev_called==*pQ){
        /* HAVE A EX_HOLED */
            phold->state = EX_RING;
            #ifdef DEBUG
```

```
                printf("DEV(%2d) CT_RING   -> IDLE (dev_hold->EX_RING)\n",*pQ);
            #endif
            phold->dev_called=pdev->dev_called;
            pssi ->dev_called=pdev->dev_hold;
            pssi ->state=EX_CALLED;
            ENTONE(pdev->dev_hold,tone_ringb);
        }else{
            pdev->state = IDLE;
            #ifdef DEBUG
                printf("DEV(%2d) CT_RING   -> IDLE       \n",*pQ);
            #endif
            pdev->dev_hold=0xff;
            /* Check Call Back */
            STATE92();
        }
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;

        pdtmf->state=IDLE;
        pdev->dev_dtmf =0xff;

        if (phold->state==HOLDED && phold->dev_called==*pQ){
            /* EX_HOLDED HOLDED */
            SSSI_RING(pdev->dev_called,OFF);
            pssi->state=IDLE;

            pdev->state=TALK;
            #ifdef DEBUG
                printf("DEV(%2d) CT_RING   -> TALK       \n",*pQ);
            #endif
            pssi=phold;
            pssi->state=TALK;
            pdev->dev_called=pdev->dev_hold;
            pdev->dev_hold=0xff;
            ENTONE(*pQ,tone_null);
            CONNECT_DEV(*pQ,pdev->dev_called);
        }else {
            /* EX_HOLD ONHOOK */
            /* Continue ringing */

/*          pdev->state=EX_RING;
            pssi->state=EX_CALLED;
*/
            #ifdef DEBUG
                printf("DEV(%2d) CT_RING   -> IDLE (X^ & NO_HOLDED) \n",*pQ);
            #endif

            pdev->state=BUSY;
            pssi->state=IDLE;
            ENTONE(*pQ,tone_busy);
            SSSI_RING(pdev->dev_called,OFF);

        }
    }

}

#ifdef FLASH_CONF
    void STATE122()
    /*  CT_TALK  */
    {
        unsigned char tmp;

        /* IS X ? */
        if (pdev->dev_hold != 0xff){
            /* X ONHOOK */
            if (pdev->ssi_hook==ONHOOK){
                pdev->state=IDLE;
                #ifdef DEBUG
                    printf("DEV(%2d) CT_TALK   -> IDLE       \n",*pQ);
                #endif
                ENTONE(*pQ,tone_null);
                if (phold->state==HOLDED){
                    pssi-> state=TALK;
                    phold->state=TALK;
                    CONNECT_DEV(pdev->dev_called,pdev->dev_hold);
                    pssi ->dev_called=pdev->dev_hold;
```

```
            phold->dev_called=pdev->dev_called;
        }else{
            pssi->state=BUSY;
            ENTONE(pdev->dev_called,tone_busy);
        }
        pdev->dev_called=pdev->dev_hold=0xff;

        /* Check Call Back */
        STATE92();
    }

    /* X FLASH -> CONFERENCE (CT_FLASH)*/
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        if (phold->state==HOLDED){
            tmp=SEARCH_CONF();
            if (tmp==0xff){
                /* NO CONF IDLE */
                ENTONE(*pQ,tone_camp);

                tmp=pdev->dev_called;
                pdev->dev_called=pdev->dev_hold;
                pdev->dev_hold  =tmp;
                #ifdef DEBUG
                    printf("DEV(%2d) CT_TALK   -> CT_FLASH   \n",*pQ);
                #endif
                pdev->state=CT_FLASH;
                pssi->state=HOLDED;
                ENMUS(pdev->dev_hold);
                phold->state=CT_FLASH;
                CONNECT_DEV(*pQ,pdev->dev_called);

            }else{
                #ifdef DEBUG
                    printf("DEV(%2d) CT_TALK   -> CONFERENCE   \n",*pQ);
                #endif

                pdevtmp=DEV+tmp;
                pdevtmp->a=(pdevtmp+1)->b=(pdevtmp+2)->c=*pQ;
                pdevtmp->c=(pdevtmp+1)->a=(pdevtmp+2)->b=pdev->dev_called;
                pdevtmp->b=(pdevtmp+1)->c=(pdevtmp+2)->a=pdev->dev_hold;

                /* Set conf group */
                pdevtmp->dev_called=tmp;
                (pdevtmp+1)->dev_called=tmp;
                (pdevtmp+2)->dev_called=tmp;

                pdevtmp    ->state=BUSY;
                (pdevtmp+1)->state=BUSY;
                (pdevtmp+2)->state=BUSY;

                CONNECT_DEV(*pQ,tmp);
                CONNECT_DEV(pdev->dev_called,tmp+1);
                CONNECT_DEV(pdev->dev_hold  ,tmp+2);

                pdev ->dev_called=tmp;
                pssi ->dev_called=tmp+1;
                phold->dev_called=tmp+2;

                pdev->state =CONFERENCE;
                pssi->state =CONFERENCE;
                phold->state=CONFERENCE;

                pdev->dev_hold=0xff;
                pssi->dev_hold=0xff;
                phold->dev_hold=0xff;

            }
        }else{
            /* EXHOLDED ONHOOK */
            pdev->state=HOLD_BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) CT_TALK   -> HOLD_BUSY  \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
            pdev->dev_hold=pdev->dev_called;
            pssi->state=HOLDED;
            ENMUS(pdev->dev_hold);
        }
    }
```

```
        }else { /* IS EXT */
            /* CALLED ONHOOK */
            if (pdev->ssi_hook==ONHOOK){
                if (DEV[pssi->dev_hold].state==HOLDED){
                    /* HOLDED_EX IS HOLDED */
                    pdev->state=IDLE;
                    ENTONE(*pQ,tone_null);
                    #ifdef DEBUG
                        printf("DEV(%2d) CT_TALK   -> IDLE   \n",*pQ);
                    #endif
                    pssi->state= TALK;   /*  ????? CT_TALK */
                    phold=DEV+pssi->dev_hold;
                    phold->state=TALK;  /*  ????? CT_TALK */
                    pssi->dev_called=pssi->dev_hold;
                    pssi->dev_hold=0xff;
                    CONNECT_DEV(phold->dev_called,pssi->dev_called);
                }else{
                    /* HOLD ONHOOK */
                    pssi->state=BUSY;
                    pdev->state=IDLE;
                    ENTONE(*pQ,tone_null);
                    ENTONE(pdev->dev_called,tone_busy);
                }
                /* Check Call Back */
                STATE92();
            }
        }
        /* HOLD  ONHOOK */
    }

#else
    void STATE122()
    /*  CT_TALK  -> Change ->state123 (mpu016.c)*/
    {
        unsigned char tmp;

        /* IS X ? */
        if (pdev->dev_hold != 0xff){
            /* X ONHOOK */
            if (pdev->ssi_hook==ONHOOK){
                pdev->state=IDLE;
                #ifdef DEBUG
                    printf("DEV(%2d) CT_TALK   -> IDLE       \n",*pQ);
                #endif
                ENTONE(*pQ,tone_null);
                if (phold->state==HOLDED){
                    pssi-> state=TALK;
                    phold->state=TALK;
                    CONNECT_DEV(pdev->dev_called,pdev->dev_hold);
                    pssi ->dev_called=pdev->dev_hold;
                    phold->dev_called=pdev->dev_called;
                }else{
                    pssi->state=BUSY;
                    ENTONE(pdev->dev_called,tone_busy);
                }
                pdev->dev_called=pdev->dev_hold=0xff;

                /* Check Call Back */
                STATE92();
            }

            /* X FLASH */
            if (pdev->ssi_flash==ON){
                pdev->ssi_flash=OFF;
                if (phold->state==HOLDED){
                    tmp=pdev->dev_called;
                    pdev->dev_called=pdev->dev_hold;
                    pdev->dev_hold  =tmp;
                    #ifdef DEBUG
                        printf("DEV(%2d) CT_TALK   -> CT_FLASH   \n",*pQ);
                    #endif
                    pdev->state=CT_FLASH;
                    pssi->state=HOLDED;
                    ENMUS(pdev->dev_hold);
                    phold->state=CT_FLASH;
                    CONNECT_DEV(*pQ,pdev->dev_called);
                }else{
                    /* EXHOLDED ONHOOK */
                    pdev->state=HOLD_BUSY;
                    #ifdef DEBUG
```

```
                    printf("DEV(%2d) CT_TALK   -> HOLD_BUSY  \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
                pdev->dev_hold=pdev->dev_called;
                pssi->state=HOLDED;
                ENMUS(pdev->dev_hold);
            }
        }
    }else { /* IS EXT */
        /* CALLED ONHOOK */
        if (pdev->ssi_hook==ONHOOK){
            if (DEV[pssi->dev_hold].state==HOLDED){
                /* HOLDED_EX IS HOLDED */
                pdev->state=IDLE;
                ENTONE(*pQ,tone_null);
                #ifdef DEBUG
                    printf("DEV(%2d) CT_TALK   -> IDLE   \n",*pQ);
                #endif
                pssi->state= TALK;   /*  ????? CT_TALK */
                phold=DEV+pssi->dev_hold;
                phold->state=TALK;  /*  ????? CT_TALK */
                pssi->dev_called=pssi->dev_hold;
                pssi->dev_hold=0xff;
                CONNECT_DEV(phold->dev_called,pssi->dev_called);
            }else{
                /* HOLD ONHOOK */
                pssi->state=BUSY;
                pdev->state=IDLE;
                ENTONE(*pQ,tone_null);
                ENTONE(pdev->dev_called,tone_busy);
            }
            /* Check Call Back */
            STATE92();
        }
    }
    /* HOLD  ONHOOK */
}
#endif


void STATE123()
/*  CT_FLASH  */
{
    unsigned char tmp;

    /* IS X ? */
    if (pdev->dev_hold != 0xff){
        /* X ONHOOK */
        if (pdev->ssi_hook==ONHOOK){
            pdev->state=IDLE;
            #ifdef DEBUG
                printf("DEV(%2d) CT_TALK   -> IDLE       \n",*pQ);
            #endif
            ENTONE(*pQ,tone_null);
            if (phold->state==HOLDED){
                /* EXHOLDED OFFHOOK */
                pssi-> state=TALK;
                phold->state=TALK;
                CONNECT_DEV(pdev->dev_called,pdev->dev_hold);
              /*CONNECT_TSDEV(pdev->dev_hold,pdev->dev_called); */
                pssi ->dev_called=pdev->dev_hold;
                phold->dev_called=pdev->dev_called;

            }else{
                pssi->state=BUSY;
                ENTONE(pdev->dev_called,tone_busy);
            }
            pdev->dev_called=pdev->dev_hold=0xff;
            /* Check Call Back */
            STATE92();

        }

        /* X FLASH */
        if (pdev->ssi_flash==ON){
            pdev->ssi_flash=OFF;
            if (phold->state==HOLDED){
                tmp=pdev->dev_called;
                pdev->dev_called=pdev->dev_hold;
                pdev->dev_hold  =tmp;
```

```
                #ifdef DEBUG
                    printf("DEV(%2d) CT_TALK   -> CT_FLASH   \n",*pQ);
                #endif
                pdev->state=CT_TALK;
                pssi->state=HOLDED;
                ENMUS(pdev->dev_hold);
                phold->state=CT_TALK;
                CONNECT_DEV(*pQ,pdev->dev_called);
                /*CONNECT_TSDEV(pdev->dev_called,*pQ); */
            }else{
                /* EXHOLDED ONHOOK */
                pdev->state=HOLD_BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) CT_FLASH  -> HOLD_BUSY  \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
                pdev->dev_hold=pdev->dev_called;
                pssi->state=HOLDED;
                ENMUS(pdev->dev_hold);
            }
        }
    }else { /* IS EXT */
        /* CALLED ONHOOK */
        if (pdev->ssi_hook==ONHOOK){
            if (phold->state==HOLDED){
                pdev->state=IDLE;
                ENTONE(*pQ,tone_null);
                #ifdef DEBUG
                    printf("DEV(%2d) CT_TALK   -> IDLE   \n",*pQ);
                #endif
                pssi->state=TALK;   /*  ????? CT_TALK */
                phold=DEV+pssi->dev_hold;
                phold->state=TALK;  /*  ????? CT_TALK */
                pssi->dev_called=pssi->dev_hold;
                pssi->dev_hold=0xff;
                CONNECT_DEV(phold->dev_called,pssi->dev_called);
              /*CONNECT_TSDEV(pssi->dev_called,phold->dev_called); */
            }else{
                /* HOLD ONHOOK */
                pssi->state=BUSY;
                pdev->state=IDLE;
                ENTONE(*pQ,tone_null);
                ENTONE(pdev->dev_called,tone_busy);

            }
            /* Check Call Back */
            STATE92();

        }
        /* CALLED FLASH */
    }
    /* HOLD  ONHOOK */
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU012.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpustate.h"
#include "mpufn.h"

void STATE151();
void STATE152();
void STATE92();

void STATE13()
/*  HOLD BUSY  */
{
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        if (phold->state==HOLDED && phold->dev_called==*pQ){
            /* HAVE A EX_HOLED */
            pdev->state = EX_HOLDED;
            #ifdef DEBUG
                printf("DEV(%2d) HOLD_BUSY -> EX_HOLDED\n",*pQ);
            #endif
            SSSI_RING(*pQ,ON);
        }else{
            /* EX_HOLDED ONHOOK */
            pdev->state = IDLE;
            #ifdef DEBUG
                printf("DEV(%2d) HOLD_BUSY -> IDLE       \n",*pQ);
            #endif
            pdev->dev_hold=0xff;

            /* Check Call Back */
            STATE92();
        }
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        if (phold->state==HOLDED && phold->dev_called==*pQ){
            pdev->state=TALK;
            #ifdef DEBUG
                printf("DEV(%2d) HOLD_BUSY -> TALK       \n",*pQ);
            #endif
            pssi=phold;
            pssi->state=TALK;
            pdev->dev_called=pdev->dev_hold;
            pdev->dev_hold=0xff;
            ENTONE(*pQ,tone_null);
            CONNECT_DEV(*pQ,pdev->dev_called);
/*          CONNECT_TSDEV(pdev->dev_called,*pQ);  */
        }else {
            pdev->state=BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) HOLD_BUSY -> IDLE       \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }
    }
}

void STATE14()
/*  EX_HOLDED  */
{
    if (pdev->ssi_hook==OFFHOOK){
        /* if EX_HOLDED */
        pdev->dev_called=pdev->dev_hold;
        pdev->dev_hold=0xff;
        pssi=DEV+pdev->dev_called;
```

```
        SSSI_RING(*pQ,OFF);
        CONNECT_DEV(*pQ,pdev->dev_called);
        /*CONNECT_TSDEV(pdev->dev_called,*pQ); */
        pdev->state=TALK;
        #ifdef DEBUG
            printf("DEV(%2d) FLASH_DIAL-> TALK \n",*pQ);
        #endif
        pssi->state=TALK;
    }
}

void STATE15()
/*  S_CALLF  */
{
    unsigned char digit;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) S_CALLF   -> IDLE       \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();

    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) S_CALLF   -> BUSY (X)  \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==S_CALLF){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
            pdev->state=CALLF_DIAL;
            ENTONE(*pQ,tone_null);
            /* STATE151();  */
        }
    }
}

void STATE151()
/*  CALLF_DIAL */
{
    unsigned int digit;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) CALLF_DIAL-> IDLE       \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
```

```
            printf("DEV(%2d) CALLF DIAL-> BUSY (X') \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==CALLF_DIAL){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;      /* save digit to dial array */
            (pdev->dial_cnt)++;
        }
    }
    if (pdev->dial_cnt>=MAX_STN){
          /* 3 NUMBER = EX CALL */
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=CALLF_CHECK;
        #ifdef DEBUG
            printf("DEV(%2d) CALLF DIAL -> CALLF_CHECK (%d)\n",*pQ,pdev->tmp);
        #endif
        STATE152();
    }else pdev->state=CALLF_DIAL;
}

void STATE152()
/*  CALLF_CHECK */
{
    unsigned char *p,tmp;
    unsigned int STN;

    p=pdev->dial; STN=0;
    for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
        STN=STN*10 + *p;
    tmp=SEARCH_SSI(STN);
    if (tmp!=0xff && tmp!=*pQ){
        /* STN FOUND */
        pdev->state=FEATURE_SET;
        switch (pdev->tmp){
            case 1:
                pdev->dev_callf=tmp;
        #ifdef DEBUG
            printf("DEV(%2d) CALLF CHK  -> FEATURE_OK (CALLF ALL)\n",*pQ);
        #endif
              break;
            case 2:
                pdev->dev_callfbusy=tmp;
        #ifdef DEBUG
            printf("DEV(%2d) CALLF CHK  -> FEATURE_OK (CALLF BUSY)\n",*pQ);
        #endif
              break;
            case 3:
                pdev->dev_callfno=tmp;
        #ifdef DEBUG
            printf("DEV(%2d) CALLF CHK  -> FEATURE OK (CALLF NOA\n",*pQ);
        #endif
              break;
        }
        ENTONE(*pQ,tone_sst);
    }else{
        /* NO STN FOUND */
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CALLF CHK  -> BUSY (NO STN)\n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }
    /*---------------------------
    /* Check for CALLF LOOP */
    /* NOT USED */

    unsigned char *p,tmp,fdev,fcnt;
    unsigned int STN;

    p=pdev->dial; STN=0;
    for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
        STN=STN*10 + *p;

    tmp=SEARCH_SSI(STN);
    if (tmp==0xff){
```

```
        /* NO STN */
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) S_CALLF   -> BUSY        \n",*pQ);
        #endif
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_busy);
    }else{
        /* EXT FINDED / SET CALLF */
        pssi=DEV+tmp;
        if (pssi->dev_callf==0xff){
            /* Des no CallF */
            pdev->state=FEATURE_SET;
            #ifdef DEBUG
                printf("DEV(%2d) CALLF_CHK -> FEATURE OK \n",*pQ);
            #endif
            pdev->dev_callf=tmp;
            ENTONE(*pQ,tone_sst);
        }else{
            /* Des have CallF */
            fdev=tmp,fcnt=0;
            do{
                pssi=DEV+fdev;  /* Callf dev */
                if( (fdev=pssi->dev_callf) ==0xff){
                    /* NO CALLF */
                    pdev->state=FEATURE_SET;
                    #ifdef DEBUG
                        printf("DEV(%2d) CALLF_CHK -> FEATURE OK \n",*pQ);
                    #endif
                    pdev->dev_callf=tmp;
                    ENTONE(*pQ,tone_sst);
                }else{
                    fcnt++;
                }
            }while (fcnt<4 || pdev->state==FEATURE_SET);
            if (fcnt>=4){
                /* CallF more than 3 call */
                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) CALLF_CHK -> BUSY (CALLF>3) \n",*pQ);
                #endif
                pdev->dev_callf=0xff;
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    ----------------------------*/
}

void STATE153()
/*  CALLF_OK  */
{
}

void STATE15S()
/*  C_CALLF  */
{
    switch(pdev->tmp){
        case 1:
            if (pdev->dev_callf==0xff){
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else{
                pdev->dev_callf=0xff;
                pdev->state=FEATURE_SET;
                ENTONE(*pQ,tone_sst);
            }
          break;
        case 2:
            if (pdev->dev_callfbusy==0xff){
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else{
                pdev->dev_callfbusy=0xff;
                pdev->state=FEATURE_SET;
                ENTONE(*pQ,tone_sst);
            }
          break;
        case 3:
            if (pdev->dev_callfno==0xff){
```

```
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }else{
                pdev->dev_callfno=0xff;
                pdev->state=FEATURE_SET;
                ENTONE(*pQ,tone_set);
            }
          break;
    }
}

void STATE17()
/*  EX_BUSY  */
{

    unsigned char i,tmp;
    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        /* SAVE OLD NUMBER */
        if (pdev->dial[0]!=11){
            pdev->redi_cnt=pdev->dial_cnt;
            pdev->pdial=pdev->dial;
            pdev->predi=pdev->redi;
            for (i=0;i<pdev->dial_cnt;i++)
                *pdev->predi++=*pdev->pdial++;
        }

        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) EX_BUSY   -> IDLE      \n",*pQ);
        #endif
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdev->dial_cnt=0;
        pdev->pdial=pdev->dial;
        tmp=SEARCH_DTMF();
        if (tmp==0xff){
            pdev->state=BUSY;
            ENTONE(*pQ,tone_busy);
            #ifdef DEBUG
                printf("DEV(%2d) EX_BUSY   -> BUSY (X No DTMF)\n",*pQ);
            #endif
        }else{
            pdev->state=S_CALLB;
            pdtmf=DEV+tmp;
            pdev ->dev_dtmf=tmp;
            pdtmf->dev_dtmf=*pQ;
            pdtmf->state=BUSY;
            pdtmf->dev_digit=0xff;
            CONNECT_TSDEV(*pQ,tmp);
            ENTONE(*pQ,tone_sdt);
            #ifdef DEBUG
                printf("DEV(%2d) EX_BUSY   -> S_CALLB   \n",*pQ);
            #endif
        }
    }
}

void STATE18()
/*  S_CALLB , SCALL_W */
{
    unsigned char tmp,*p,i,digit;
    int NUM;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) S_CALLB   -> IDLE        \n",*pQ);
```

```
        #endif

        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) S_CALLB   -> BUSY (X) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    if (pdev->state==S_CALLB){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;      /* save digit to dial array */
             (pdev->dial_cnt)++;
             ENTONE(*pQ,tone_null);
        }
        if (pdev->dial_cnt==2){
            NUM=pdev->dial[0]*10 + pdev->dial[1];
            p=code_set ,tmp=0xff;
            for (i=0;i<MAX_FEATURE;i++,p++)
                if (NUM==*p) {
                    tmp=i; break;
                }
            NUM=tmp;
            switch (NUM){
                case 0: /* CALLB */
                    if ( DEV[pdev->dev_called].dev_callb==0xff){
                        DEV[pdev->dev_called].dev_callb=*pQ;
    /*                  pdev->dev_callb=pdev->dev_called; */
                        pdev->dev_callbdes=pdev->dev_called;
                        pdev->state=FEATURE_SET;
                        ENTONE(*pQ,tone_sst);
                        #ifdef DEBUG
                            printf("DEV(%2d) S_CALLB   -> FEATURE SET (CALLB SET) \n",*pQ);
                        #endif
                    }else{
                        pdev->state=BUSY;
                        ENTONE(*pQ,tone_busy);
                    }
                  break;
                case 5:/* CALL WAITING */
                    if (DEV[pdev->dev_called].dev_callwait==0xff){
                        DEV[pdev->dev_called].dev_callwait=*pQ;
    /*                  pdev->dev_callwait=pdev->dev_called; */
                        pdev->state=HOLDED;
                        ENMUS(*pQ);
                        #ifdef DEBUG
                            printf("DEV(%2d) S_CALLB   -> FEATURE SET (CALL WAIT) \n",*pQ);
                        #endif
                    }else{
                        pdev->state=BUSY;
                        ENTONE(*pQ,tone_busy);
                    }
                  break;
                default:
                    pdev->state=BUSY;
                    ENTONE(*pQ,tone_busy);
                    #ifdef DEBUG
                        printf("DEV(%2d) S_CALLB   -> BUSY (ERROR CODE) \n",*pQ);
                    #endif
            }

            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;

        }

    }

}

void STATE19()
```

```
/*  S_CALLB2  */
{}
void STATE181()
/*  CALLB_DIAL  */
{}
void STATE182()
/*  CALLB_OK  */
{}

void STATE20()
/*  C_CALLB  */
{
    if (pdev->dev_callbdes==0xff){
        pdev->state=BUSY;
        ENTONE(*pQ,tone_busy);
    }else{
        DEV[pdev->dev_callbdes].dev_callb=0xff;
        pdev->dev_callbdes=0xff;
        pdev->state=FEATURE_SET;
        ENTONE(*pQ,tone_set);
    }
}

void STATE60()
/* EX_CALLED */
{
    unsigned char tmp;

    if (pdev->ssi_hook==OFFHOOK){

        STSAC(*pQ);
        pdev->state=TALK;
        #ifdef DEBUG
            printf("DEV(%2d) EX_CALLED -> TALK \n",*pQ);
        #endif

        SSSI_RING(*pQ,OFF);

        pdevtmp=DEV+(tmp=pdev->dev_called);
        pdevtmp->state=TALK;

        if (pdev->dev_typ==OPI){
            SOPI_DIAL(*pQ);
            send_data(pdev->pos >>3 ,_set_tsac,0,pdev->tspcm>>3);
        }

        if (pssi->dev_typ==COI){
            SCO_HOOK(pdev->dev_called,OFF);
        }

        ENTONE(tmp,tone_null);
        CONNECT_DEV(*pQ,tmp);

/*      CONNECT_TSDEV(tmp,*pQ); */
    }
}

void STATE61()
/* HOLDED */
{
    unsigned char tmp;
    if (pdev->ssi_hook==ONHOOK){
        switch (pssi->state){
            case EX_HOLDED:
                SSSI_RING(tmp=pdev->dev_called,OFF);
                ENTONE(tmp,tone_null);
                pssi->state=IDLE;
              break;
            case CT_TALK:
            case CT_FLASH:
/*              pssi->state =TALK;
                pssi->dev_hold=0xff;
                pdevtmp=DEV+pssi->dev_called;
                pdevtmp->state=TALK;
*/              CONNECT_DEV(pssi->dev_called,pdevtmp->dev_called);
              break;
            case CT_RING:
/*              pssi->state=EX_RING; */
                pssi->dev_hold=0xff;
/*              DEV[pssi->dev_called].state=EX_CALLED; */
```

```
                break;
            case FLASH_DIAL:
            case FLASH:
                ENTONE(pdev->dev_called,tone_busy);
                pssi->state=BUSY;
                pssi->dev_hold=0xff;
              break;
        }

        pdev->state=IDLE;
/*      pdev->dev_called=0xff; */
        #ifdef DEBUG
            printf("DEV(%2d) HOLDED    -> IDLE \n",*pQ);
        #endif
        ENTONE(*pQ,tone_null);
/*      pssi->dev_hold=0xff; */
        /* Check Call Back */
        STATE92();


    }
}

void STATE52()
/* CT_EX_CALLED */
{
    unsigned char tmp;
    if (pdev->ssi_hook==OFFHOOK){
        STSAC(*pQ);
        pdev->state=CT_TALK;
        #ifdef DEBUG
            printf("DEV(%2d) CT_EX_CALLED -> CT_TALK \n",*pQ);
        #endif

        SSSI_RING(*pQ,OFF);
        pdevtmp=DEV+(tmp=pdev->dev_called);
        pdevtmp->state=CT_TALK;
        if (pdev->dev_typ==OPI){
            SOPI_DIAL(*pQ);
            send_data(pdev->pos >>3 ,_set_tsac,0,pdev->tspcm>>3);
        }

        ENTONE(tmp,tone_null);
        CONNECT_DEV(*pQ,tmp);
/*      CONNECT_TSDEV(tmp,*pQ);  */
    }
}

void STATE63()
/* DIAL_SPEED */
{
    STATE6();
}

void STATE64()
/* SSPEED */
{
    STATE6();
}

void STATE65()
/* SSPEED_DIAL */
{
}

void STATE66()
/* SSPEED_OK */
{
}

void STATE67()
/* CSPEED */
{
    STATE6();
}

void STATE68()
/* CSPEED_DIAL */
{
}
```

```
void STATE69()
/* CSPEED_OK */
{
}

void STATE78()
/* OPI_HOLD */
{
    if (pdev->ssi_hook==OFFHOOK && pdev->opi_hold==OFF){
        /* end of hold state */
        send_data(pdev->pos >>3 , set_tsac,0,pdev->tspcm>>3);
        if (phold->state==HOLDED && phold->dev_called==*pQ){
            pdev->state=pdev->ostate;
            phold->state=phold->ostate;
            pdev->dev_called=pdev->dev_hold;
            pdev->dev_hold=0xff;
            CONNECT_DEV(*pQ,pdev->dev_called);
        }else{
            pdev->state=BUSY;
            ENTONE(*pQ,tone_busy);
        }
    }

    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
    }

    if (pdev->opi_hold /* Hold flag */ ==ON){
        /* Start to hold device */
        pdev->opi_hold=OFF;
        /* check this case */
        switch(pdev->ostate){
            case EX_RING:
                pdev->state=BUSY;
                pssi->state=IDLE;
                ENTONE(*pQ,tone_busy);
                SSSI_RING(pdev->oev_called,OFF);
              break;
            case TALK:
            case CT_TALK:
            case CT_FLASH:
                /* pdev->ostate=pdev->state; */
                pdev->dev_hold=pdev->dev_called;
                pssi->ostate=pssi->state;
                pssi->state=HOLDED;
                ENMUS(pdev->dev_called);
                ENTONE(*pQ,tone_null);
              break;
            case OUTGOING:
            case CO_DIAL:
                pssi->state=IDLE;
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
                SCO_HOOK(pdev->dev_called,ON);
                RELEASE_DTMF();
              break;
            case IC:
                pdev->state=OPI_HOLD;
                ENMUS(pdev->dev_called);
                pssi->state=HOLDED;
              break;
            case FLASH:
            case FLASH_DIAL:
            default:
                pssi->state=IDLE;
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
                RELEASE_DTMF();
             break;
        }

    }
}

void STATE80()
/* FEATURE_SET */
{
    STATE6();
}
```

```
void STATE31()
/*  SCALLFD */
{
    unsigned char digit;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) SCALLFD   -> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) SCALLFD   -> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==SCALLFD){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++; pdev->state = SCALLFD_SRC;
        }
    }
}

void STATE32()
/* SCALLFD_SRC
 * CALLF DIAL SOURCE/DES
 */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;


    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) SCALLFD_SRC-> IDLE        \n",*pQ);
        #endif

        /* Check Call Back */
        STATE92();

    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) SCALLFD_DES-> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==SCALLFD_SRC){
```

```
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
              /* 3 NUMBER = EX CALL */
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);
            if (DEV[tmp].dev_callf==0xff && tmp!=0xff){
                /* STN FOUND */
                /* X is not CALLF_SET -> get DES# */
                pdev->state=SCALLFD_DES;
                #ifdef DEBUG
                    printf("DEV(%2d) SCALLFD_SRC-> SCALLFD_DES\n",*pQ);
                #endif
                pdev->src=tmp;
                ENTONE(*pQ,tone_sdt);
                pdev->dial_cnt=0;
                pdev->pdial=pdev->dial;
            }else{
                /* X IS CALLF SET */
                pdtmf->state=IDLE;
                pdev->dev_dtmf=0xff;
                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) SCALLFD_SRC-> BUSY  ( no CALLF SET )\n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }

    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE33()
/* SCALLFD_DES */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) SCALLFD_DES-> IDLE       \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) SCALLFD_DES-> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==SCALLFD_DES){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }
        if (pdev->dial_cnt>=MAX_STN){
            /* 3 NUMBER = EX CALL */
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
```

```
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);

            if (tmp!=0xff && tmp!=pdev->src){
                /* DES STN FOUND */
                pdev->state=FEATURE_SET;
                #ifdef DEBUG
                    printf("DEV(%2d) SCALLFD_DES-> FEATURE_OK \n",*pQ);
                #endif
                pssi=DEV+pdev->src; /* SRC */
                pssi->dev_callf=tmp;
                ENTONE(*pQ,tone_sst);
            }else{
                /* NO STN FOUND */

                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) SCALLFD_DES-> BUSY (NO DES STN) \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE34()
/* CCALLFD */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;


    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) CCALLFD     -> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CCALLFD     -> BUSY (X) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==CCALLFD){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;      /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
            /* 3 NUMBER = EX CALL */
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);

            if (DEV[tmp].dev_callf!=0xff && tmp!=0xff){
                /* DES STN FOUND */
                pdev->state=FEATURE_SET;
```

```
                #ifdef DEBUG
                    printf("DEV(%2d) CSCALLFD DES-> FEATURE_OK \n",*pQ);
                #endif
                pssi=DEV+tmp; /* SRC */
                pssi->dev_callf=0xff;
                ENTONE(*pQ,tone_sst);
            }else{
                /* NO STN FOUND */

                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) CCALLFD DES-> BUSY (NO DES STN) \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }

    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE92()
/* CHECK_CALLB_W */
{
    if (pdev->dev_callwait != 0xff){
        /* CALLW */
            pdev->dev_hold =pdev->dev_callwait;
        pdev->state=EX_HOLDED;
        #ifdef DEBUG
            printf("DEV(%2d) CHECK_CALLB_W -> EX_HOLDED \n",*pQ);
        #endif
        pdev->dev_callwait=0xff;
        SSSI_RING(*pQ,ON);
    }else {
        /* NO CALLW */
        if (pdev->dev_callb !=0xff){
            /* CALLB */
            pdevtmp=DEV+pdev->dev_callb;
            if (pdevtmp->state==IDLE ){
                pdevtmp->state=CALLB_RING;
                pdevtmp->dev_hold=*pQ;
                pdev->state=IDLE;
                #ifdef DEBUG
                    printf("DEV(%2d) CHECK_CALLB_W -> CALLB_W_RING (HAVE CALL BACK)\n",*pQ);
                #endif
                SSSI_RING(pdev->dev_callb,ON);
            }else{
                switch (pdev->state){
                    case BUSY:
                    case TALK:
                    case DIAL:
                    case NEXT:
                    case EX_RING:
                      break;
                    default:
                      break;
                }
                pdev->state=IDLE;
                ENTONE(*pQ,tone_null);
            }
            pdev->dev_callb=0xff;
        }else{
            /* NO CALLB & NO CALLW */
            pdev->state=IDLE;
            ENTONE(*pQ,tone_null);
            #ifdef DEBUG
                printf("DEV(%2d) CHECK_CALLB_W -> IDLE \n",*pQ);
            #endif
        }
    }
}

void STATE93()
/* CALLB_RING */
{
    if (pdev->ssi_hook==OFFHOOK){
        /* if EX_HOLDED */
        pdev->dev_called=pdev->dev_hold;
        pdev->dev_hold=0xff;
        pssi=DEV+pdev->dev_called;
```

```
            SSSI_RING(*pQ,OFF);
            SSSI_RING(pdev->dev_called,ON);
            pdev->state=EX_RING;
            pssi->state=EX_CALLED;
            pssi->dev_called=*pQ;
            ENTONE(*pQ,tone_ringb);
            #ifdef DEBUG
                printf("DEV(%2d) CALLB_RING -> EX_RING \n",*pQ);
            #endif
        }
    }
```

```
/*-----------------------------------------
 *   MODULE:  MPU013.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpustate.h"
#include "mpufn.h"

void STATE31();

void STATE21()
/*  S_HOTLINE  */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) CCALLFD    -> IDLE       \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CCALLFD    -> BUSY (x^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==S_HOTLINE){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
              /* 3 NUMBER = EX CALL */
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);
            if (DEV[tmp].hot==0xff && tmp!=0xff && tmp!=*pQ){
                /* DES STN FOUND */
                pdev->state=FEATURE_SET;
                #ifdef DEBUG
                    printf("DEV(%2d) S_HOTLINE    -> FEATURE_OK \n",*pQ);
                #endif
                pdev->hot=tmp;
                ENTONE(*pQ,tone_sst);
            }else{
                /* NO STN FOUND */

                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) CCALLFD_DES-> BUSY (NO DES STN) \n",*pQ);
```

```
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE211()
/*   */
{
}

void STATE212()
/*  HOT_OK  */
{
}

void STATE22()
/*  C_HOTLINE  */
{
    if (pdev->hot==0xff){
        pdev->state=BUSY;
        ENTONE(*pQ,tone_busy);
    }else{
        pdev->hot=0xff;
        pdev->state=FEATURE_SET;
        ENTONE(*pQ,tone_sst);
    }
}

void STATE23()
/*  OPERATOR  */
{
    unsigned char tmp;
    /* CALL OPERATOR */
    pdtmf->state  =IDLE;
    pdev->dev_dtmf=0xff;
    tmp=SEARCH_OPI();
    if (pdev->dev_typ==OPI) tmp=0xff; /* if dev=OPI not connect*/
    if (tmp==0xff){
        /* NO IDLE OPI */
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) COMMAND   -> BUSY     \n",*pQ);
        #endif
            ENTONE(*pQ,tone_busy);
    }else{
        pssi=DEV+tmp;
        pdev->dev_called=tmp;
        pdev->state=EX_RING;
        #ifdef DEBUG
            printf("DEV(%2d) COMMAND   -> EX_RING  \n",*pQ);
        #endif
        pssi->state=EX_CALLED;
        pssi->dev_called=*pQ;
        SSSI_RING(tmp,ON);
        ENTONE(*pQ,tone_ringb);
    }
}

void STATE24()
/*  REDIAL  */
{
    unsigned char i;
    if (pdev->redi_cnt==0){
        pdev->state=BUSY;
        ENTONE(*pQ,tone_busy);
        #ifdef DEBUG
            printf("DEV(%2d) COMMAND   -> REDIAL (ERROR) \n",*pQ);
        #endif
    }else{
        #ifdef DEBUG
            printf("DEV(%2d) COMMAND   -> REDIAL  \n",*pQ);
        #endif
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->dial_cnt=pdev->redi_cnt;
        pdev->pdial=pdev->dial;
        pdev->predi=pdev->redi;
```

```
        for (i=0;i<pdev->dial_cnt;i++)
            *pdev->pdial++=*pdev->predi++;
        if (pdev->dial[0]==9){
            /* OG CALL */
        }else{
            /* CALL_EX */
            if (pdev->dial_cnt==MAX_STN){
                /* CALLEX */
                check_des();
            }else{
                /* ERROR */
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }
        }
    }
}

void STATE25()
/*  HOLD  */
{}

void STATE251()
/*  HOLD_DIAL  */
{}

void STATE252()
/*  HOLD_CALLEX  */
{}

void STATE253()
/*  HOLD_RING  */
{}

void STATE254()
/*  HOLD_TALK  */
 {}

void STATE27()
/*  CONF_DIAL  */
{}

void STATE28()
 /*  CONF_CHECK  */
{}

void STATE29()
 /*  CONF_SET  */
{}

void STATE30()
/*  CALL_PU  */
{

    unsigned int digit;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) CCALLFD     -> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CCALLFD     -> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }
```

```
    /* DIAL DIGIT */

    if (pdev->state==CALL_PU){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
            /* PU_CHECK */
            STATE31();
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE31()
 /*  PUCHECK */
 {
    unsigned int STN;
    unsigned char *p,tmp;

    /* 3 NUMBER = EX CALL */
    pdtmf->state=IDLE;
    pdev->dev_dtmf=0xff;
    p=pdev->dial; STN=0;
    for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
        STN=STN*10 + *p;
    tmp=SEARCH_SSI(STN);
    pdevtmp=DEV+tmp;
    if ( (pdevtmp->state==EX_CALLED || pdevtmp->state==CT_EX_CALLED) && tmp!=0xff){
        /* DES STN FOUND */
        pdev->state=FEATURE_SET;
        #ifdef DEBUG
        printf("DEV(%2d) CSCALLED DES-> FEATURE OK \n",*pQ);
        #endif
        pdevtmp->state=IDLE;
        pssi=DEV+pdevtmp->dev_called;
        pdev->dev_called=pdevtmp->dev_called;
        pssi->dev_called=*pQ;
        if (pssi->state==EX_RING) {
            pssi->state=TALK;
            pdev->state=TALK;
        }else {
            pssi->state=CT_TALK;
            pdev->state=TALK;
        }
        ENTONE(pdev->dev_called,tone_null);
        CONNECT_DEV(*pQ,pdev->dev_called);
        SSSI_RING(tmp,OFF);
    }else{
        /* NO STN FOUND */
        /* EX not ringing */
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CCALLED DES-> BUSY (NO DES STN) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }
}

void STATE32()
/*  HOTLINE  */
{
    unsigned char tmp;
    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) HOTLINE    -> IDLE       \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
        PSSI->STAT =IDLE
    }   SSSI_RING(pdev->dev_called,OFF);
```

```
    /* Flash Dial */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pssi->state=IDLE;
        SSSI_RING(pdev->dev_called,OFF);
        if ( (tmp=SEARCH_DTMF())==0xff){
            /* NO DTMF */
            pdev->state=BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) HOTLINE -> BUSY (no DTMF) \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }else{
            STSAC(tmp);
            pdev ->state=HOTLINE_CNL;
            #ifdef DEBUG
                printf("DEV(%2d)  HOTLINE   -> HOTLINE_CNL  \n",*pQ);
            #endif
            pdtmf=DEV+tmp;
            pdev ->dev_dtmf=tmp;
            pdtmf->dev_dtmf=*pQ;
            pdtmf->state=BUSY;
            pdtmf->dev_digit=0xff;
            CONNECT_TSDEV(*pQ,tmp);
            ENTONE(*pQ,tone_sit);
            pdev->dial_cnt=0;
            pdev->pdial=pdev->dial;
        }
    }
}

void STATE33()
/* HOTLINE_CNL */
{
     unsigned char digit;
     /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) HOTLINE     -> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* Flash Dial */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
    }

    if (pdev->state==HOTLINE_CNL){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;      /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=2){
            if ( (pdev->dial[0]*10+pdev->dial[1])==Code_cnl[7]){
                pdev->state=FEATURE_SET;
                pdev->hot=0xff;
                ENTONE(*pQ,tone_sst);
            }else{
                /* Code ERROR */
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE50()
/*  COI_ONHOOK  */
{}

void STATE51()
/*  COI_OFFHOOK  */
```

```
{}

void STATES2()
/*  COI_FLASH   */
{}

void STATES3()
/*   COI_DIAL   */
{
    if (pdev->a < pdev->dial_cnt){
        SCCI_DIAL(*pG, *(pdev->predi++));
        pdev->dev.dtmf=OFF;
        pdev->a++;
    }
}
```

```
/*-----------------------------------------
 *   MODULE:  MPU014.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include <dos.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpustate.h"
#include "mpufn.h"

void STATE36();
void STATE94();

void STATE34()
/*  OUTGOING  */
{
    unsigned char tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
/*      pdev->state = IDLE; */
        #ifdef DEBUG
            printf("DEV(%2d) DIAL      -> IDLE     \n",*pQ);
        #endif
        pdtmf->state=IDLE;
        pssi->state =IDLE; /* Release COI */
        SCO_HOOK(pdev->dev_called,ON);

        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);

        /* Check Call Back */
        STATE92();
        STATE76();
    }

    /* FLASH */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
    }

/*  pdev->pdial=pdev->dial;
    pdev->dial_cnt=0;
*/
    gettime(&pssi->start);

    if (pdev->state==OUTGOING){
        /* Check ssi -> DTMF/PULSE */
        if ( (tmp=pdev->ssi_digit)!=0xff){
            /* SSI PULSE */
            pdev->tmp=1;
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            pssi->pdial=pssi->dial;
            pssi->dial_cnt=1;
            pdev->dial_cnt++;
            *(pdev->pdial++)=tmp;
            *(pssi->pdial++)=tmp;
            pdev->state=OG_DIAL;
            if (pssi->redi_cnt /* CO type */ ==0){
                /* COI DTMF */
                CONNECT_TSDEV(*pQ,pdev->dev_dtmf);
                CONNECT_DEV(pdev->dev_called,*pQ);
                SEND_DTMF(pdev->dev_called,tmp);
                pssi->state=BUSY;
            }else{
                /* COI PULSE */
                pdtmf->state=IDLE;
                pdev->dev_dtmf=0xff;
                CONNECT_TSDEV(pdev->dev_called,*pQ);
                SCO_DIAL(pdev->dev_called,tmp);
                pssi->dev_dtmf /* dial_rdy */ =OFF;
                pssi->state=COI_DIAL;
```

```
                pssi->predi=pssi->dial+1;
                pssi->a=1;  /* COI DIAL COUNT */
            }
        }else{
            if ( (tmp= pdtmf->dev_digit) !=0xff){
                /* DTMF */
                pdev->tmp=0;    /* SSI->dtmf */
                pdtmf->dev_digit=0xff;
                pssi->pdial=pssi->dial;
                pssi->dial_cnt=1;

                pdev->dial_cnt++;
                *(pdev->pdial++)=tmp;
                *(pssi->pdial++)=tmp;
                pdev->state=OG_DIAL;

                if (pssi->redi_cnt /* CO type */ ==0){
                    /* COI DTMF */
                    CONNECT_DEV(pdev->dev_called,*pQ);
/*                  pdtmf->state=IDLE; */
/*                  pdev->dev_dtmf=0xff; */
                }else{
                    /* COI PULSE */
                    CONNECT_TSDEV(*pQ,pdev->dev_dtmf);
                    CONNECT_TSDEV(pdev->dev_called,*pQ);
                    SCO_DIAL(pdev->dev_called,tmp);
                    pssi->dev_dtmf /* dial_rdy */ =OFF;
                    pssi->state=COI_DIAL;
                    pssi->predi=pssi->dial+1;
                    pssi->a=1;  /* COI DIAL COUNT */
                }
            }
        }
    }/* endif state=outgoing */
}

void STATE35()
/*  OG_DIAL  */
{
    unsigned char digit,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        /*pdev->state = IDLE; */
        #ifdef DEBUG
            printf("DEV(%2d) DIAL       -> IDLE      \n",*pQ);
        #endif
        pdtmf->state=IDLE;
        pssi->state =IDLE; /* Release COI */
        SCO_HOOK(pdev->dev_called,ON);
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);

        /* Check Call Back */
        STATE76();
        STATE92();
    }

    /* FLASH */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdev->dev_hold=pdev->dev_called;
        ENMUS(pdev->dev_hold);
/*      tmp=SEARCH_DTMF(); */
        tmp=pdev->dev_dtmf;
        if ( tmp==0xff){
            /* No DTMF */
            pdev->state=HOLD_BUSY;
            #ifdef DEBUG
                printf("DEV(%2d) TALK       -> HOLD BUSY  \n",*pQ);
            #endif
            ENTONE(*pQ,tone_busy);
        }else{
            /* DTMF found */
            if (pdev->dev_typ!=OPI){
                STSAC(tmp);
                pdev->dev_dtmf = tmp;
                pdtmf=DEV+tmp;
                pdtmf->state=BUSY;
                pdtmf->dev_dtmf=*pQ;
```

```
                pdtmf->dev_digit=0xff;
                CONNECT_TSDEV(*pQ,tmp);
            }
            pdev->state=FLASH;
            #ifdef DEBUG
                printf("DEV(%2d) TALK      -> FLASH      \n",*pQ);
            #endif
            pssi->state=HOLDED;
            ENTONE(*pQ,tone_dial);
            pdev->dial_cnt=0;
            pdev->pdial=pdev->dial;
        }
    }

    /* DIAL DIGIT */
    if (pdev->state==OG_DIAL && (digit=GET_DIGIT(*pQ))!=0xff) {
        *(pdev->pdial++)=digit;     /* save digit to dial array */
        (pdev->dial_cnt)++;

        /* Check ssi -> DTMF/PULSE */
        if ( pdev->tmp==0){
            /* DTMF */

            pssi->dial_cnt++;
            *(pssi->pdial++)=digit;

            pdev->state=OG_DIAL;

            if (pssi->redi_cnt /* CO type */ ==0){
                /* CO1 DTMF */
            }else{
                /* CO1 PULSE */
                if (pssi->dev_dtmf==ON && pdev->a <pdev->dial_cnt){
                    SCO_DIAL(pdev->dev_called,*(pssi->predi++));
                    pssi->a++;
                    pssi->dev_dtmf=OFF;
                }
            }
        }else{
            if (pdev->tmp==1){
                /* SSI PULSE */

                pssi->dial_cnt++;
                *(pssi->pdial++)=digit;
                pdev->state=OG_DIAL;

                if (pssi->redi_cnt /* CO type */ ==0){
                    /* CO1 DTMF */
                    SEND_DTMF(pdev->dev_called,digit);
                    CONNECT_DEV(pdev->dev_called,*pQ);
                }else{
                    /* CO1 PULSE */
                    if (pssi->dev_dtmf==ON && pdev->a <pdev->dial_cnt){
                        SCO_DIAL(pdev->dev_called,*(pssi->predi++));
                        pssi->a++;
                        pssi->dev_dtmf=OFF;
                    }
                }
            }/* ssi_digit=0xff */
        }
        STATE36();
    }
}


void STATE35()
/*  OG_CHECK  */
{
    switch (pdev->svc){
        case 1:
            if (pdev->dial_cnt>15){
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
                pssi->state =IDLE; /* Release CO1 */
                SCO_HOOK(pdev->dev_called,ON);
            }
          break;
        case 2:
            /* IN,OG,LONG */
```

```
            if (pdev->dial_cnt>11){
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
                pssi->state =IDLE; /* Release COI */
                SCO_HOOK(pdev->dev_called,ON);
            }
          break;
        case 3:
            /* IN,OG */
            if (pdev->dial_cnt>3){
                pdev->state=BUSY;
                pssi->state=IDLE;
                SCO_HOOK(pdev->dev_called,ON);
                ENTONE(*pQ,tone_busy);
            }
          break;
    }
}

void STATE37()
/*  OG_OK     */
{
}
```

```
/*-----------------------------------------
 *   MODULE:  MFU015.C
 *   REL    :  1.0
 *   AUTHOR:  Surasak Uthayopas
 *   DATE   :  28 December 1990
 * ----------------------------------------
 */

#include <stdio.h>
#include <dos.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpustate.h"
#include "mpufn.h"

void STATE39();

void STATE38()
/*  IC  */
{
    if (pdev->col_ring==ON){
        /*  SCO_HOOK(*pQ,OFF); */
        STATE39();
    }
}

void STATE39()
/*  CHECK_NIGHT  */
{
    unsigned char tmp;

    gettime(&pdev->start);

    if (Night_flag){
        /* Night Connection */
        if ( (tmp=SEARCH_NIGHT())==0xff){
            /* Night busy */
            SCO_HOOK(*pQ,OFF);
            SCO_HOOK(*pQ,ON);
            pdev->state=IDLE;
        }else{
            pdev->state=IC_EX_RING;
            pdev->dev_called=tmp;
            pssi=DEV+tmp;
            pssi->state=EX_CALLED;
            pssi->dev_called=*pQ;
            SSSI_RING(pdev->dev_called,ON);
            ENTONE(*pQ,tone_ringb);
        }
    }else{
        if ( (tmp=SEARCH_OPI())==0xff){
            /* OPI busy */
            SCO_HOOK(*pQ,OFF);
            SCO_HOOK(*pQ,ON);
            pdev->state=IDLE;
        }else{
            pdev->state=IC_EX_RING;
            pdev->dev_called=tmp;
            pssi=DEV+tmp;
            pssi->state=EX_CALLED;
            pssi->dev_called=*pQ;
            SSSI_RING(pdev->dev_called,ON);
            ENTONE(*pQ,tone_ringb);
        }
    }
}

void STATE40()
/*  NIGHT_RING  */
{
}

void STATE41()
/*  NIGHT_OK  */
{
}

void STATE42()
```

```
/*  OPI_H_SRC  */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;


    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) OPI_H_SRC-> IDLE        \n",*pQ);
        #endif

        /* Check Call Back */
        /* STATE92(); */
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state   =BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) OPI_H_SRC-> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==OPI_H_SRC){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
              /* 3 NUMBER = EX CALL */
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);
            if (DEV[tmp].hot==0xff && tmp!=0xff){
                /* STN FOUND */
                pdev->state=OPI_H_DES;
                #ifdef DEBUG
                    printf("DEV(%2d) OPI_H_SRC-> OPI_H_DES\n",*pQ);
                #endif
                pdev->src=tmp;
                ENTONE(*pQ,tone_sdt);
                pdev->dial_cnt=0;
                pdev->pdial=pdev->dial;
            }else{
                /* X IS HOTLINE SET */
                pdtmf->state=IDLE;
                pdev->dev_dtmf=0xff;
                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) OPI_H_SRC-> BUSY  ( no HOT SET )\n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }

    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE43()
/*  OPI_H_DES  */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
```

```
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) OPI_H_DES-> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        /* STATE92(); */
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) OPI_H_DES-> BUSY (X) ; \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==OPI_H_DES){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;      /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
              /* 3 NUMBER = EX CALL */
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);

            if (tmp!=0xff && tmp!=pdev->src){
                /* DES STN FOUND */
                pdev->state=FEATURE_SET;
                #ifdef DEBUG
                    printf("DEV(%2d) OPI_H_DES-> FEATURE_OK \n",*pQ);
                #endif
                pssi=DEV+pdev->src; /* SRC */
                pssi->hot=tmp;
                ENTONE(*pQ,tone_set);
            }else{
                /* NO STN FOUND */
                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) OPI_H_DES-> BUSY (NO DES STN) \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE44()
/*  OPI_CHOT  */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) OPI_CHOT  -> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        /* STATE92(); */
    }
```

```
    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) OPI_CHOT  -> BUSY (X) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */
    if (pdev->state==OPI_CHOT){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
            /* 3 NUMBER = EX CALL */
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);
            if (DEV[tmp].hot!=0xff && tmp!=0xff){
                /* DES STN FOUND */
                pdev->state=FEATURE_SET;
                #ifdef DEBUG
                    printf("DEV(%2d) OPI CHOT  -> FEATURE OK \n",*pQ);
                #endif
                pssi=DEV+tmp; /* SRC */
                pssi->hot=0xff;
                ENTONE(*pQ,tone_sst);
            }else{
                /* NO STN FOUND */

                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) OPI CHOT  -> BUSY (NO DES STN) \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}


void STATE45()
/*  IC_EX_RING  */
{
    if (pdev->coi_ring==OFF) {
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) IC_EX_RING   -> IDLE  \n",*pQ);
        #endif
        pssi->state = IDLE;

        SSSI_RING(pdev->dev_called,OFF);
        ENTONE(*pQ,tone_null);
    }
}

void STATE47()
/*  CAMP_ON  */
{}

void STATE46()
/*  CO_TALK  */
{}

void STATE70()
/*  CONDCI */
{
    unsigned char digit;
```

```
    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) CONDCI    -> IDLE       \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CONDCI    -> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==CONDCI){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
            pdev->state=CONDCI_SRC;
            ENTONE(*pQ,tone_null);
        }
    }
}

void STATE71()
/* CONDCI_SRC
 * DCI DIAL SOURCE
 */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) CONDCI_SRC-> IDLE       \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CONDCI_SRC-> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==CONDCI_SRC){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
              /* 3 NUMBER = EX CALL */
```

```
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);
            pdevtmp=DEV+tmp;
            if (pdevtmp->dev_typ==DCI && tmp!=0xff && pdevtmp->state==IDLE){
                /* DCI FOUND */
                pdev->state=CONDCI_DES;
                #ifdef DEBUG
                    printf("DEV(%2d) CONDCI_SRC-> CONDCI_DES\n",*pQ);
                #endif
                pdev->src=tmp;
                ENTONE(*pQ,tone_sdt);
                pdev->dial_cnt=0;
                pdev->pdial=pdev->dial;
            }else{
                /* DCI BUSY or DCI no found or STN is not DCI */
                pdtmf->state=IDLE;
                pdev->dev_dtmf=0xff;
                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) CONDCI_SRC-> BUSY  ( no DCI  SET )\n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE72()
/* CONDCI_DES */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) CONDCI_DES -> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) CONDCI_DES -> BUSY (X^) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==CONDCI_DES){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
              /* 3 NUMBER = EX CALL */
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);

            pdevtmp=DEV+tmp;
            if ( (tmp!=0xff /* || tmp==pdev->src */) /* && pdevtmp->dev_typ==DCI && pdevtmp->state==IDLE
```

```
                /* DES STN FOUND ,DES=DCI , DES=IDLE */
                pdev->state=FEATURE_SET;
                #ifdef DEBUG
                    printf("DEV(%2d) CONDCI DES -> FEATURE_OK \n",*pQ);
                #endif
                pssi=DEV+pdev->src; /* SRC */
                pssi->state=BUSY;
                pssi->dev_called=tmp;
                pdevtmp->state=BUSY;
                pdevtmp->dev_called=pdev->src;
                STSAC(pdev->src);
                STSAC(tmp);
                CONNECT_DEV(tmp,pdev->src);
                ENTONE(*pQ,tone_sst);
            }else{
                /* NO STN FOUND */

                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) CONDCI DES -> BUSY (NO DES STN) \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }
        }
    }
    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE73()
/* DISDCI */
{
    unsigned int digit,STN;
    unsigned char *p,tmp;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) DISDCI      -> IDLE        \n",*pQ);
        #endif
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) DISDCI      -> BUSY (X) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    /* DIAL DIGIT */

    if (pdev->state==DISDCI){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
            (pdev->dial_cnt)++;
        }

        if (pdev->dial_cnt>=MAX_STN){
              /* 3 NUMBER = EX CALL */
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
            p=pdev->dial; STN=0;
            for (tmp=0;tmp<pdev->dial_cnt;tmp++,p++)
                STN=STN*10 + *p;
            tmp=SEARCH_SSI(STN);

            pssi=DEV+tmp;
            if (pssi->dev_typ==DCI && pssi->state==BUSY && tmp!=0xff){
                /* DES STN FOUND */
                pdev->state=FEATURE_SET;
```

```
                #ifdef DEBUG
                    printf("DEV(%2d) DISDCI      -> FEATURE_OK \n",*pQ);
                #endif
                pdevtmp=DEV+pssi->dev_called;
                pssi   ->state=IDLE;
                pdevtmp->state=IDLE;
                pdev   ->src=pssi->dev_called; /* DES */

                pssi   ->dev_called=0xff;
                pdevtmp->dev_called=0xff;

                ENTONE(tmp,tone_null);
                ENTONE(pdev->src,tone_null);
                ENTONE(*pQ,tone_sst);
            }else{
                /* NO STN FOUND */

                pdev->state=BUSY;
                #ifdef DEBUG
                    printf("DEV(%2d) DISDCI      -> BUSY (NO DES STN) \n",*pQ);
                #endif
                ENTONE(*pQ,tone_busy);
            }

        }
    }

    if (pdev->dial_cnt==1) ENTONE(*pQ,tone_null);
}

void STATE74()
/*  OG_BUSY */
{

    unsigned char i,tmp;
    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
      /* SAVE OLD NUMBER */
        if (pdev->dial[0]!=11){
            pdev->redi_cnt=pdev->dial_cnt;
            pdev->pdial=pdev->dial;
            pdev->predi=pdev->redi;
            for (i=0;i<pdev->dial_cnt;i++)
                *pdev->predi++=*pdev->pdial++;
        }

        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) EX_BUSY   -> IDLE      \n",*pQ);
        #endif
        pdtmf->state  =IDLE;
        pdev->dev_dtmf=0xff;
        ENTONE(*pQ,tone_null);
        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdev->dial_cnt=0;
        pdev->pdial=pdev->dial;
        pdev->state=OG_SCALLB;
        ENTONE(*pQ,tone_sdt);
        #ifdef DEBUG
            printf("DEV(%2d) EX_BUSY   -> S_CALLB    \n",*pQ);
        #endif
    }
}

void STATE75()
/*  OG_SCALLB */
{
    unsigned char tmp,*p,i,digit;
    int NUM;

    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pQ,tone_null);
        pdtmf->state  =IDLE;
```

```
        pdev->dev_dtmf=0xff;
        pdev->state = IDLE;
        #ifdef DEBUG
            printf("DEV(%2d) S_CALLB   -> IDLE        \n",*pQ);
        #endif

        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
        pdtmf->state=IDLE;
        pdev->dev_dtmf=0xff;
        pdev->state=BUSY;
        #ifdef DEBUG
            printf("DEV(%2d) S_CALLB   -> BUSY (X) \n",*pQ);
        #endif
        ENTONE(*pQ,tone_busy);
    }

    if (pdev->state==OG_SCALLB){
        if ( (digit=GET_DIGIT(*pQ))!=0xff) {
            *(pdev->pdial++)=digit;     /* save digit to dial array */
             (pdev->dial_cnt)++;
             ENTONE(*pQ,tone_null);
        }
        if (pdev->dial_cnt==2){
            NUM=pdev->dial[0]*10 + pdev->dial[1];
            p=Code_set ,tmp=0xff;
            for (i=0;i<MAX_FEATURE;i++,p++)
                if (NUM==*p) {
                    tmp=i; break;
                }
            NUM=tmp;
            if (NUM==0){
                if (coQ_cnt<MAX_COQ){
                    *pcoQ++=*pQ;
                    coQ_cnt++;
                    pdev->state=FEATURE_SET;
                    ENTONE(*pQ,tone_sst);
                    #ifdef DEBUG
                        printf("DEV(%2d) S_CALLB   -> FEATURE_SET (CALLB SET) \n",*pQ);
                    #endif
                }
            }else{
                pdev->state=BUSY;
                ENTONE(*pQ,tone_busy);
                #ifdef DEBUG
                    printf("DEV(%2d) S_CALLB   -> BUSY (ERROR CODE) \n",*pQ);
                #endif
            }
            pdtmf->state=IDLE;
            pdev->dev_dtmf=0xff;
        }
    }
}

void STATE76()
/* OG_CALLB */
{
    unsigned char i,j,tmp,*p,*pp;

    printf("COQ : ");
    for (i=0;i<MAX_COQ;i++) printf("%2d ",coQ[i]);
    printf("\n");

    if (/* pdev->state==OG_DIAL && */ pssi->start.ti_hour!=0xff){
        getdate(&OGbuff.today);
        gettime(&OGbuff.now);
        p=OGbuff.number;
        pp=pssi->dial;
        for (i=0;i<=pssi->dial_cnt;i++,p++,pp++)
            *p=*pp+'0';
        *(--p)=0;
        fprintf(fpt,"DATE %2d/%2d/%2d  TIME %02d:%02d-%02d:%02d   STN : %3d  DIAL : %s\n",
            OGbuff.today.da_mon,OGbuff.today.da_day,OGbuff.today.da_year,
            pssi->start.ti_hour,pssi->start.ti_min,
            OGbuff.now.ti_hour,OGbuff.now.ti_min,pdev->stn,OGbuff.number
```

```
            );
        if (print)
            fprintf(prn,"DATE %2d/%2d/%2d  TIME %02d:%02d-%02d:%02d   STN : %3d  DIAL : %s\n",
            OGbuff.today.da_mon,OGbuff.today.da_day,OGbuff.today.da_year,
            pssi->start.ti_hour,pssi->start.ti_min,
            OGbuff.now.ti_hour,OGbuff.now.ti_min,pdev->stn,OGbuff.number
            );

        printf("DATE %2d/%2d/%2d  TIME %02d:%02d-%02d:%02d   STN : %3d  DIAL : %s\n",
            OGbuff.today.da_mon,OGbuff.today.da_day,OGbuff.today.da_year,
            pssi->start.ti_hour,pssi->start.ti_min,
            OGbuff.now.ti_hour,OGbuff.now.ti_min,pdev->stn,OGbuff.number
            );
        pssi->start.ti_hour=0xff;
        pssi->dial_cnt=0;
    }

    SCO_HOOK(pdev->dev_called,ON);
    pssi->state=IDLE;

    if (coQ_cnt>0){
        for (i=0;i<MAX_COQ;i++){
            pdevtmp=DEV+(tmp=coQ[i]);
            if (pdevtmp->state==IDLE && tmp!=0xff){
                pdevtmp->state=OG_CALLB_RING;
                pdevtmp->dev_called=pdev->dev_called;
                pssi->state=BUSY;
                SSSI_RING(tmp,ON);

                for (j=i;j<MAX_COQ-1;j++){
                    coQ[j]=coQ[j+1];
                }
            }
        }
    }
}

void STATE77()
/* OG_CALLB_RING */
{
    if (pdev->ssi_hook==OFFHOOK){
        pdev->state=OUTGOING;
        SCO_HOOK(pdev->dev_called,OFF);
        SSSI_RING(*pQ,OFF);
        CONNECT_DEV(*pQ,pdev->dev_called);
    }
}
```

```
/*-----------------------------------------
*   MODULE:  MPU016.C
*   REL    :  1.0
*   AUTHOR:  Surasak Uthayopas
*   DATE   :  28 December 1990
* -----------------------------------------
*/

#include <stdio.h>
#include "mpu.h"
#include "mpuintf.h"
#include "mpucmd.h"
#include "mpuvar.h"
#include "mpustate.h"
#include "mpufn.h"

void STATE26()
/*  CONFERENCE  */
{
    /* ONHOOK */
    if (pdev->ssi_hook==ONHOOK) {
        ENTONE(*pU,tone_null);
        pdev->state=IDLE;
        pssi->state=IDLE;

        #ifdef DEBUG
            printf("DEV(%2d) CONFERENCE-> IDLE       \n",*pQ);
        #endif

        /* Release Conference */
        DEV[pdev->dev_called].state=IDLE;
        pdevtmp=pssi;
        pssi=DEV+pdevtmp->b;
        phold=DEV+pdevtmp->c;
        pssi->state=TALK;
        phold->state=TALK;
        pssi->dev_called=pdevtmp->c;
        phold->dev_called=pdevtmp->b;

        CONNECT_DEV(pdevtmp->b,pdevtmp->c);

        /* Check Call Back */
        STATE92();
    }

    /* FLASH HOOK */
    if (pdev->ssi_flash==ON){
        pdev->ssi_flash=OFF;
    }
}
```

```
          list  a,e
          nlist m
;-------------------------------------------------------------
;   8 SSI Module
;   Name        : ssi010.src
;   Description : 8SSI module Control Program
;   Version     : 10
;   Description : Each line circuit (lc) can dial to the other.
;                 If called lc is idle ,called lc will ringing.
;                 if called lc is busy ,calling cancel.
;   Date        : 24 January 1991
;-------------------------------------------------------------
;   8SSI-PPU I/O Port
;
P8255           equ 90h
P8253           equ 80h
;   Port 8255
P8255A          equ P8255
P8255B          equ P8255+1
P8255C          equ P8255+2
P8255CMD        equ P8255+3

;   Port 8253
P8253C0         equ P8253
P8253C1         equ P8253+1
P8253C2         equ P8253+2
P8253CMD        equ P8253+3
;   Port TSAC
PTSAC_CS        equ P8255A
PTSAC_DI        equ P8255C
PTSAC_CTS       equ P8255C
PRING_EN        equ 0eh
PHOOK_STATUS    equ 0fh
;-------------------------------------------------------------
;   Constant
;
MAX_COMMAND     equ 240     ; Maximum command
MAX_LC          equ 8       ; Maximum SSI module
;-------------------------------------------------------------
;   8SSI-Command

SSI_Dev_Num     equ 03h     ;Device number
Dev_Num         equ  SSI_Dev_Num
Chk_Dev         equ 20h     ;Check device number
Set_TSAC        equ 30h     ;Set TSAC
Set_PCM         equ 31h     ;Set PCM
Off_Hook        equ 32h     ;SSI off-hook
On_Hook         equ 33h     ;SSI on-hook
Flash           equ 34h     ;SSI flash
Dial_P          equ 35h     ;SSI dial pulse
Test_Ring       equ 36h     ;Ringing test
En_ringlc       equ 37h     ;Enable ring
dis_ringlc      equ 38h     ;Disable ring
set_test        equ 39h     ;Test Module
mod_fail        equ 26h
ssi_click       equ 27h     ;SSI set a short ring
-------------------------------------------------------------
;   Time Constant

C_on            equ 50      ;On-hook
C_off           equ 10      ;Off-hook
C_fmin          equ 9       ;Flash Minimum time
C_fmax          equ 49      ;Flash Maximum time
C_digit         equ 8       ;Dial digit
Dial digit
C_inter         equ 20      ;Inter-digit
-------------------------------------------------------------
;   Inteface PPU-MPU

NF_H_OUT        equ 0aH       ;MPU Inteface Handshake output
NF_H_IN         equ 09H       ;MPU Inteface Handshake input
NF_DATA         equ 0bH       ;MPU Inteface Data
-------------------------------------------------------------
;   I/O HANDSHAKE SIGNAL
;   <INPUT>         VALUE
PU_EN           equ 02H     ;PPU enable           BIT 1
EQ_ACK          equ 04H     ;Request acknowledge  BIT 2
ST_PM           equ 08H     ;Reset PPU            BIT 3
```

```
;   <OUTPUT>          BIT#
NAK_ACK         equ 0       ;Nak-Ack - data correct;        BIT 0
DATA_RDY        equ 1       ;Data ready                     BIT 1
REQ             equ 2       ;Request from PPU               BIT 2
RST_CNT         equ 3       ;Reset inteface address counter BIT 3
;-----------------------------------------------------------
;   ASCII Control code
;
NULL            equ 00H     ;NULL
ACK             equ 06H     ;Ack
NAK             equ 15H     ;Nak
;-----------------------------------------------------------
;   Line Circuit Table
;
table_len       equ 16          ;Table length
tstatus         equ 0           ;SSI on/off hook status
ttimer          equ 1           ;Down counter
tpulse_cnt      equ 2           ;Pulse counter
tstr_cnt        equ 3           ;String count
tstr_len        equ table_len-tstr_cnt  ;Must less than 256 byte
tsflash         equ 00001000b   ;Flash flag
tsonoff         equ 00000100b   ;On/off hook flag
tsold           equ 00000010b   ;Old hook-status flag
tsnew           equ 00000001b   ;New hook-status
;-----------------------------------------------------------
        File    ssi010_0.src
;
        org     0h              ;Start program
        di                      ;Disable interrupt
        im      1               ;Set interrupt mode 1
        ld      sp,stack_pt     ;Load stack pointer
        ld      b,0
        djnz    $               ;Power on delay
        jp      MAIN
;
        int_service             ;Macro interrupt service routine
;
        org     100h
;-----------------------------------------------------------
MAIN:       ; MAIN
;Description : Main Program
;Parameter   : none
;Return Value: none
;Used Reg    : none
;Call Sub    :
;
        call    init_8ssi       ;Initialize I/O port
        call    clear_buff      ;Clear command buffer
        call    init_ring       ;Initialize ring counter
        call    init_lctable    ;Initialize lc data table
        call    init_table      ;Initialize PCM / TS table
        call    init_tsac       ;Clear tx/rx ts
        call    prg_8253        ;Program NMI interrupt interval
        EI
main1:  call    exec_com        ; Execute command send from MPU
        ld      a,( test_module);if ( test_module)=ack then run test_mod
        cp      ack
        jp      nz,mainle
        call    test_mod
mainle: halt
        jp      main1

        File ssi010_1.src
        File ssi010_2.src
        File ssi010_3.src
-----------------------------------------------------------
   Standard PCM/TS table

pcm_std:    defb    0
s_table_std:
            defb    0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh
c_click     defb    18
-----------------------------------------------------------
        org     2000h
-----------------------------------------------------------
   Start of RAM

 IN_ADD:    defs    1       ;Handshake input status
_OUT_ADD:   defs    1       ;Handshake output status
EQ_ADD:     defs    1       ;Request flag
```

```
HOOK_ADD:    defs    1          ;Hook status flag
WR_Index:    defs    2          ;Wr cbuff pointer
Rd_Index:    defs    2          ;Rd cbuff pointer
Hook_state   defs    1
Scan_add     defs    1
lc_index     defs    1          ;lc pointer
ring_cnt     defs    2          ;Ring counter
Ring_add     defs    1          ;Ringging flag
ringing_stu  defs    1          ;Ringging state (ring / no ring)
ring_cnt1    defs    2*max_lc   ;Ring counter
old_stu      defs    max_lc     ;Old hook status
 test_module defs    1          ;Test flag
click_add    defs    1
click_cnt    defs    max_lc
;-------------------------------------------------------------
 pcm         defs    1          ;pcm line address
ts_table:
 ts_ssi0     defs    1          ;SSI0 ts#
 ts_ssi1     defs    1          ;SSI1 ts#
 ts_ssi2     defs    1          ;SSI2 ts#
 ts_ssi3     defs    1          ;SSI3 ts#
 ts_ssi4     defs    1          ;SSI4 ts#
 ts_ssi5     defs    1          ;SSI5 ts#
 ts_ssi6     defs    1          ;SSI6 ts#
 ts_ssi7     defs    1          ;SSI7 ts#
;-------------------------------------------------------------
        org     2100h
RD_CBUFF:    defs    max_command
;
        org     2200h
WR_CBUFF:    defs    max_command
temp_cbuff   defs    max_command
;
        org     2500h
;-------------------------------------------------------------
;   Line Circuit Data table
;
LC_table:    defs    table_len*max_lc
;
        org     3fffh
STACK_PT:    defs    1
;
        end
        end
```

```
;-------------------------------------------------------------
;   8 SSI Module
;   Name        : SSI010_0.SRC
;   Description : Macro Bios
;   Version     : 10
;   Date        : 24 January 1991
;-------------------------------------------------------------
Int_service    Macro
;Description : Interrupt service routine
;            :  - Service all interrupt
;Parameter   : none
;Return Value: none
;Used Reg    : all
;Call Sub    : -
;
RST28:      org   28H
            reti
RST30:      org   30H
            reti
RST38:      org   38H             ; Service communication
            di
            jp    Comm_service
            reti
NMI:        org   66H             ; Scan & analysis SSI status
            jp    NMI_Service
            retn
            endM
;-------------------------------------------------------------
Reset_Cnt     Macro
;Description : Reset counter in MPU-PPU interface
;Parameter   : none
;Return Value: none
;Used Reg    : hl,b,a
;Call Sub    : none

            ld    hl,H_out_ADD
            res   RST_CNT,(hl)
            out   (INF_H_out),a
            ld    b,0AH           ;20 MICROSEC
            djnz  $
            set   RST_CNT,a
            out   (INF_H_out),a
            ld    (hl),a
            endM
;-------------------------------------------
Dis_H_out     Macro  #H_out
;Description : Disable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none
;
            ld    hl,H_out_ADD
            set   #H_out,(hl)
            ld    a,(hl)
            out   (INF_H_out),a
            endM
;-------------------------------------------
En_H_out     Macro  #H_out
;Description : Enable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none

            ld    hl,H_out_ADD
            res   #H_out,(hl)
            ld    a,(hl)
            out   (INF_H_out),a
            endM
;-------------------------------------------
Wr_Inf         Macro
;Description : Copy C_Buff from PPU to MPU
;Parameter   : none
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    :
            LOCAL #WR_INF_1
            ld    hl,WR_CBUFF       ;Byte_count
            ld    b,(hl)
```

```
            inc   b                 ; add one byte for Byte_count
#WR_INF_1:  ld    a,(hl)
            out   (INF_DATA),a
            inc   hl
            djnz  #WR_INF_1
            endM
;---------------------------------------
Add_Wr_Cbuff  Macro #data
;Description : Add a data to wr cbuff
;Parameter   : #data
;Return Value: none
;Used Reg    : a,hl
;Call Sub    : none
              ld    a,#DATA
              ld    (hl),a
              inc   hl
              endM
;---------------------------------------
Clr_Cbuff     Macro #Cbuff
;Description : Clear Command Buffer
;Parameter   : #cbuff
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    : none
            LOCAL   #L1
            ld    hl,#CBUFF
            ld    b,MAX_COMMAND+1
            xor   a
#L1:        ld    (hl),a
            inc   hl
            djnz  #L1
            endM
;---------------------------------------
Set_Req       Macro
;Description : Activate Request signal
;Parameter   : none
;Return Value: none
;Used Reg    : a
;Call Sub    : En_h_out
            ld      a,ACK
            ld      (Req_add),a
            En_H_out  REQ
            endM
;---------------------------------------
Inc_index     Macro #index
;Description : Increment rd_index or Wr_index
;Parameter   : #index
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
            push    hl
            ld      hl,(#index)
            inc     hl
            inc     hl
            inc     hl
            ld      (#index),hl
            pop     hl
            endM
;---------------------------------------
Inc_BC        Macro
;Description : Increment Byte count
;Parameter   : none
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
            ld      a,(wr_cbuff)
            add     a,3
            ld      (wr_cbuff),a
            endM
            endf
```

```
;-------------------------------------------------------------
;   8 SSI Module
;   Name        : SSI010 1.SRC
;   Description : Bios
;   Version     : 10
;   Date        : 24 January 1991
;-------------------------------------------------------------
Clr_Wr_Index:;  Clear wr index
;Description :  Clear write commad buffer index,Clear byte count
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,a
;Call Sub    :  none
;
        ld      hl,wr_cbuff+1
        ld      (Wr_index),hl
        ld      a,0
        ld      (wr_cbuff),a
        ret
;-------------------------------------------------------------
Clr_rd_index:;  Clear rd index
;Description :  Clear read command buffer index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl
;Call Sub    :  none
;

        ld      hl,rd_cbuff+1
        ld      (Rd_index),hl
        ret
;-------------------------------------------------------------
CHK_BUSY:     ;  Busy check
;Description :  check lc#xx busy?
;Parameter   :  b=lc#
;Return Value:  a=1->busy   a=0->idle
;Used Reg    :  hl,de,b,a
;Call Sub    :
;
        inc     b
        ld      hl,lc_table
        ld      de,table_len
        xor     a           ;Clear carry flag
        sbc     hl,de
cbusy1: add     hl,de
        djnz    cbusy1
        ld      a,(hl)
        and     tsonoff     ;check off-hook
        jr      z,cbusy2
        ld      a,1         ;off-hook == busy
        jr      cbusy3
cbusy2: xor     a           ;on-hook  == idle
cbusy3: ret
;-------------------------------------------------------------
CONNECT:      ;  Connect
;Description :  check dial digit ,Check busy and Connect TS
;Parameter   :  (lc_index)=lc#,ix=start of table
;Return Value:  None
;Used Reg    :  a,b,c
;Call Sub    :  chk_busy,connect_lc,en_ring
;
        ld      a,(lc_index)
        ld      b,a             ;Check LC on/off hook
        call    chk_busy        ;  if a=1->offhook(OK) else return
        and     a
        jr      z,connectE
        ld      a,(ix+tstr_cnt) ;Check String counter=0 ?
        and     a               ;   if no digit then return
        jr      z,connectE
        ld      b,(ix+tstr_cnt+1)  ;Get First digit (Destination lc)
                                   ; This step digit is 1-8
        dec     b                  ; This step digit is 0-7
        call    chk_busy           ;  Check this Des Lc busy ?
        and     a                  ;  If a=1 ->busy
        jr      nz,connect1        ;  If busy then clear string
        ld      b,(ix+tstr_cnt+1)
        dec     b
        call    en_ring            ; Enable ring for Des lc
        ld      a,(lc_index)       ; Connect TS from source to des
        ld      b,a
```

```
        ld      c,(ix+tstr_cnt+1)
        dec     c
        call    connect_lc
connect1: xor   a
        ld      (ix+tstr_cnt),a
connectE:  ret
;-------------------------------------------------------------
Disconnect:  ;  Disconnect
;Description :  If source LC on-hook then clear Ts,clear ring
;Parameter   :  (lc_index)=lc#.ix=start of table
;Return Value:  None
;Used Reg    :  a,b,c
;Call Sub    :  prg_ts,chk_busy,dis_ring
;
        ld      a,(lc_index)
        ld      b,a             ;Check LC on/off hook
        call    chk_busy        ;  if a=0-->onhook(OK) else return
        and     a
        jr      nz,disconE
        ld      a,(ix+tstr_cnt+1)   ;Get first digit
        and     a
        jp      z,disconE           ;If first digit=0 then return
        ld      a,(ix+tstr_cnt+1)   ;   else (dis_ring ,clr Ts,
        ld      b,a                 ;           clr first digit)
        dec     b                   ;** change digit to 0-7
        call    dis_ring            ;Disable Ring
        ld      a,(lc_index)
        ld      b,a
        ld      c,0ffh
        call    prg_tsac             ;Clr TS
        ld      b,(ix+tstr_cnt+1)    ;First digit  is 1-8
        dec     b                    ;Change digit to 0-7
        ld      c,0ffh
        call    prg_tsac
        xor     a                    ;Clr first digit
        ld      (ix+tstr_cnt+1),a
disconE: ret
;-------------------------------------------------------------
CONNECT_LC:  ;  Connect Line circuit
;Description :  Connect 2 Line circuit
;Parameter   :  b=lc#xx  c=lc#yy
;Return Value:  None
;Used Reg    :  bc,a
;Call Sub    :  prg_ts
;
        push    bc
        ld      c,b
        ld      a,40h
        or      c
        ld      c,a
        call    prg_tsac
        pop     bc
        push    bc
        ld      b,c
        ld      a,40h
        or      c
        ld      c,a
        call    prg_tsac
        pop     bc
        push    bc
        ld      a,80h
        or      c
        ld      c,a
        call    prg_tsac
        pop     bc
        ld      a,b
        ld      b,c
        ld      c,a
        ld      a,80h
        or      c
        ld      c,a
        call    prg_tsac
        ret
;-------------------------------------------------------------
Prg_TSAC:    ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value:  B--> 1=Sucess / 0=Timeout
;Used Reg    :  a,bc
;Call Sub    :  none
```

```
;
        push    bc
        inc     b               ;B = (0-7) ,increment b before use
stsacs: ld      a,80h
stsac1: rlca
        djnz    stsac1
        out     (PTSAC_CS),a    ;Set CS
        ld      a,c
        ld      b,08h
        rlc     c
        rlc     c
stsac2: ld      a,c
        or      0fdh            ;1111 1101
        out     (PTSAC_DI),a    ;send data to port c low
        dec     a
        out     (PTSAC_DI),a    ;send data to port c hi
        rlc     c
        djnz    stsac2

        ld      b,0
stsac3: inc     b
        jp      z,stsac_to      ;If b>256 Time out
        in      a,(PTSAC_CTS)
        and     10h             ;CTS
        jp      nz,stsac3
        pop     bc
        ld      b,1             ;Success
        jp      stsac_e
stsac_to:
        pop     bc
        ld      hl,(wr_index)
        add wr_cbuff_MOD_FAIL
        ld      a,b
        ld      (hl),a
        inc_BC
        inc_index wr_index
        set_req
        ld      b,0
stsac_e:  ld    a,0
        out     (PTSAC_CS),a
        ret
;-------------------------------------------------------------
En_Ring:      ;  Enable Ring
;Description  :  Enable ringing signal to lc
;Parameter    :  b=lc#  (0-7)
;Return Value:  NONE
;Used Reg     :  b,a,c
;Call Sub     :  none
;
        inc     b
        ld      a,80h   ;1000 00000
en_ril: rlca
        djnz    en_ril
        cpl
        ld      b,a
        ld      a,(ring_add)
        and     b
        ld      (ring_add),a
        ret
;-------------------------------------------------------------
Dis_Ring:     ;  Disable Ringing
;Description  :  Disable ringing signal of lc
;Parameter    :  b=lc#  (0-7)
;Return Value:  NONE
;Used Reg     :  b,a
;Call Sub     :  none
;
        inc     b
        ld      a,80h   ;1000 00000
di_ril: rlca
        djnz    di_ril
        ld      b,a
        ld      a,(ring_add)
        or      b
        ld      (ring_add),a
        ret

            ENDF
```

```
;-------------------------------------------------------
;   8 SSI Module
;   Name        : SSI010_2.SRC
;   Description : Service Function
;                   - NMI service . Comm. service
;                   - Command service ,Command output
;                   - Initialize
;   Version     : 10
;   Date        : 24 January 1991
;-------------------------------------------------------
Exec com:    ;Execute command
;Description : Execute MPU command
;Parameter   : command in rd cbuff
;Return Value: none
;Used Reg    :
;Call Sub    : chk dev,set tsac,set pcm,en ringlc,dis ringlc,test ring
        DI
        ld      hl,rd cbuff     ;Read byte count
        ld      a,(hl)
        cp      0               ;If byte count equal 0 then RET
        jp      z,exec2ee
        ld      b,a             ;   else do until last command
        inc     hl
        ld      (rd index),hl   ;Read index <- first command
exec21: ld      hl,(rd index)
        push    bc
        ld      a,(hl)          ;Read command ,Compare commad. call sub
        cp      chk dev         ;Check device number
        call    z,chk dev
        cp      set tsac        ;Set tsac
        call    z,set tsac
        cp      set pcm         ;Set pcm
        call    z,set pcm
        cp      en ringlc       ;Enable ring
        call    z,en ringlc
        cp      dis ringlc      ;Disable ring
        call    z,dis ringlc
        cp      test ring       ;Test ring
        call    z,test ring
        cp      ssi click       ;Click
        call    z,ssi click
        cp      set test
        call    z,set test      ;Test Module
        cp      0
        jp      z,exec22
        call    other
        pop     bc
        jp      exec2e
exec22: inc index rd index      ;Go to next command
        pop     bc              ;Decrement couunter
        dec     b
        dec     b               ;Decrement loop counter
        djnz    exec21          ;Repeat until Last command
exec2e: ld      hl,rd cbuff     ;Set rd index to first command
        ld      (rd index),hl
        ld      a,0             ;Clear Byte count
        ld      (hl),a
exec2ee: EI
        RET
;-------------------------------------------------------
Comm service:: Communnication service routine
;Description : Communication routine
;               - communicate with MPU
;Parameter   : h in,h out
;Return Value: content in C Buff
;Used Reg    : none
;Call Sub    : none
;
        exx                         ;Save old content
        ex      af,af'
        in      a,(INF_H_in)        ;Check REQ_ACK before check mode
        and     REQ_ACK             ;Mask for REQ_ACK
                                    ;  If a=0 then REQ_ACK active
        cp      00H                 ;After compare with Zero
                                    ;
        jp      nz,RD               ;If not ZERO then NO REQ_ACK ,goto rd
        ld      a,(REQ_ADD)         ;Check Request Flag
        cp      ACK                 ;   If ACK then Request to Send data
        jp      nz,NR_NW            ;If no Request then No_read,No_write
```

```
                                        ;Write command to inteface
WR:     WR_INF                          ;Command write complete
        ld      a,NAK                   ;Clear request flag
        ld      (REQ_ADD),a
        Dis_H_Out REQ                   ;Disable request signal
        En_H_Out  DATA_RDY              ;Enable data_rdy signal
        call    clr_wr_index            ;Clear wr_index ,clear byte count
        jp      comms_e
                                        ;Read command from interface
RD:     ld      hl,RD_CBUFF
        in      a,(INF_DATA)            ;Read Byte count
        ld      (hl),a
        ld      b,a
        inc     hl
RD_1:   in      a,(INF_DATA)            ;Read Command/Data
        ld      (hl),a
        inc     hl
        djnz    RD_1                    ;Repeat Until Byte count=0
RD_E:   En_H_Out  DATA_RDY              ;Enable DATA_RDY
        call    clr_rd_index            ;Clear rd_index
        jp      comms_e
                                        ;No read ,No write
NR_NW:  ld      hl,WR_CBUFF             ; This Case = REQ_ACK = active
        ADD_WR_CBUFF  3                 ;       out     NO REQUEST
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        En_H_Out   DATA_RDY
comms_e:Dis_H_Out  DATA_RDY
        ld      hl,WR_CBUFF+1
        ld      (Wr_Index),hl
        ld      hl,rd_Cbuff+1           ; Skip Byte count
        ld      (Rd_index),hl
        ex      af,af'
        exx
        ei
        RETI
;---------------------------------------------------------
TEST_MOD:     ; Test Subroutine
;Description :  Subroutine for testing S SSI module
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,ix,de
;Call Sub    :  connect,disconnect

        ld      b,8
        ld      ix,lc_table
        ld      de,table_len
maint2: ld      a,8
        sub     b
        ld      (lc_index),a
        push    bc
        push    ix
        call    connect
        pop     ix
        pop     bc
        push    bc
        push    ix
        call    disconnect
        pop     ix
        pop     bc
        add     ix,de
        djnz    maint2
        ret
;---------------------------------------------------------
NMI_Service: ;  Non Maskable Interrupt service routine
;Description :  Get lc status , analysis status,
:               and ring generation
;Parameter   :
;Return Value:  data in lc table
;Used Reg    :  all
;Call Sub    :  scan_lc,anlst,clrring,ring
;
        exx
        ex      af,af'
        push    ix
        push    iy
        call    scan_lc         ; Scan hook-status
        call    anlst           ; Analysis hook-status
        call    clrring         ; Clear ringing
```

```
        call    ring            ; Ringging generation
        call    click
        pop     iy
        pop     ix
        ex      af,af'
        exx
        ei
        retn
;---------------------------------------------------------
SCAN_LC:        ; Scan LC
;Description :  Scan input every 11 ms
;Parameter   :  none
;Return Value:  New hook status in status byte of each lc table
;Used Reg    :  a, bc, de, hl, ix
;Call Sub    :  none
;
        in      a,(Phook_status)
        out     (Phook_status),a
        cpl
        ld      (scan_add),a
        ld      hl,lc_table
        ld      IX,old_stu
        ld      b,max_lc
        ld      de,table_len
        ld      c,1
scan_lc0: ld    a,(hl)
        ld      (ix+0),a
        ld  a,(scan_add)
        and     c
        jr      z,scan_lc1
        set     0,(hl)
        jr      scan_lc2
scan_lc1: res   0,(hl)
scan_lc2: rlc   c
        add     hl,de
        inc     ix
        djnz    scan_lc0
        ret
;---------------------------------------------------------
ANLST:          ; Analysis Hook-status
;Description :  Analysis change of lc hook status
;Parameter   :  Old/New status in lc_table
;Return Value:  Changes of flag in status byte of each lc table
;Used Reg    :  max_lc-B=lc#
;Call Sub    :  onhook,offhook,flash,dial p
;
        ld      a,max_lc
        ld      b,a
        ld      ix,lc_table
        ld      DE,OLD_STU
anlst0: push    bc
        push    de
        call    Anlst1
        ld      bc,table_len
        add     ix,bc
        pop     de
        pop     bc
        inc     de
        djnz    anlst0
        ret
Anlst1: ld      a,(ix+tstatus)
        and     3
        jp      z,Anlstcase0
        dec     a
        jp      z,Anlstcase1
        dec     a
        jp      z,Anlstcase2
        jp      Anlstcase3
Anlstcase0: ;From 0 to 0
        ld      a,(ix+ttimer)
        cp      tc_on
        jr      c,anlstcase0_1
        res     2,(ix+tstatus)      ;Set on hook flag
        CALL    ONHOOK
anlstcase0_1:
        inc     (ix+ttimer)
        ret     nz
        dec     (ix+ttimer)         ;If timer overflow ,dec timer
        ret
Anlstcase1:
```

```
        ld      a,(ix+ttimer)
        cp      tc_fmin
        jr      c,anlstcase1_1
        cp      tc_fmax
        jr      nc,anlstcase1_1
        set     3,(ix+tstatus)      ;SET FLASH FLAG
        CALL    FLASH
anlstcase1_1:
        ld      a,(ix+ttimer)
        cp      tc_digit
        jr      nc,anlstcase1_2
        inc     (ix+tpulse_cnt)
anlstcase1_2:
        set     1,(ix+tstatus)
        ld      (ix+ttimer),1
        ret
Anlstcase2:
        res     1,(ix+tstatus)
        ld      (ix+ttimer),1
        ret
Anlstcase3:
        ld      a,(ix+ttimer)
        cp      tc_off
        jr      c,anlstcase3_1
        set     2,(ix+tstatus)
        CALL    OFFHOOK
anlstcase3_1:
        ld      a,(ix+ttimer)
        cp      tc_inter
        jr      nz,anlstcase3_2
        ld      a,(ix+tpulse_cnt)
        and     a
        jr      z,anlstcase3_2
        ld      c,a
        ld      a,(ix+tstr_cnt)
        cp      tstr_len
        jr      z,anlstcase3_2
        ld      hl,tstr_cnt+1       ;start of string
        push    ix
        pop     de
        add     hl,de
        ld      e,a
        ld      d,0
        add     hl,de
        ld      (hl),c
        inc     a
        ld      (ix+tstr_cnt),a
        ld      (ix+tpulse_cnt),0
        CALL    DIAL_P
anlstcase3_2:
        inc     (ix+ttimer)
        ret     nz
        dec     (ix+ttimer)
        ret
;-------------------------------------------------------------
ClrRing:      ;  Analysis Hook-status
;Description :  Analysis change of lc hook status
;               If off hook then Disable ring,clear ringing flag
;Parameter   :  Old/New status in lc table
;Return Value:  Changes of flag in status byte of each lc table
;Used Reg    :  bc,de,hl,a
;Call Sub    :
;
        ld      b,max_lc
        ld      c,01h
        ld      de,table_len
        ld      hl,lc_table
clrring1: ld    a,(hl)              ;Get hook status
        and     tsonoff             ;if not zero offhook,clear ringing
        jr      z,clrring2
        push    bc
        ld      a,max_lc
        sub     b
        ld      b,a
        call    dis_ring
        pop     bc
        ld      a,(ring_add)        ;* Set flag in ring status address
        or      c                   ;*
        ld      (ring_add),a        ;*
clrring2: add   hl,de
```

```
        rlc     c
        djnz    clrring1
        ;********************
        ld      a,(ring_add)
        ld      c,a
        ld      a,(ringing_stu)
        or      c
        out     (pring_en),a
        ;********************
        ret
;-------------------------------------------------------------
RING:       ;
:Description : Generate ringing pattern
;              Ringing signal : -___-___-___
;                ring 1s ,null 3 sec
:Parameter   :
:Return Value:
;Used Reg    :
;Call Sub    :
;
;
        ld      hl,ring_cnt1    ;Load ring_cnt address to HL
        ld      b,max_lc        ;Load lc_max to loop counter
        ld      c,01h           ;Load lc# to C
ring_s: push    bc
        ld      a,(ring_add)    ;Check ringing flag
        and     c
        jp      nz,ring_ee      ;If no ringing flag
                                ;   then go to next lc
        ld      e,(hl)          ;Load ring counter
        inc     hl
        ld      d,(hl)
        dec     de              ;Dec counter
        ld      a,d
        or      e
        jr      nz,ring_eee     ;if not zero then save counter ,go next
        ld      a,(ringing_stu) ;Check status ring or wait
        and     c
        jp      z,ring_w
ring_r: ld      a,c             ;Change to Ring status
        cpl
        ld      c,a
        ld      a,(ringing_stu)
        and     c
        ld      (ringing_stu),a ;Save new status
        ld      de,tc_ring      ;Load Ring time const to DE
        jr      ring_o
ring_w: ld      a,(ringing_stu)
        or      c
        ld      (ringing_stu),a ;Save new status
        ld      de,tc_wait      ;Load Wait time const to DE
ring_o:
        ld      c,a
        ld      a,(ring_add)
        or      c
        out     (pring_en),a    :Ringing
ring_eee:
        dec     hl              ;Save data in DE to counter
        ld      (hl),e
        inc     hl
        ld      (hl),d
        dec     hl
ring_ee:inc     hl
        inc     hl
        pop bc
        rlc c               ;Go to next LC#
        djnz    ring_s          ;Repeat until LC#>lc_max
;       ret
;****************
;   Clear ring status to 0 if ring_add=no ringing
        ld      c,01h
        ld      b,max_lc            ; b -> count
        ld      hl,ring_cnt1        ; hl-> ring_counter
clr_rstu:
        ld      a,(ring_add)        ; Get ring status
        and     c                   ; Check
        jr      z,clrst_ee          ; If ring_status= ring jmp to next
        ld      a,c
        cpl
        ld      d,a
```

```
        ld      a,(ringing_stu)     ; If ring_status=no then
                                    ;      clear ringing_status t
        and     d
        ld      (ringing_stu),a
        ld      a,tc_ring
        ld      (hl),a
        inc     hl
        ld      a,0
        ld      (hl),a
        dec     hl                  ; Go to next status
clrst_ee:inc    hl
        inc hl
        rlc c
        djnz    clr_rstu
clr_rstu_e:
        ret
;-------------------------------------------------------------
;   Ringing pattern time constant
;
ring_st equ     0
wait_st equ     1
tc_ring equ     91      ;ringing interval
tc_wait equ     272     ;silent  interval
;-------------------------------------------------------------
CLICK:      ;  CLICK
;Description :  Send a short ring
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,de,hl
;Call Sub    :  none
;
        ld      b,max_lc
        ld      hl,click_cnt        ; bit = 0->state click
        ld      c,80h                       ;1000 0000
        ld      d,0         ;mod#
click1: rlc     c
        ld      a,(click_add)
        and     c
        jp      z,click2    ;Check Click status
        ld      a,(hl)      ;Get click cnt
        cp      0           ;Check cnt=0
        jp      z,click2
        dec     a
        ld      (hl),a
        jp      nz,click2   ;counter not zero
        ld      a,c         ;1000 0000
        cpl
        ld      c,a         ;01111111
        ld      a,(click_add)
        and     c
        ld      (click_add),a

        push    bc
        ld      b,d
        call    dis_ring
        pop     bc
click2: inc     hl
        inc     d
        djnz    click1
        ret
;-------------------------------------------------------------
init_8251:
;Description :
;Parameter   :
;Return Value:
;Used Reg    :
;Call Sub    :
;
        ld      a,0ffh          ;Clear Handshake output
        out     (INF_H_OUT),a
        out     (PRING_EN),a
        ld      (H_OUT_ADD),a   ;Clear Handshake output Status flag
        ld      a,nak
        ld      (REQ_ADD),a     ;Clear Request flag
        ld      (_test_module),a;Clear test status
;
        ld      a,88h           ;Program 8255
        out     (P8255CMD),a
        ld      a,0
        out     (PTSAC_CS),a    ;Clear tsac_cs to 0
```

```
        ld      (ringing_stu),a ;Clear lc_ring status to zero
        ld      (click_add),a
        ret
;-------------------------------------------------------------
Clear_buff:
;Description :  Clear Read/Write command buffer
;Parameter   :
;Return Value:
;Used Reg    :
;Call Sub    :
;
        CLR_CBUFF  WR_CBUFF     ;Clear Write Command Buffer
        CLR_CBUFF  RD_CBUFF     ;Clear Read  Command Buffer
        call    clr_wr_index    ;Clear wr_index
        call    clr_rd_index    ;Clear rd_index
        ret

;-------------------------------------------------------------
init_RING:    ;  Initilize Ring counter
;Description :  Clear ring counter, init ring counter/status
;               for each SSI Module
;Parameter   :
;Return Value:
;Used Reg    :
;Call Sub    :
;
        ld      a,0ffh
        ld      (Ring_add),a    ;Clear Ringing address
        ld      (ringing_stu),a ;Clear Ringing status
;
        ld      b,max_lc
        ld      hl,Ring_cnt1
        ld      a,tc_ring
        ld      c,0
intr:   ld      (hl),a
        inc     hl
        ld      (hl),c
        inc     hl
        djnz    intr
        ret
;-------------------------------------------------------------
Init_lctable:; Initlize lc_table
;Description : Clear lc table
;Parameter   : none
;Return Value: new contents in lc table
;Used Reg    : a,bc,hl
;Call Sub    : none
        ld      hl,lc_table
        ld      a,0
        ld      b,table_len*max_lc
clr_tbl1: ld    (hl),a
        inc     hl
        djnz    clr_tbl1
        ld      a,0             ;clear old status table
        ld      b,max_lc
        ld      hl,old_stu
clr_tbl2: ld    (hl),a
        inc     hl
        djnz    clr_tbl2
        ret
;-------------------------------------------------------------
init_table:   ;  Initialize table
;Description :  init ts table, pcm_table
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,hl,de
;Call Sub    :  none
;
        ld      hl,ts_table_std
        ld      de,ts_table
        ld      b,0
        ld      c,max_lc
        ldir
        ld      a,(_pcm_std)
        ld      (_pcm),a
        ret
;-------------------------------------------------------------
Prg_8253:     ; Program 8253
;Description :  To generate 11 ms NMI signal, Cnt #2 is programmed
;                   to devide clock (2MHz) by 22000 .
```

```
;Parameter    :  none
;Return Value:  none
;Used Reg     :  a
;Call Sub     :  none
;
        ld      a,0b4h          ; Read/Load LSB before MSB
                                ; Rate  Generator, bin
        out     (P8253cmd),a
        ld      a,0f0h          ; Devide clock 2MHz by 22000 (55f0h)
        out     (P8253c2),a
        ld      a,55h;
        out     (P8253c2),a
        ret
;-----------------------------------------------------------
init_tsac:    ;  Initialize tsac
;Description :  Set Tx/Rx time slot to 0
;Parameter    :  none
;Return Value:  none
;Used Reg     :  hl,de,a
;Call Sub     :  none
;
        ld      a,( pcm)
        out     (P8255B),a      ;Set pcm to default value
        ld      hl,ts table
        ld      b,8
        lo      a,0
init tsac1:
        push    bc
        ld      b,a             ; b=a -> ts# (0-7)
        ld      c,(hl)          ; c -> TS#
        call    prg_tsac
        inc     a
        inc     hl
        pop     bc
        djnz    init_tsac1
        ret
        endf
```

```
;-----------------------------------------------
;   8 SSI Module
;   Name        : SSI010_3.SRC
;   Description : Service Function
;                   - Command input  service
;                   - Command output service
;   Version     : 10
;   Date        : 24 January 1991
;-----------------------------------------------
Chk_Dev:        ;Check Device Number to MFU
;Description :  Send Device Number to MFU
;Parameter   :  None
;Return Value:  Content in Wr_Cbuff
;Used Reg    :  HL,A
;Call Sub    :  none
;
        ld      hl,wr_cbuff
        ld      de,temp_cbuff
        ld      a,(hl)              ;Read byte count
        inc     hl                  ;Point to first command
        cp      0
        jp      z,chkdev1
        ld      b,0
        ld      c,a
        ldir
        ld      hl,temp_cbuff
        ld      de,wr_cbuff+1+3
        ld      b,0
        ld      c,a
        ldir
chkdev1: ld     hl,wr_cbuff+1
        Add_Wr_Cbuff    DEV_NUM
        Add_Wr_Cbuff    Null
        Add_Wr_Cbuff    Null
        inc_bc
        inc_index   wr_index
        Set_Reg
        ld      a,0
        ret
;-----------------------------------------
Set_TSAC:       ;Set Time Slot Assigner Circuit
;Description :  Set -both tx/rx timeslot  (00h)
;                   -only Tx              (40h)
;                   -Only Rx              (80h)
;Parameter   : 30h,xxh,yyh  xx=SSI
;                           yy=TS# or (TSAC Command)
;Return Value:  0=Fail  1=Sucess   0=Timeout
;Used Reg    :  a,bc,hl
;Call Sub    :  Prg_TSAC
;
        ld      hl,(Pd_index)   ;Command
        inc     hl
        ld      a,(hl)          ;SSI#
        ld      b,a             ;B <- SSI#  (0-7)
        inc     hl
        ld      a,(hl)          ;TSAC command + TS#
        ld      c,a             ;C <- TSAC com + TS#
        call    prg_tsac
;       inc_index   rd_index
        ld      a,0
        ret
;-----------------------------------------
Set_PCM:        ;Set Tx/Rx PCM line
;Description :  Set -only Tx
;                   -Only Rx
;Parameter   :  31h,xxh,yyh  xx=00000000b xx=00-> both
;                                       xx=01-> tx
;                                       xx=10-> rx
;                           yy=data 00tttrrr
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  none
;
        ld      hl,(Pd_Index)
        inc     hl
        ld      a,(hl)      ; xx000000b  xx=00->Both
                            ;            xx=01->tx
                            ;            xx=10->rx
        and     0f0h        ;If a=0 then Tx
        cp      01000000b   ;if Zero -> Tx
```

```
        jr      z,spcmtx
        cp      10000000b    ;if zero -> Rx
        jr      z,spcmrx
        ld      a,(hl)
both:   jr      spcm2
spcmtx: ld      a,(hl)       ; 00000ttth ->TX
        and     00000111b
        rlca
        rlca
        rlca
        ld      b,a          ; 00ttt000h
        ld      a,( pcm)   00xxxrrrh
        and     07h          ; 00000111h
        or      b            ; 00tttrrrh ->A
        jr      spcm2
spcmrx: ld      a,(hl)       ; 00010rrrh
        and     07h          ; 00000111h
        ld      b,a          ; 00000rrrh
        ld      a,( pcm) ; 00ttxxxh
        and     00111000b    ; 00111000h
        or      b            ; 00ttrrrrh
spcm2:  ld      ( pcm),a
        out     (P82558),a
;       inc index   rd index
        ld      a,0
        ret
;-----------------------------------------------
EN RingLC:    ; Enable ring lc
;Description : Enable ringing for each lc
;               command sent from MPU
;Parameter   : None
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    :
;
        ld      hl,(Rd_index)    ;Command
        inc     hl
        ld      a,(hl)           ;SSI#
        ld      b,a              ;B <- SSI#
        call    en_ring
;       inc index   rd index
        ld      a,0
        ret
;-----------------------------------------------
Dis_RingLC:   ; Disable ring lc
;Description : Disable ringing for each lc
;               command sent  from MPU
;Parameter   : None
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    :
;
        ld      hl,(Rd_index)    ;Command
        inc     hl
        ld      a,(hl)           ;SSI#
        ld      b,a              ;B <- SSI#
        call    dis_ring
;       inc index   rd index
        ld      a,0
        ret
;-----------------------------------------------
Test_ring:    ; ringing test
;Description :  send ringing to each lc
;Parameter   :  none
;Return Value:  none
;Used Reg    :  bc,e
;Call Sub    :  en_ring,dis_ring
;
        ld      a,(ring_add)
        push    af
        ld      a,(ringing_stu)
        push    af
        ld      c,2              ; 2 cycle
testlc0:ld      b,0
testlc1:push    bc
        call    en_ring
        call    delay
        pop     bc
        push    bc
        call    dis_ring
```

```
        call    delay
        pop     bc
        inc     b
        ld      a,b
        cp      8
        jp      nz,testlc1
        dec     c
        jp      nz,testlc0
        pop     af
        ld      (ringing stu),a
        pop     af
        ld      (ring add),a
        ld      a,0
        ret
delay:  ld      bc,0501h
delay0: push    bc
        ld      bc,0055h
delay1: djnz    $
        dec     c
        jr      nz,delay1
        pop     bc
        djnz    delay0
        dec     c
        jr      nz,delay0
        ret
;-----------------------------------------
Set_Test:     ; Set test
;Description : Set test
;Parameter    : None
;Return Value: none
;Used Reg     :
;Call Sub     :
;
        ld      a,ack
        ld      ( test module),a
        ld      a,0
        ret
;-----------------------------------------
OffHook:      ; Off Hook
;Description :  Report Off Hook to MPU
;Parameter    :  33h,xxh,00h  xx=SSI Module#
;Return Value:  Content in Wr Cbuff.Hook Add
;Used Reg     :  a,bc,d,hl
;Call Sub     :  none
;
        LD      A,(DE)
        AND     00000100B
        JR      NZ,OFFHOOK_E
        LD      HL,(WR_INDEX)
        ADD WR CBUFF     OFF HOOK
        LD      A,MAX_LC
        SUB     B
        LD      (HL),A
        INC     HL
        ADD WR CBUFF    NULL
        INC DE
        inc index   wr index
        set req                         ;Set Request flag
offhook_e:
        ret
;-----------------------------------------
OnHook:   ; On Hook
;Description :  Report On Hook to MPU
;Parameter    :  33h,xxh,00h  xx= SSI Module#
;             :  Max lc-B = lc#
;Return Value:  Content in Wr Cbuff.Hook Add
;Used Reg     :  a,bc,d,hl
;Call Sub     :  none
;
        LD      A,(DE)
        AND     00000100B
        JR      Z,ONHOOK_E
        LD      (IX+TSTR_CNT),0
        LD      HL,(WR_INDEX)
        ADD WR CBUFF     ON HOOK
        LD      A,MAX_LC
        SUB     B
        LD      (HL),A
        INC     HL
        ADD WR CBUFF    NULL
```

```
        INC BC
        Inc index Wr index
        set req                 ;Set Request flag
ONHOOK_E:
        ret
;-----------------------------------------
Flash:          ; Off Hook
;Description :  Report Flash Status to MPU
;Parameter   :  34h,xxh,00h  xx= SSI Module#
;               Max lc-B= lc#
;Return Value:  Content in Wr Cbuff,Hook Add
;Used Reg    :
;Call Sub    :  none
;
        LD      HL,(WR INDEX)
        ADD WR CBUFF    FLASH
        LD      A,MAX LC
        SUB     B
        LD      (HL),A
        INC     HL
        ADD WR CBUFF    NULL
        INC BC
        inc index   wr index
        set req                 ;Set Request flag
        ret
;-----------------------------------------
Dial P:         ; Off Hook
;Description :  Report Dial Number to MPU
;Parameter   :  35h,xxh,nn  xx= SSI Module#
;                           nn= Dial Number
;Return Value:  Content in Wr Cbuff,Hook Add
;Used Reg    :  a,bc,d,hl
;Call Sub    :  none
;
        LD      HL,(WR INDEX)
        ADD WR CBUFF    DIAL P
        LD      A,MAX LC
        SUB     B
        LD      (HL),A                  ;ssi module#
        inc     hl
        ld      a,c
        ld      (hl),a
        INC BC
        inc index   wr index
        set req                 ;Set Request flag
        ret
;-----------------------------------------------------------
SSI_CLICK:      ; SSI CLICK
;Description :  Send a short ring
;Parameter   :
;Return Value:
;Used Reg    :
;Call Sub    :
;
        ld      hl,(Rd index)   ;Command
        inc     hl
        ld      a,(hl)          ;SSI#
        ld      b,a             ;B <- SSI#
        inc     b
        ld      de,click cnt-1
        ld      c,80h
ssicl1: rlc     c
        inc     de
        djnz    ssicl1
        ld      a,(click add)
        or      c
        ld      (click add),a
        ld      b,(hl)
        call    en ring
        ld      a,(tc click)
        ld      (de),a
        ld      a,0
        ret

;-----------------------------------------
OTHER:                  ;Send back last command
          push  bc
          ld    c,max command
          ld    b,0
          LD    HL,RD CBUFF       ;(hl) ->Source
```

```
LD      DE,WR_CE00FF        ;(de) - Destination
LDIR                        ;(bc) - Number
set_req                     ;Set Request flag
pop     bc
RET
```

```
            list    a.e
;-------------------------------------------------------------
;   COI-Central Office Interfce Module
;   Name        : COI003.SRC
;   Description : COI-CO module Control Program
;                 Main Program / Address-Data Set
;   Version     : 1.0
;   Date        : 3  Dec 1990
;  Last Edit    : 24 January 1991
;-------------------------------------------------------------
ROM             equ 0           ;Rom start address
RAM             equ 2000h       ;Ram start address
;
P8255           equ 90h         ;Port 8255
P8253           EQU 80h
;
P8253C0         EQU P8253
P8253C1         EQU P8253+1
P8253C2         EQU P8253+2
P8253CMD        EQU P8253+3
;
P8255A          equ P8255
P8255B          equ P8255+1
P8255C          equ P8255+2
P8255CMD        equ P8255+3
;
PTSAC_CS        equ P8255A      ;Port Chip select
PTSAC_DI        equ P8255C      ;Port Data input
PTSAC_CTS       equ P8255C      ;Port Clear to send
COI_RING        equ 0fh         ;Ring Input port
COI_LED         equ 0fh         ;LED Diplay
COI_HOOK        equ 0eh         ;Hook Port
;
MAX_COMMAND     equ 240         ;Maximum command
max_coi         equ 8           ;(256=255+1 ->0)
tc_wait             equ 400
;
_COI_Dev_Num    equ 04h         ;TSI device number
_Dev_Num        equ _COI_Dev_Num
_Chk_Dev        equ 20h         ;Check device number command
_Set_TSAC       equ 30h         ;Set TSAC
_Set_PCM        equ 31h         ;Set PCM
_CO_ONHOOK      equ 41h
_CO_OFFHOOK     equ 42h
_CO_FLASH       equ 43h
_CO_DIAL        equ 44h
_CO_RING        equ 45h
_DIAL_RDY       equ 46h
_CO_RCNL        equ 47h
_MOD_FAIL       equ 26h         ;Module #xx absent
;-------------------------------------------------------------
;   Inteface to MPU
;-------------------------------------------------------------
INF_H_OUT       equ 0AH         ;MPU Inteface Handshake output
INF_H_IN        equ 09H         ;MPU Inteface Handshake input
INF_DATA        equ 08H         ;MPU Inteface Data
;-------------------------------------------------------------
;   I/O HANDSHAKE SIGNAL
;   <INPUT>
;-------------------------------------------------------------
;                   VALUE
PM_EN           equ 02H         ;PM enable          BIT 1
REQ_ACK         equ 04H         ;Request acknowledge BIT 2
RST_PM          equ 08H         ;Reset PM           BIT 3
;   <OUTPUT>        BIT#
NAK_ACK         equ 0           ;Nak-Ack ->cata correct;      BIT 0
DATA_RDY        equ 1           ;Data ready                   BIT 1
REQ             equ 2           ;Request from PM              BIT 2
RST_CNT         equ 3           ;Reset inteface address counter BIT 3
;-------------------------------------------------------------
;   COI STATE
;
ONHOOK_ST       equ     0
RINGDET_ST      equ     1
OFFHOOK_ST      equ     2
FLASH_ST        equ     3
```

```
DIALON_ST       equ     4
DIALOFF_ST      equ     5
DINTER_ST       equ     6
;-------------------------------------------------------
;   ASCII Control code
;
NULL            equ 00H         ;NULL
ACK             equ 06H         ;Ack
NAK             equ 15H         ;Nak
;-------------------------------------------------------
;
        FILE    coi003_0.src     ;BIOS MACRO FILE
;
        org     RAM
        ld      sp,stk_pt       ;Set stack pointer
        di
        im      1
        ld      b,0
        djnz    $
        jp      MAIN
;
        int_service             ;Interruot service macro
;
        org     100h
;-------------------------------------------------------
MAIN:           : MAIN
:Description : Main Program
:Parameter   : none
:Return Value: none
:Used Reg    : none
;Call Sub    :
;
        call    init_hs         ;Initialize handshake
        call    clear_buff      ;Clear command buffer
        call    init_cnt        ;initialize COI counter
        call    init_stu        ;initialize COI initial status
        call    init_8255       ;initialize 8255
        call    init_table
        call    init_tsacpcm    ;initialize TSAC/PCM
        call    init_8253
main1:
        halt
        call    exec_com        ;Execute command send from MPU
        jr      main1
;-------------------------------------------------------
;Bios file
        file    coi003_1.src
;Service Function file
        file    coi003_2.src
;Service Function file
        file    coi003_3.src
;-------------------------------------------------------
;   Time Contant
;
tc_don:         defb    7
tc_doff:        defb    4
tc_flash:       defb    30
tc_inter:       defb    20
;-------------------------------------------------------
;   Constant value
;   pcm, ts, tone ts table
;
pcm_std:    defb    00001001b   ;PCM#7
;
ts_std:     defb    0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh
;-------------------------------------------------------
;   Define value
;
        org     RAM
COI_CNT:    defs    max_coi
RING_CNT:   defs    2*max_coi
COI_ST:     defs    max_coi
COI_NUM:    defs    max_coi
RSTU        defs    max_coi
ring_stu:   defs    1
HOOK_STU:   defs    1
H_IN_ADD:   defs    1
H_OUT_ADD:  defs    1
MUX_ADD:    defs    2
REQ_ADD:    defs    1
```

```
WR_Index:   defs    2
Rd_Index:   defs    2
;-------------------------------------------------------------
_pcm        defs    1       ;pcm line address
ts_table:   defs    8
;-------------------------------------------------------------
;   Command Buffer
;
            org     2500h
RD_CBUFF:   defs    max_command
            org     2500h
WR_CBUFF:   defs    max_command
            org     2700h
temp_cbuff  defs    max_command
;-------------------------------------------------------------
;   Timeslot interchange location
;
            org     ram+1fffh
stk_pt:     defs    1
        end
        end
```

```
;-------------------------------------------------------
;   COI-Central Office Interface Module
;   Name        : COICO3_0.SRC
;   Description : Bios Macro
;                 - Macro initialize interrupt service routine
;                 - Macro support communicate with MPU
;   Version     : 1.0
;   Date        :  3 December 1990
;   Last Edit   : 24 January  1991
;-------------------------------------------------------
Int_service    Macro
;Description : Interrupt service routine
;            :  - Service all interrupt
;Parameter   : none
;Return Value: none
;Used Reg    : all
;Call Sub    : none
;
RST28:  org     28H
        reti
RST30:  org     30H
        reti
RST38:  org     38H             ; Service communication
        di
        jp      Comm_service
        reti
NMI:    org     66H             ; Scan & analysis SSI status
        jp      NMI_Service
        retn
        endM
;---------------------------------------
Reset_Cnt     Macro
;Description : Reset counter in MPU-PPU interface
;Parameter   : none
;Return Value: none
;Used Reg    : hl,b,a
;Call Sub    : none

        ld      hl,H_out_ADD
        res     RST_CNT,(hl)
        out     (INF_H_out),a
        ld      b,0AH           ;20 MICROSEC
        djnz    $
        set     RST_CNT,a
        out     (INF_H_out),a
        ld      (hl),a
        endM

;---------------------------------------
Dis_H_out     Macro  #H_out
;Description : Disable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none
;
        ld      hl,H_out_ADD
        set     #H_out,(hl)
        ld       a,(hl)
        out     (INF_H_out),a
        endM
;---------------------------------------
En_H_out     Macro  #H_out
;Description : Enable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none

        ld       hl,H_out_ADD
        res      #H_out,(hl)
        ld       a,(hl)
        out      (INF_H_out),a
        endM
;---------------------------------------
Wr_Inf         Macro
;Description : Copy C_Buff from PPU to MPU
;Parameter   : none
;Return Value: none
;Used Reg    : a,b,hl
```

```
;Call Sub    :
        LOCAL #WR_INF_1
        ld      hl,WR_CBUFF       ;Byte_count
        ld      b,(hl)
        inc     b                 ; add one byte for Byte count
#WR_INF_1: ld   a,(hl)
        out     (INF_DATA),a
        inc     hl
        djnz    #WR_INF_1
        endM
;-----------------------------------------
Add_Wr_Cbuff  Macro #data
;Description : Add a data to wr_cbuff
;Parameter   : #data
;Return Value: none
;Used Reg    : a,hl
;Call Sub    : none
        ld      a,#DATA
        ld      (hl),a
        inc     hl
        endM
;-----------------------------------------
Clr_Cbuff       Macro #Cbuff
;Description : Clear Command Suffer
;Parameter   : #cbuff
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    : none
        LOCAL   #L1
        ld      hl,#CBUFF
        ld      b,MAX_COMMAND+1
        xor     a
#L1:    ld      (hl),a
        inc     hl
        djnz    #L1
        endM
;-----------------------------------------
Set_Req         Macro
;Description : Activate Request signal
;Parameter   : none
;Return Value: none
;Used Reg    : a
;Call Sub    : En_h_out
        ld        a,ACK
        ld        (Req_add),a
        En_H_out REQ
        endM
;-----------------------------------------
Inc_index       Macro #index
;Description : Increment rd_index or Wr_index
;Parameter   : #index
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        push    hl
        ld      hl,(#index)
        inc     hl
        inc     hl
        inc     hl
        ld      (#index),hl
        pop     hl
        endM
;-----------------------------------------
Inc_BC          Macro
;Description : Increment Byte count
;Parameter   : none
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        ld      a,(wr_cbuff)
        add     a,3
        ld      (wr_cbuff),a
        endM
;-----------------------------------------------------------
GEN_TONE    Macro   #tone,#tcon,#tcoff
;Description :  Genreate Tone
;Parameter   :  #tone, time interval on/off
;Return Value:  none
;Used Reg    :  a,hl,de,ix
;Call Sub    :  none
```

```
;
        local   #gentt,#gentw,#gente
        ld      a,(hl)          ;load status to a
        and     80h             ;Mask 1000 0000b ,if zero->wait
                                ;                else ->tone
        jr      z,#gentw
#gentt: ld      a,(#tcoff)      ;load rot-off tc
        ld      (de),a
        ld      a,(_tsint)
        ld      (ix),a          ;disable tone
        res     7,(hl)          ;set status to wait
        jr      #gente
#gentw: ld      a,(#tcon)       ;load rot-on tc
        ld      (de),a
        ld      a,(#tone)
        ld      (ix),a          ;enable tone
        set     7,(hl)          ;set status to tone
#gente: ld      a,0
        endM
;-----------------------------------------
hookst_on    Macro
;Description : Set address Hook_stu to on
;Parameter   : c=COI Mask
;Return Value: none
;Used Reg    : a,c
;Call Sub    : none
        push    bc
        ld      a,c
        cpl
        ld      c,a
        ld      a,(hook_tu)
        and     c
        ld      (hook_stu),a
        pop     bc
        endM
;-----------------------------------------
hookst_off   Macro
;Description : Set address Hook_stu to on
;Parameter   : c=COI Mask
;Return Value: none
;Used Reg    : a,c
;Call Sub    : none
        ld      a,(hook_stu)
        or      c
        ld      (hook_stu),a
        endM
        endf
```

```
;-------------------------------------------------------
;   COI-Central Office Interface Module
;   Name        : COI003_1.SRC
;   Description : BIOS, Service Function
;                 - Program tsac
;                 - Clear command index
;   Version     : 1.0
;   Date        : 3 Dec  1990
;   Last Edit   : 24 January 1991
;-------------------------------------------------------
Prg_TSAC:    ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: B-> 1=Sucess / 0=Timeout
;Used Reg    : a,bc
;Call Sub    : none
;
        push    bc
        inc     b               ;B = (0-7) ,increment b before use
sstsacs:    ld      a,80h
sstsac1: rlca
        djnz    sstsac1
        out     (PTSAC_CS),a    ;Set CS
        ld      a,c
        ld      b,08h
        rlc     c
        rlc     c
sstsac2:    ld      a,c
        or      0fdh            ;1111 1101
        out     (PTSAC_DI),a    ;send data to port c low
        dec     a
        nop
        nop
        out     (PTSAC_DI),a    ;send data to port c hi
        rlc     c
        djnz    sstsac2
        ld      b,0
sstsac3: dec        b
        jp      z,sstsac_to     ;If b=256 Time out
        in      a,(PTSAC_CTS)
        bit     4,a             ;CTS
        jp      nz,sstsac3
        pop     bc
        ld      b,1             ;Success
        jp      sstsac_e
sstsac_to:
        ld      hl,(wr_index)
        add_wr_cbuff_MOD_FAIL
        pop     bc
        ld      a,b
        ld      (hl),a
        inc_BC
        inc_index_wr_index
        set_req
        ld      b,0
sstsac_e:  ld   a,0
        out     (PTSAC_CS),a
        ret
;-------------------------------------------------------
;Prg_TSAC:    ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: B-> 1=Sucess / 0=Timeout
;Used Reg    : a,bc
;Call Sub    : none
;
;       inc     b               ;B = (0-7) ,increment b before use
;stsacs:    ld      a,80h
;stsac1: rlca
;       djnz    stsac1
;       out     (PTSAC_CS),a    ;Set CS
;       ld      a,c
;       ld      b,08h
;       rlc     c
;       rlc     c
;stsac2:    ld      a,c
;       or      0fdh            ;1111 1101
;       out     (PTSAC_DI),a    ;send data to port c low
;       dec     a
;       out     (PTSAC_DI),a    ;send data to port c hi
```

```
;       rlc     c
;       djnz    stsac2
;       ld      b,0
;stsac3: inc       b
;       jp      z,stsac_to      ;If b>256 Time out
;       in      a,(PTSAC_CTS)
;       and     10h             ;CTS
;       jp      nz,stsac3
;       ld      b,1             ;Success
;       jp      stsac_e
;stsac_to: ld    b,0
;stsac_e:  ld    a,0
;       out     (PTSAC_CS),a
;       ret
;-------------------------------------------------------------
Clr_wr_index:;  Clear write Index
;Description :  Clear write Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,hl
;Call Sub    :  none
;
        ld      hl,wr_cbuff+1
        ld      (Wr_index),hl
        ld      a,0
        ld      (wr_cbuff),a
        ret
;-------------------------------------------------------------
Clr_rd_index:;  Clear read Index
;Description :  Clear read Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl
;Call Sub    :  none
;
        ld      hl,rd_cbuff+1
        ld      (Rd_index),hl
        ret
        ENDF
```

```
;-------------------------------------------------------
;   COI-Central Office Interface Module
;   Name        : COI003_2.SRC
;   Description : Service function
;                   - Execute command
;                   - NMI Service
;                   - Communication service
;                   - Initialize
;   Version     : 1.0
;   Date        :  3  Dec  1990
;   Last Edit   : 24 January 1991
;-------------------------------------------------------
Exec_com:    ;  Exec command
;Description :  Execute command which sent from MPU
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  Chk_dev,Clr_tsi,C_tone,Ex_ts,Test
;
        di
        ld      hl,rd_cbuff     ;Read byte count
        ld      a,(hl)
        cp      0               ;If byte count equal 0 then RET
        jp      z,execee
        ld      b,a             ;   else do until last command
        inc     hl
        ld      (rd_index),hl   ;Read index -- first command
exec1:  ld      hl,(rd_index)
        push    bc
        ld      a,(hl)          ;Read command
        cp      chk_dev
        call    z,CHK_DEV
        cp      set_tsac
        call    z,set_tsac
        cp      set_pcm
        call    z,set_pcm
        cp      CO_ONHOOK       ;Compare commad
        call    z,CO_ONHOOK     ; Jump if correct else skip
        cp      CO_OFFHOOK
        call    z,CO_OFFHOOK
        cp      CO_FLASH
        call    z,CO_FLASH
        cp      CO_DIAL
        call    z,CO_DIAL
        cp      0
        jp      z,exec2
        call    other
        pop     bc
        jp      exece
exec2:  inc_index   rd_index    ;Go to next command
        pop     bc
        dec     b
        dec     b               ;Decrement loop counter
        djnz    exec1            ;Repeat until Last command
exece:  ld      hl,rd_cbuff     ;Set rd_index to first command
        ld      (rd_index),hl
        ld      a,0
        ld      (hl),a
execee: EI
        ret
;-------------------------------------------------------
NMI_Service: ;  Non maskable Interrupt service routine
;Description :  Increment tone counter - Generate all tones
;Parameter   :  none
;Return Value:  none
;Used Reg    :  all
;Call Sub    :  inc_cnt2, anl_tone
;
        exx
        ex      af,af'
        push    ix
        push    iy
        ld      a,(hook_stu)
        out     (COI_HOOK),a
        cpl
        out     (COI_LED),a
        call    inc_cnt
        call    scan_ring
        call    anl_ring
        call    anl_st
```

```
        pop     iy
        pop     ix
        ex      af,af'
        exx
        ei
        retn
;-------------------------------------------------------
Comm_service:: Communnication service routine
;Description : Communication routine
;               - communicate with MPU
;Parameter    : h_in,h_out
;Return Value: content in C_Buff
;Used Reg     : none
;Call Sub     : none
;
        exx                             ;Save old content
        ex      af,af'
        in      a,(INF_H_in)            ;Check REQ_ACK before check mode
        and     REQ_ACK                 ;Mask for REQ_ACK
                                        ;  If a=0 then REQ_ACK active
        cp      00H                     ;After compare with Zero
                                        ;
        jp      nz,RD                   ;If not ZERO then NO REQ_ACK ,goto rd
        ld      a,(REQ_ADD)             ;Check Request flag
        cp      ACK                     ;   If ACK then Request to Send data
        jp      nz,NR_NW                ;If no Request then No_read,No_write
                                        ;Write command to inteface
WR:     WR_INF                          ;Command write complete
        ld      a,NAK                   ;Clear request flag
        ld      (REQ_ADD),a
        Dis_H_Out_REQ                   ;Disable request signal
        En_H_Out  DATA_RDY              ;Enable data rdy signal
        call    clr_wr_index            ;Clear wr_index ,clear byte count
        jp      comms_e
                                        ;Read command from interface
RD:     ld      hl,RD_CBUFF
        in      a,(INF_DATA)            ;Read Byte count
        ld      (hl),a
        ld      b,a
        inc     hl
RD_1:   in      a,(INF_DATA)            ;Read Command/Data
        ld      (hl),a
        inc     hl
        djnz    RD_1                    ;Repeat Until Byte count=0
RD_E:   En_H_Out  DATA_RDY              ;Enable DATA_RDY
        call    clr_rd_index            ;Clear rd_index
        jp      comms_e
                                        ;No read ,No write
NR_NW:  ld      hl,WR_CBUFF             ; This Case -> REQ_ACK = active
        ADD_WR_CBUFF  3                 ;       but    NO REQUEST
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        En_H_Out   DATA_RDY
comms_e:Dis_H_Out  DATA_RDY
        ld      hl,WR_CBUFF+1
        ld      (Wr_Index),hl
        ld      hl,rd_Cbuff+1           ; Skip Byte count
        ld      (Rd_index),hl
        ex      af,af'
        exx
        ei
        RETI
;-------------------------------------------------------
Inc_Cnt:      :  Increment counter
;Description :  Increment COI Counter
;Parameter    :  none
;Return Value:  none
;Used Reg     :  a,bc,hl,de
;Call Sub     :  none
;
        ld      de,coi_cnt
        ld      b,max_coi
inccnt1: ld     a,(de)
        and     a
        jp      z,inccnt2
        dec     a
        ld      (de),a
inccnt2:inc     de
        djnz    inccnt1
```

```
        ret
;
        jp      $
;-----------------------------------------------------------
SCAN_RING:    ; Scan ringing
;Description :  Scan Ringing from Input Port
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,b,de,hl,ix
;Call Sub    :
;
        in      a,(COI_RING)
        cpl
        ld      (ring_stu),a
        ld      hl,RSTU
        ld      b,max_coi
scr1:   rrca
        rl      (hl)
        inc     hl
        djnz    scr1
        ret
;-----------------------------------------------------------
ANL_RING:     ; Analyze COI RING
;Description :  Analyze COI Ring Absent/Present
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,de,hl
;Call Sub    :  none
;
        ld      DE,RSTU
        ld      HL,RING_CNT
        ld      ix,COI_ST
        ld      b,max_coi
Anlr1:  ld      a,(DE)
        and     00000011b
        jp      z,Anlr2
        cp      01h
        jp      z,Anlr3
        cp      02h
        jp      nz,ANLRE
        push    DE
        ld      DE,tc_wait
        ld      (HL),e
        inc     HL
        ld      (HL),d
        dec     HL
        pop     DE
        jp      ANLRE
ANLR2:  ld      a,(ix+0)
        cp      RingDet_st
        jp      nz,ANLRE
        push    DE
        ld      e,(HL)
        inc     HL
        ld      d,(HL)
        dec     HL
        dec     DE                  ;Dec Ring cnt
        ld      (HL),e
        inc     HL
        ld      (HL),d
        dec     HL
        ld      a,d
        or      e
        jp      nz,Anlr5
        pop     DE
        ld      a,ONHOOK_ST
        ld      (ix+0),a
        push    HL
        call    RING_CNL
        pop     HL
        jp      ANLRE
Anlr5:  pop     DE
        jp      ANLRE
Anlr3:  ld      a,(ix+0)
        cp      RingDet_ST
        jp      z,ANLRE
        ld      a,RINGDET_ST
        ld      (ix+0),a
        push    HL
        Call    Ringing
```

```
        pop     HL
        jp      ANLRE
ANLRE:  inc     DE
        inc     HL
        inc     HL
        inc     ix
        djnz    ANLR1
        RET
;-------------------------------------------------------------
ANL_ST:         ; Analyze COI State
;Description :  Analyze COI State
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,de,hl
;Call Sub    :  DFSUB,DONSUB,DOFFSUB,DInSUB
;
        ld      hl,COI_ST
        ld      de,COI_CNT
                ld              ix,COI_NUM
        ld      b,max_coi
        ld      c,10000000b
anl1:   ric     c
        ld      a,(de)
        and     a
        jp      nz,anle
        ld      a,(hl)
        cp      ONHOOK_ST
        jp      z,ANLE
        cp      FLASH_ST
        jp      z,DFSUB
        cp      DIALON_ST
        jp      z,DONSUB
        cp      DIALOFF_ST
        jp      z,DOFFSUB
        cp      DInter_ST
        jp      z,DInSUB
        jp      anle
ANLE:   inc     de
        inc     hl
        inc             ix
        djnz    anl1
        RET
;-------------------------------------------------------------
DFSUB:          ; Dial Flash Subroutine
;Description :  Generate  Flash hook
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,c,hl
;Call Sub    :  none
;
        ld      (hl),OFFHOOK_ST
        ld      a,(hook_stu)            ;Set hook stu to OFFHOOK
        or      c
        ld      (hook_stu),a
        ld      a,0ffh
        jp      ANLE
;-------------------------------------------------------------
DOFFSUB:        ; Dial-Off Subroutine
;Description :  Generate  Dial Pulse (Offhook)
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,c,hl
;Call Sub    :  none
;
        ld              a,(ix+0)
        dec     a
        ld              (ix+0),a
        and     a
        jp      z,doff1
        ld      (hl),DIALON_ST
        ld      a,(tc_don)
        ld      (de),a
        push    bc                      ;Set hook stu to ONHOOK
        ld      a,c
        cpl
        ld      c,a
        ld      a,(hook_stu)
        and     c
        ld      (hook_stu),a
        pop     bc
```

```
        jp      doffe
doff1:  ld      (hl),DInter st
        ld      a,(tc inter)
        ld      (de),a
        ld      a,(hook stu)        ;Set hook stu to OFFHOOK
        or      c
        ld      (hook stu),a
doffe:  ld      a,0ffh
        jp      ANLE
;-----------------------------------------------------------
DONSUB:         ; Dial-ON Subroutine
;Description : Generate  Dial Pulse (Onhook)
;Parameter   : none
;Return Value: none
;Used Reg    : a,c,hl
;Call Sub    : none
;
        ld      (hl),DIALOFF ST
        ld      a,(tc doff)
        ld      (de),a
        ld      a,(hook stu)        ;Set hook stu to OFFHOOK
        or      c
        ld      (hook stu),a
        ld      a,0ffh
        jp      ANLE
;-----------------------------------------------------------
DINSUB:         ; Dial-Interdigit Subroutine
;Description : Generate  Dial Pulse (Interdigit)
;Parameter   : none
;Return Value: none
;Used Reg    : a,c,hl
;Call Sub    : none
;
        ld      (hl),OFFHOOK ST
        ld      a,(hook stu)        ;Set hook stu to OFFHOOK
        or      c
        ld      (hook stu),a
        push    hl
        call    Dial Ready
        pop     hl
        ld      a,0ffh
        jp      ANLE
        jp      $
;-----------------------------------------------------------
Init_8255:      ; init 8255
;Description :  Set up 8255
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,88h           ;Program 8255
        out     (P8255CMD),a
        out     (P8255CMD),a
        ret
;-----------------------------------------------------------
init_table:
;Description :
;Parameter   :
;Return Value:
;Used Reg    :
;Call Sub    :
;
        ld      hl,ts std
        ld      de,ts table
        ld      b,0
        ld      c,max col
        ldir
        ld      a,(pcm std)
        ld      (_pcm),a
        out     (P8255b),a      ;Set pcm line  to default
        out     (P8255b),a
        ret
;-----------------------------------------------------------
init_tsacpcm:   ;  Initialize tsac
;Description :  Set Tx/Rx time slot to 0
;Parameter   :  NONE
;Return Value:  NONE
;Used Reg    :  hl,de,a
;Call Sub    :  none
```

```
;
        ld      a,(_pcm)
        out     (PB2558),a      ;Set pcm to default value
        ld      hl,ts_table
        ld      b,max_coi
        ld      a,0
init_tsac1:
        push    bc
        ld      b,a             ; b=a -> loop (0-7)
        ld      c,(hl)          ; c -> TS#
        call    prg_tsac
        inc     a
        inc     hl
        pop     bc
        djnz    init_tsac1
        ret
;-------------------------------------------------------------------
init_cnt:       ;Initialize Counter
;Description :  Clear tone counter
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,b,hl
;Call Sub    :  none
;
        ld      hl,coi_cnt
        ld      b,1
        ld      a,max_coi
initcnt1: ld    (hl),a
        inc     hl
        djnz    initcnt1
        ld      b,max_coi*2
        ld      hl,ring_cnt
        ld      a,0
initcnt2: ld    (hl),a
        inc     hl
        djnz    initcnt2
        ret
;-------------------------------------------------------------------
init_stu:       ;Initialize status
;Description :  Set up tone status to null tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,a,b
;Call Sub    :  none
;
        ld      a,0
        ld      (ring_stu),a
        ld      (hook_stu),a
        ld      hl,COI_ST
        ld      de,RST_
        ld      a,COI_NUM
        ld      a,(HOOK_ST
        ld      b,max_coi
initstu1:
        ld      (hl),a
        ld      (de),a
        ld      (ix+0),a
        inc     hl
        inc     de
                inc             ix
        djnz    initstu1
        ret
;-------------------------------------------------------------------
init_hs:        ;Initialize Handshake
;Description :  Initialize Handshake
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :
;
        ld      a,0ffh          ;Clear Handshake output
        out     (INF_H_OUT),a
        ld      (H_OUT_ADD),a   ;Clear Handshake output Status flag
        ld      a,msk
        ld      (REQ_ADD),a     ;Clear Request flag
        ld      a,0
        out     (COI_HOOK),a
        cpl
        out     (COI_LED),a
        ret
```

```
;-------------------------------------------------------------
clear_buff:   ;  Clear buffer
;Description :  Clear wr_cbuff, rd_cbuff
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  macro clear_buff, clr_rd_index, clr_wr_index
;
        clr_cbuff   rd_cbuff
        clr_cbuff   wr_cbuff
        call    clr_rd_index
        call    clr_wr_index
        ret
;-------------------------------------------------------------
init_8253:    ; Program 8253
;Description :  To generate 11 ms NMI signal, Cnt #2 is programmed
;               to devide clock (2MHz) by 22000 .
;Parameter   :  NONE
;Return Value:  NONE
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,0b4h          ; Read/Load LSB before MSB
                                ; Rate  Generator , bin
        out     (P8253cmd),a
        ld      a,0F0h          ; Devide clock 2MHz by 22000 (55F0h)
        out     (P8253c2),a
        ld      a,55h;
        out     (P8253c2),a
        ret
        endf
```

```
;-----------------------------------------------------------
;   COI-Central Office Interface module
;   Name        : COI003 3.SRC
;   Description : Service Function
;                   - clr tsi, c tone, ex ts, test,
;                   - chk dev, other
;   Version     : 1.0
;   Date        :
;   Last Edit   : 24 January 1991
;-----------------------------------------------------------
Chk_Dev:        ;Check Device Number to mPU
;Description :  Send Device Number to MPU
;Parameter   :  None
;Return Value:  Content in Wr Cbuff
;Used Reg    :  HL,A
;Call Sub    :  none
;
        ld      hl,wr_cbuff
        ld      de,temp_cbuff
        ld      a,(hl)              ;Read byte count
        inc     hl                  ;Point to first command
        cp      0
        jp      z,chkdev1
        ld      b,0
        ld      c,a
        ldir
        ld      hl,temp_cbuff
        ld      de,wr_cbuff+1+3
        ld      b,0
        ld      c,a
        ldir
chkdev1: ld     hl,wr_cbuff+1
        Add_Wr_Cbuff    DEV_NUM
        Add_Wr_Cbuff    Null
        Add_Wr_Cbuff    Null
        inc_bc
        inc_index   wr_index
        Set_Req
        ld      a,0
        ret
;-----------------------------------------------------------
Ringing:        ; Ringing
;Description :  Send  CO_RING to mPU
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,a,b
;Call Sub    :  add_wr_cbuff,inc_index,inc_bc
;
        ld      hl,(wr_index)
        add_wr_cbuff  CO_RING
        ld      a,max_coi
        sub     b
        ld      (hl),a
        inc     hl
        add_wr_cbuff null
        INC_BC
        INC_INDEX wr_index
        set_req
        ret
;-----------------------------------------------------------
Ring_CNL:       ; Ring Cancel
;Description :  Send  CO_RCNL to mPU
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,a,b
;Call Sub    :  add_wr_cbuff,inc_index,inc_bc
;
        ld      hl,(wr_index)
        add_wr_cbuff  CO_RCNL
        ld      a,max_coi
        sub     b
        ld      (hl),a
        inc     hl
        add_wr_cbuff null
        INC_BC
        INC_INDEX wr_index
        set_req
        ret
;-----------------------------------------------------------
Dial_Ready:   ; Dial Pulse Ready
```

```
;Description : Dial Pulse Success
;Parameter   : none
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    : add_wr_buff,inc_index,inc_bc
;
        ld      hl,(wr_index)
        add_wr_cbuff  DIAL_RDY
        ld      a,max_coi
        sub     b
        ld      (hl),a
        inc     hl
        add_wr_cbuff_null
        inc_bc
        inc_index_wr_index
        set_res
        ret
;-------------------------------------------------------------
CO_ONHOOK:    ;  CO_ONHOOK
;Description :  Sub supported command  CO_ONHOOK
;               Set COI -> onhook
;Parameter   :  rd_cbuff
;Return Value:  none
;Used Reg    :  a,bc,de,hl
;Call Sub    :  none
;
        ld      hl,(rd_index)
        inc     hl
        ld      b,(hl)
        inc     b
        ld      de,coi_st-1
        ld      c,01111111b
con1:   inc     de
        rlc     c
        djnz    con1
        ld      a,ONHOOK_ST
        ld      (de),a
        ld      a,(hook_stu)
        and     c
        ld      (hook_stu),a
        ld      a,0
        ret
;-------------------------------------------------------------
CO_OFFHOOK:   ;  CO_OFFHOOK
;Description :  Sub supported command  CO_OFFHOOK
;               Set COI -> offhook
;Parameter   :  rd_cbuff
;Return Value:  none
;Used Reg    :  a,bc,de,hl
;Call Sub    :  none
;
        ld      hl,(rd_index)
        inc     hl
        ld      b,(hl)
        inc     b
        ld      de,coi_st-1
        ld      c,10000000b
coff1:  inc     de
        rlc     c
        djnz    coff1
        ld      a,OFFHOOK_ST
        ld      (de),a
        ld      a,(hook_stu)
        or      c
        ld      (hook_stu),a
        ld      a,0
        ret
;-------------------------------------------------------------
CO_FLASH:     ;  CO_FLASH
;Description :  Sub supported command  CO_FLASH
;               Set COI flash
;Parameter   :  rd_cbuff
;Return Value:  none
;Used Reg    :  a,bc,de,hl
;Call Sub    :  none
;
        ld      hl,(rd_index)
        inc     hl
        ld      b,(hl)
        inc     b
```

```
        ld      de,coi_st-1
        ld      c,01111111b
cofl1:  inc     de
        rlc     c
        djnz    cofl1
        ld      a,FLASH_ST
        ld      (de),a
        ld      de,COI_CNT
        ld      a,(tc_flash)
        ld      (de),a
        ld      a,(hook_stu)
        and     c
        ld      (hook_stu),a
        ld      a,0
        ret
;-------------------------------------------------------------
CO_DIAL:        ; CO DIAL
;Description :  Sub supported command  CO DIAL
;               Set COI DIAL number
;Parameter   :  rd_stuff
;Return Value:  none
;Used Reg    :  a,bc,de,hl
;Call Sub    :  none
;
        ld      hl,(rd_index)
        inc     hl
        ld      b,(hl)
        inc     b
        ld      de,coi_st-1
        ld      c,01111111b
codi1:  inc     de
        rlc     c
        djnz    codi1
        ld      a,DIALOK_ST
        ld      (de),a
        ld      a,(hook_stu)
        and     c
        ld      (hook_stu),a
        ld      de,COI_CNT              ;Get start add of COI CNT
        ld      a,(hl)                  ;Get COI#
        add     a,e
        ld      e,a
        ld      a,(tc_don)
        ld      (de),a                  ;Save Time const to COI CNT
        ld      de,COI_NUM              ;Get start add of COI NUM
        ld      a,(hl)                  ;GET COI#
        add     a,e
        ld      e,a
        inc     hl                      ;point to digit number
        ld      a,(hl)                  ;Get digit number
        cp      10
        jp      nc,codi3                    ; if COI_NUM >9 then COI_NUM=1
        cp      0
        jp      nz,codi2
        ld      a,10
codi2:  ld      (de),a                  ;Save Digit number to COI_NUM
        jp      codie
codi3:  ld      a,1
        ld      (de),a
codie:  ld      a,0
        ret
;-------------------------------------------------------------
Set_TSAC:       ;Set Time Slot Assigner Circuit
;Description :  Set -both tx/rx timeslot  (00h)
;                   -only Tx              (40h)
;                   -only Rx              (80h)
;Parameter   : 30h,xxh,yyh  xx=SSI
;                           yy=TS# or (TSAC Command)
;Return Value:  B=-  1=Sucess / 0=Timeout
;Used Reg    :  a,bc,hl
;Call Sub    :  Prg_TSAC
;
        ld      hl,(Rd_index)   ;Command
        inc     hl
        ld      a,(hl)          ;SSI#
        ld      b,a             ;B <- SSI#  ;0-7)
        inc     hl
        ld      a,(hl)          ;TSAC command + TS#
        ld      c,a             ;C<- TSAC com + TS#
        call    prg_Tsac
```

```
;       inc_index   rd_index
        ld      a,0
        ret
;-------------------------------------------------------------
Set_PCM:        ;Set Tx/Rx PCM line
;Description :  Set -only Tx
;                   -only Rx
;Parameter   :  31h,xxh,xxh  xx000000b xx=00--> both
;                                      xx=01--> tx
;                                      xx=10--> rx
;                              yy=data 00tttrrr
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  none
;
        ld      hl,(Rd_Index)
        inc     hl
        ld      a,(hl)          ; xx000000h  xx=00->both
                                ;            xx=01->tx
                                ;            xx=10->rx
        and     0c0h            ;if a=0 then Tx
        cp      01000000b       ;if Zero -> Tx
        jr      z,spcmtx
        cp      10000000b       ;if zero -> Rx
        jr      z,spcmrx
        ld      a,(hl)
both:   jr      spcm2
spcmtx: ld      a,(hl)          ; 00000ttth -->Tx
        and     00000111b
        rlca
        rlca
        rlca
        ld      b,a             ; 00ttt000h
        ld      a,(_pcm)        00xxxrrrh
        and     07h             ; 00000111h
        or      b               ; 00tttrrrh -->A
        jr      spcm2
spcmrx: ld      a,(hl)          ; 00010rrrh
        and     07h             ; 00000111h
        ld      b,a             ; 00000rrrh
        ld      a,(_pcm) ; 00tttxxxh
        and     00111000b       ; 00111000h
        or      b               ; 00tttrrrh
spcm2:  ld      (_pcm),a
        out     (P8255B),a
;       inc_index   rd_index
        ld      a,0
        ret
;-------------------------------------------------------------
OTHER:          ;OTHER
;Description :  Send Unrecognize command back to MPU (TEST)
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,de,bc
;Call Sub    :  set_req
;
        push    de
        ld      hl,(wr_index)
        add wr_cbuff 6h
        ld      de,(rd_cbuff)
        ld      b,6
o1:     ld      a,(de)
        ld      (hl),a
        djnz    o1
        INC_BC
        INC_INDEX wr_index
        set_req
        pop     de
        ret

        endf
```

```
            list    a.e
;-------------------------------------------------------------
;   TSI-Tone Gen Module
;   Name        : TG009.SRC
;   Description : TSI-Tone Gen module Control Program
;   Version     : 1.0
;   Date        : January 1991
;-------------------------------------------------------------
ROM             equ 0           ;Rom start address
RAM             equ 2000h       ;Ram start address
TSI_RAM         equ 0ff00h      ;Time slot interchange address
;
P8255           equ 90h         ;Port 8255
;
P8255A          equ P8255
P8255B          equ P8255+1
P8255C          equ P8255+2
P8255CMD        equ P8255+3
;
PTSAC_CS        equ P8255A      ;Port Chip select
PTSAC_DI        equ P8255C      ;Port Data input
PTSAC_CTS       equ P8255C      ;Port Clear to send
;==PRING_EN      equ 0eh         ;Port Ring enable
;==PHOOK_STATUS  equ 0fh         ;Port Hook status
;
MAX_COMMAND     equ 240         ;Maximum command
max_ts          equ 255         ;(256=255+1 -->0)
max_tone        equ 10
max_std_tone    equ 5
;
_TSI_Dev_Num    equ 02h         ;TSI device number
_Dev_Num        equ _TSI_Dev_Num
_Chk_Dev        equ 20h         ;Check device number command
_Clr_TSI        equ 21h         ;Clear TSI command
_C_Tone         equ 23h         ;Connect tone command
_EX_TS          equ 24h         ;Exchange timeslot command
_TEST           equ 25h         ;Test command
_MOD_FAIL       equ 26h         ;Module #xx absent
;-------------------------------------------------------------
;   Inteface to MPU
;-------------------------------------------------------------
INF_H_OUT       equ 0AH         ;MPU Inteface Handshake output
INF_H_IN        equ 09H         ;MPU Inteface Handshake input
INF_DATA        equ 08H         ;MPU Inteface Data
;-------------------------------------------------------------
;   I/O HANDSHAKE SIGNAL
;   <INPUT>
;-------------------------------------------------------------
;                   VALUE
PM_EN           equ 02H         ;PM enable          BIT 1
REQ_ACK         equ 04H         ;Request acknowledge BIT 2
RST_PM          equ 08H         ;Reset PM           BIT 3
;   <OUTPUT>        BIT#
NAK_ACK         equ 0           ;Nak-Ack -->data correct;       BIT 0
DATA_RDY        equ 1           ;Data ready                     BIT 1
REQ             equ 2           ;Request from PM                BIT 2
RST_CNT         equ 3           ;Reset inteface address counter BIT 3
;-------------------------------------------------------------
;   ASCII Control code
;
NULL            equ 0CH         ;NULL
ACK             equ 06H         ;Ack
NAK             equ 15H         ;Nak
;-------------------------------------------------------------
;   Tone Constants
;
DT              equ 0           ;Dial tone
RBT             equ 1           ;Ring back tone
Busy            equ 2           ;Busy tone
NT              equ 3           ;Null tone
SDT             equ 4           ;Special dial tone
ROT             equ 5           ;Reorder tone
SST             equ 6           ;Service Set tone
WT              equ 7           ;Warning tone
hold            equ 8           ;Hold tone
Camp            equ 9           ;Camp tone
Music           equ 10          ;Music on hold
DTMF1           equ 11
```

```
DTMF2           equ 12
DTMF3           equ 13
DTMF4           equ 14
DTMF5           equ 15
DTMF6           equ 16
DTMF7           equ 17
DTMF8           equ 18
DTMF9           equ 19
DTMF0           equ 20
DTMF11          equ 21
DTMF12          equ 22
;
;T_NMI           125 mS          ;Non maskable interrupt interval 12.5 ms
                                ;Tone  on/off   fundamental
;-----------------------------------------------------------
;
        FILE    tg009_0.src     ;BIOS MACRO FILE
;
        org     PCM
        ld      sp,stk_pt       ;Set stack pointer
        di
        im      1
        ld      b,0
        djnz    $
        jp      MAIN
;
        int_service             ;Interrupt service macro
;
        org     100h
;-----------------------------------------------------------
MAIN:           ; MAIN
;Description : Main Program
;Parameter   : none
;Return Value: none
;Used Reg    : none
;Call Sub    :
;
        call    init_hs         ;Initialize handshake
        call    clear_buff      ;Clear command buffer
        call    init_cnt        ;initialize tone counter
        call    init_stu        ;initialize status
        call    init_tsi        ;initialize TSI
        call    init_8255       ;initialize 8255
        call    init_tone       ;initialize tone TSAC
        call    init_PCM        ;initialize PCM
        call    en_nmi          ;Enable NMI
main1:
        halt
        call    exec_com        ;Execute command send from MPU
        jr      main1
;-----------------------------------------------------------
;Bios file
        file    tg009_1.src
        ;Service Function file
        file    tg009_2.src
        ;Service Function file
        file    tg009_3.src
;-----------------------------------------------------------
;   Tone time intervals
;
_sdton:     defb    1       ;SDT  125/125   (DT)
_sdtoff:    defb    1
_busyon:    defb    4       BUSY 500/500   (BUSY)
_busyoff:   defb    4
_rbton:     defb    8       ;RBT 1000/3000  (RBT)
_rbtoff:    defb    24
_rotoff:    defb    2       ;ROT 250/250    (BUSY)
_roton:     defb    2
_wton:      defb    1       ;WT  125/125    (DT)
_wtoff:     defb    1
_campon:    defb    7       ;Camp 300 ms    (DT)
_dtmfon:    defb    2       ;DTMF 250 ms
;-----------------------------------------------------------
;   Constant value
;   pcm, ts, tone ts table
;
pcm_std:    defb    00000000b   ;PCM#7
;
tsi_table:                      ;All Tone sent to PCM7
_TSi_DT:    defb    8fh         ;ts#17
```

```
_TSi_RBT:    defb    97h          ;ts#18
_TSi_Busy:   defb    9fh          ;ts#19
_TSi_NT:     defb    00h          ;ts#0
_tsi_music:  defb    0afh         ;ts#21
_tsi_dtmf1   defb    0fh          ;ts#1
_tsi_dtmf2   defb    17h          ;ts#2
_tsi_dtmf3   defb    1fh          ;ts#3
_tsi_dtmf4   defb    27h          ;ts#4
_tsi_dtmf5   defb    2fh          ;ts#5
_tsi_dtmf6   defb    37h          ;ts#6
_tsi_dtmf7   defb    3fh          ;ts#7
_tsi_dtmf8   defb    47h          ;ts#8
_tsi_dtmf9   defb    4fh          ;ts#9
_tsi_dtmf0   defb    57h          ;ts#10
_tsi_dtmf11  defb    5fh          ;ts#11
_tsi_dtmf12  defb    67h          ;ts#12
;
ts_table:
_ts_dt:      defb    0ffh
_ts_rbt:     defb    0ffh
_ts_busy:    defb    0ffh
 ts_nt:      defb    0ffh
_ts_music:   defb    0ffh
;-----------------------------------------------------------------
;   Define value
;
        org     RAM
ts_cnt: defs    256
ts_stu: defs    256
H_IN_ADD:    defs    1
H_OUT_ADD:   defs    1
MUX_ADD:     defs    2
REQ_ADD:     defs    1
WR_Index:    defs    2
Rd_Index:    defs    2
        ORG     2400H
TSI_MEM_BAK:
        DEFS    256
;-----------------------------------------------------------------
;   Command Buffer
;
            org     2500h
RD_CBUFF:   defs    max_command
            org     2600h
WR_CBUFF:   defs    max_command
            org     2700h
temp_cbuff  defs    max_command

;-----------------------------------------------------------------
;   Timeslot interchange location
;
            org     TSI_PAM
Tsi_mem:    defs    256
            org     ram+1fffh
stk_pt:     defs    1
        end
        end
```

```
;-------------------------------------------------------
;   TSI-Tone_Gen Module
;   Name        : TG009_0.SRC
;   Description : Bios Macro
;                 - Macro initialize interrupt service routine
;                 - Macro support communicate with MPU
;                 - Macro support generate tone
;   Version     : 1.0
;   Date        : January 1991
;-------------------------------------------------------
Int_service    Macro
;Description : Interrupt service routine
;            :  - Service all interrupt
;Parameter   : none
;Return Value: none
;Used Reg    : all
;Call Sub    : none
;
RST28:  org     28H
        reti
RST30:  org     30H
        reti
RST38:  org     38H             ; Service communication
        di
        jp      Comm_service
        reti
NMI:    org     66H             ; Scan & analysis SSI status
        jp      NMI_Service
        retn
        endM
;---------------------------------------
Reset_Cnt     Macro
;Description : Reset counter in MPU-PPU interface
;Parameter   : none
;Return Value: none
;Used Reg    : hl,b,a
;Call Sub    : none

        ld      hl,H_out_ADD
        res     RST_CNT,(hl)
        out     (INF_H_out),a
        ld      b,0AH           ;20 MICROSEC
        djnz    $
        set     RST_CNT,a
        out     (INF_H_out),a
        ld      (hl),a
        endM

;---------------------------------------
Dis_H_out     Macro  #H_out
;Description : Disable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none
;
        ld      hl,H_out_ADD
        set     #H_out,(hl)
        ld       a,(hl)
        out     (INF_H_out),a
        endM
;---------------------------------------
En_H_out     Macro  #H_out
;Description : Enable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none

        ld       hl,H_out_ADD
        res      #H_out,(hl)
        ld       a,(hl)
        out      (INF_H_out),a
        endM.
;---------------------------------------
Wr_Inf          Macro
;Description : Copy C_Buff from PPU to MPU
;Parameter   : none
;Return Value: none
;Used Reg    : a,b,hl
```

```
;Call Sub    :
        LOCAL #WR_INF_1
        ld      hl,WR_CBUFF       ;Byte_count
        ld      b,(hl)
        inc     b                 ; add one byte for Byte_count
#WR_INF_1: ld   a,(hl)
        out     (INF_DATA),a
        inc     hl
        djnz    #WR_INF_1
        endM
;-----------------------------------------
Add_Wr_Cbuff  Macro #data
;Description : Add a data to wr_cbuff
;Parameter   : #data
;Return Value: none
;Used Reg    : a,hl
;Call Sub    : none
        ld      a,#DATA
        ld      (hl),a
        inc     hl
        endM
;-----------------------------------------
Clr_Cbuff     Macro #Cbuff
;Description : Clear Command Buffer
;Parameter   : #cbuff
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    : none
        LOCAL   #L1
        ld      hl,#CBUFF
        ld      b,MAX_COMMAND+1
        xor     a
#L1:    ld      (hl),a
        inc     hl
        djnz    #L1
        endM
;-----------------------------------------
Set_Req         Macro
;Description : Activate Request signal
;Parameter   : none
;Return Value: none
;Used Reg    : a
;Call Sub    : En_h_out
        ld        a,ACK
        ld        (Req_add),a
        En_H_out REQ
        endM
;-----------------------------------------
Inc_index       Macro #index
;Description : Increment rd_index or Wr_index
;Parameter   : #index
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        push    hl
        ld      hl,(#index)
        inc     hl
        inc     hl
        inc     hl
        ld      (#index),hl
        pop     hl
        endM
;-----------------------------------------
Inc_BC          Macro
;Description : Increment Byte count
;Parameter   : none
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        ld      a,(wr_cbuff)
        add     a,3
        ld      (wr_cbuff),a
        endM
;-------------------------------------------------------------
En_Tone:    Macro   #data
;Description :  Enable tone to TS#
;Parameter   :  c=TS#
;Return Value:  none
;Used Reg    :  a,hl,b
;Call Sub    :  none
```

```
;
        push    hl
        ld      hl,Tsi_mem
        ld      a,l
        add     a,c
        ld      l,a
        ld      a,(#data)
        ld      (hl),a
        pop     hl
        endM
;-------------------------------------------------------------
Dis_Tone:   Macro
;Description :  Disable tone to TS#
;Parameter   :  c=TS#
;Return Value:  none
;Used Reg    :  a,hl,b
;Call Sub    :  none
;
        push    hl
        ld      hl,Tsi_mem
        ld      a,l
        add     a,c
        ld      l,a
        ld      a,(_tsi_nt)
        ld      (hl),a
        pop     hl
        endM
;-------------------------------------------------------------
GEN_TONE    Macro   #tone,#tcon,#tcoff
;Description :  Genreate Tone
;Parameter   :  #tone, time interval on/off
;Return Value:  none
;Used Reg    :  a,hl,de,ix
;Call Sub    :  none
;
        local   #gentt,#gentw,#gente
        ld      a,(hl)          ;load status to a
        and     80h             ;Mask 1000 0000b ,if zero->wait
                                ;                else ->tone
        jr      z,#gentw
#gentt: ld      a,(#tcoff)      ;load not-off tc
        ld      (de),a
        ld      a,(_tsi_nt)
        ld      (ix),a          ;disable tone
        res     7,(hl)          ;set status to wait
        jr      #gente
#gentw: ld      a,(#tcon)       ;load not-on tc
        ld      (de),a
        ld      a,(#tone)
        ld      (ix),a          ;enable tone
        set     7,(hl)          ;set status to tone
#gente: ld      a,0
        endM

        endf
```

```
;-------------------------------------------------------
;   TSI-Tone_Gen Module
;   Name        : TG009_1.SRC
;   Description : BIOS, Service Function
;                   - Program tsac
;                   - Enable NMI
;                   - Generate tone
;                   - Clear command index
;                   - Convert form ts/pcm -> tsi-add
;   Version     : 1.0
;   Date        : January 1991
;-------------------------------------------------------
Prg_TSAC:    ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: B-> 1=Sucess / 0=Timeout
;Used Reg    : a,bc
;Call Sub    : none
;
        push    bc
        inc     b               ;B = (0-7) ,increment b before use
stsacs: ld      a,80h
stsac1: rlca
        djnz    stsac1
        out     (PTSAC_CS),a    ;Set CS
        ld      a,c
        ld      b,08h
        rlc     c
        rlc     c
stsac2: ld      a,c
        or      0fdh            ;1111 1101
        out     (PTSAC_DI),a    ;send data to port c low
        dec     a
        nop
        nop
        out     (PTSAC_DI),a    ;send data to port c hi
        rlc     c
        djnz    stsac2
        ld      b,128
stsac3: dec     b
        jp      z,stsac_to      ;If b>256 Time out
        in      a,(PTSAC_CTS)
        bit     4,a             ;CTS
        jp      nz,stsac3
        pop     bc
        ld      b,1             ;Success
        jp      stsac_e
stsac_to:
        pop     bc
        ld      hl,(wr_index)
        add_wr_cbuff _MOD_FAIL
        ld      a,b
        ld      (hl),a
        inc BC
        inc index wr_index
        set_req
        ld      b,0
stsac_e:  ld    a,0
        out     (PTSAC_CS),a
        ret
;-------------------------------------------------------
;Prg_TSAC:    ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: B-> 1=Sucess / 0=Timeout
;Used Reg    : a,bc
;Call Sub    : none
;
;       inc     b               ;B = (0-7) ,increment b before use
;stsacs:    ld       a,80h
;stsac1: rlca
;       djnz    stsac1
;       out     (PTSAC_CS),a    ;Set CS
;       ld      a,c
;       ld      b,08h
;       rlc     c
;       rlc     c
;stsac2:    ld       a,c
;       or      0fdh            ;1111 1101
;       out     (PTSAC_DI),a    ;send data to port c low
```

```
;       dec     a
;       out     (PTSAC_OI),a    ;send data to port c hi
;       rlc     c
;       djnz    stsac2
;       ld      b,0
;stsac3: inc     b
;       jp      z,stsac_to      ;If b>256 Time out
;       in      a,(PTSAC_CTS)
;       and     10h             ;CTS
;       jp      nz,stsac3
;       ld      b,1             ;Success
;       jp      stsac_e
;stsac_to: ld    b,0
;stsac_e:  ld    a,0
;       out     (PTSAC_CS),a
;       ret
;-----------------------------------------------------------
EN_NMI:  ;      Enable non maskable interrupt
;Description :  Send one pulse to CS4 for enable NMI to system
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,10000000b             ;1000 0000b   bit 7
        out     (PTSAC_CS),a    ;Send 1 to data
        ld      a,11000000b
        out     (PTSAC_CS),a    ;Send 1 to data/clock
        ld      a,0
        out     (PTSAC_CS),a    ;Send 0 to data/clock
        ret
;-----------------------------------------------------------
GEN_RBT:        ;  Generate RBT
;Description :  Generate Ringback tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  macro gen_tone
;
        gen_tone _tsi_RBT,_RBTon,_RBToff
        ret
;-----------------------------------------------------------
GEN_BUSY:       ;  Genreate BUSY
;Description :  Generate busy tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  macro gen_tone
;
        gen_tone _tsi_busy,_busyon,_busyoff
        ret
;-----------------------------------------------------------
GEN_SDT:        ;  Generate SDT
;Description :  Generate Speical dial tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  macro gen_tone
;
        gen_tone _tsi_dt,_sdton,_sdtoff
        ret
;-----------------------------------------------------------
GEN_ROT:        ;  Generate ROT
;Description :  Generate Reorder tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  gen_tone
;
        gen_tone _tsi_RBT,_ROTon,_ROTcff
        ret
;-----------------------------------------------------------
GEN_WT:
;Description :  Genreate Warning tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  macro gen_tone
;
        gen_tone _tsi_DT,_WTon,_WToff
```

```
        ret
;-------------------------------------------------------------
GEN_Camp:
;Description :  Genreate Warning tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,de,hl,ix
;Call Sub    :  none
;
        ld      a,(hl)          ;load status to a
        and     80h             ;mask 1000 0000b ,if zero->wait
                                ;                else ->tone
        jr      z,gencw
genct:  ld      a,1             ;set tc
        ld      (de),a
;       ld      a,(_tsi_nt)
        LD      A,(IY)
        ld      (ix),a          ;disable tone
        ld      a,NT
        ld      (hl),a          ;clear status to NT
        jr      gence
gencw:  ld      a,(_campon)     ;load camp_on-on tc
        ld      (de),a
        ld      a,(_tsi_dt)
        ld      (ix),a          ;enable tone
        set     7,(hl)          ;set status to tone
gence:  ld      a,0
        ret
;-------------------------------------------------------------
GEN_DT:       ;  Generate DT
;Description :  Generate Dial tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  none
;
        ld      bc,tsi_ram
        ld      c,(hl)
        ld      a,(_tsi_dt)
        ld      (bc),a
        ld      a,0
        ret
;-------------------------------------------------------------
GEN_NT:       ;  Generate NT
;Description :  Generate Null tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  none
;
        ld      bc,tsi_ram
        ld      c,(hl)
        ld      a,(_tsi_nt)
        ld      (bc),a
        ld      a,0
        ret
;-------------------------------------------------------------
GEN_SST:      ;  Generate SST
;Description :  Generate Service Set tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  none
;
        ld      bc,tsi_ram
        ld      c,(hl)
        ld      a,(_tsi_dt)
        ld      (bc),a
        ld      a,0
        ret
;-------------------------------------------------------------
GEN_MUS:      ;  Generate MUS
;Description :  Generate music
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  none
;
        ld      bc,tsi_ram
        ld      c,(hl)
```

```
        ld      a,(_tsi_MUSic)
        ld      (bc),a
        ld      a,0
        ret
;-------------------------------------------------------------
GEN_DTMF:
;Description :  Genreate DTMF tone
;Parameter   :  tone number in A
;Return Value:  none
;Used Reg    :  a,de,hl,ix
;Call Sub    :  none
;
        sub     dtmf1           ; a-->dtmf#
        ld      b,a
        ld      a,(hl)          ;load status to a
        and     80h             ;mask 1000 0000b ,if zero-->wait
                                ;                 else -->tone
        jr      z,gendtmfw
gendtmft:
        ld      a,1             ;set tc
        ld      (de),a          ;de-->ts counter
;       ld      a,(_tsi_nt)
        LD      A,(IY)
        ld      (ix),a          ;disable tone (ix-->tsi mem)
        ld      a,NT
        ld      (hl),a          ;Clear status to NT
        jr      gendtmfe
gendtmfw:
        push    hl
        ld      a,(_dtmfon)     ;load dtmf tc
        ld      (de),a
        ld      a,b
        ld      hl, tsi_dtmf1
        add     a,l
        ld      l,a
        ld      a,(hl)
        ld      (ix),a          ;enable tone
        pop     hl
        set     7,(hl)          ;set status to tone
gendtmfe:
        ld      a,0
        ret
;-------------------------------------------------------------
Clr_wr_index::  Clear write Index
;Description :  Clear write Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,hl
;Call Sub    :  none
;
        ld      hl,wr_cbuff+1
        ld      (Wr_index),hl
        ld      a,0
        ld      (wr_cbuff),a
        ret
;-------------------------------------------------------------
Clr_rd_index:;  Clear read Index
;Description :  Clear read Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl
;Call Sub    :  none
;
        ld      hl,rd_cbuff+1
        ld      (Rd_index),hl
        ret
;-------------------------------------------------------------
Cnv_TS:         ;  Convert timeslot
;Description :  Convert format ts/pcm to tsi add
;Parameter   :  B=ts#  ,  C=PCM#
;Return Value:  a=TSI_ADD
;Used Reg    :  A,BC
;Call Sub    :  None
;
        ld      a,b
        sla     a
        sla     a
        sla     a
        or      c
        ret
```

```
;-------ENDE-----------------------------------------------------
;   TSI-Tone_Gen Module
;   Name        : TG009_2.SRC
;   Description : Service Function
;                   - Execute command
;                   - NMI Service
;                   - Communication service
;                   - Initialize
;   Version     : 1.0
;   Date        : January 1991
;--------------------------------------------------------------
Exec_com:     ;  Exec command
;Description :  Execute command which sent from MPU
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  Chk_dev,Clr_tsi,C_tone,Ex_ts,Test
;
        di
        ld      hl,rd_cbuff     ;Read byte count
        ld      a,(hl)
        cp      0               ;If byte count equal 0 then RET
        jp      z,execee
        ld      b,a             ;   else do until last command
        inc     hl
        ld      (rd_index),hl   ;Read index  - first command
exec1:  ld      hl,(rd_index)
        push    bc
        ld      a,(hl)          ;Read command
        cp      _chk_dev        ;Compare commad
        call    z,chk_dev       ; Jump if correct else skip
        cp      _Clr_tsi
        call    z,Clr_tsi
        cp      _C_Tone
        call    z,C_Tone
        cp      _Ex_Ts
        call    z,Ex_ts
        cp      _Test
        call    z,Test
        cp      0
        jp      z,exec2
        call    other
        pop     bc
        jp      execc
exec2:  inc index   rd_index    ;Go to next command
        pop     bc
        dec     b
        dec     b               ;Decrement loop counter
        djnz    exec1            ;Repeat until Last command
execc:  ld      hl,rd_cbuff     ;Set rd_index to first command
        ld      (rd_index),hl
        ld      a,0
        ld      (hl),a
execee: EI
        ret
;--------------------------------------------------------------
NMI_Service:  ;  Non maskable Interrupt service routine
;Description :  Increment tone counter - Generate all tones
;Parameter   :  none
;Return Value:  none
;Used Reg    :  all
;Call Sub    :  inc_cnt2, anl_tone
;
        exx
        ex      af,af'
        push    ix
        push    iy
        call    inc_cnt2
        call    anl_tone
        pop     iy
        pop     ix
        ex      af,af'
        exx
        ei
        retn
;--------------------------------------------------------------
Comm_service: ; Communnication service routine
;Description : Communication routine
;               - communicate with MPU
;Parameter   : h_in,h_out
```

```
;Return Value: content in C_Buff
;Used Reg     : none
;Call Sub     : none
;
        exx                             ;Save old content
        ex      af,af'
        in      a,(INF_H_in)            ;Check REQ_ACK before check mode
        and     REQ_ACK                 ;Mask for REQ_ACK
                                        ;  If a=0 then REQ_ACK active
        cp      00H                     ;After compare with Zero
                                        ;
        JP      nz,RD                   ;If not ZERO then NO REQ_ACK ,goto rd
        ld      a,(REQ_ADD)             ;Check Request Flag
        cp      ACK                     ;   If ACK then Request to Send data
        JP      nz,NR_NW                ;If no Request then No read,No write
                                        ;Write command to inteface
WR:     WR_INF                          ;Command write complete
        ld      a,NAK                   ;Clear request flag
        ld      (REQ_ADD),a
        Dis_H_Out_REQ                   ;Disable request signal
        En_H_Out  DATA_RDY              ;Enable data_rdy signal
        call    clr_wr_index            ;Clear wr_index ,clear byte count
        JP      comms_e
                                        ;Read command from interface
RD:     ld      hl,RD_CBUFF
        in      a,(INF_DATA)            ;Read Byte count
        ld      (hl),a
        ld      b,a
        inc     hl
RD_1:   in      a,(INF_DATA)            ;Read Command/Data
        ld      (hl),a
        inc     hl
        djnz    RD_1                    ;Repeat Until Byte count=0
RD_E:   En_H_Out  DATA_RDY              ;Enable DATA_RDY
        call    clr_rd_index            ;Clear rd_index
        JP      comms_e
                                        ;No read ,No write
NR_NW:  ld      hl,WR_CBUFF             ; This Case - REQ_ACK = active
        ADD_WR_CBUFF  3                 ;       but     NO REQUEST
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        En_H_Out   DATA_RDY
comms_e:Dis_H_Out  DATA_RDY
        ld      hl,WR_CBUFF+1
        ld      (Wr_Index),hl
        ld      hl,rd_Cbuff+1           ; Skip Byte count
        ld      (Rd_index),hl
        ex      af,af'
        exx
        ei
        RETI
;-------------------------------------------------------------
Inc_Cnt2:     ;  Increment counter
;Description :  Increment Tone Counter
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,hl,de
;Call Sub    :  none
;
        ld      hl,ts_stu
        ld      de,ts_cnt
        ld      b,max_ts
incnt1:ld       a,(hl)
        cp      nt
        jp      z,incnt2
        ld      a,(de)
        and     a
        JP      z,incnt2
        dec     a
        ld      (de),a
incnt2:inc      de
        inc     hl
        djnz    incnt1
        ret
;
        jp      $
;-------------------------------------------------------------
Inc_Cnt:
;Description :  NOt USE ***
```

```
;Parameter    :
;Return Value:
;Used Reg     :
;Call Sub     :
;
        ld      a,07h
        out     (p8255b),a
        ld      de,ts_cnt
        ld      b,max_ts
incnt01:ld      a,(de)
        and     a
        jr      z,incnt02
        dec     a
        ld      (de),a
incnt02:inc     de
        djnz    incnt01
        ld      a,C0h
        out     (p8255c),a
        ret
;-------------------------------------------------------------
ANL_TONE:     ;  Analyze Tone
;Description :  Generate Tone for each TS
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,b,de,hl,ix
;Call Sub    :  gen_rbt, gen_busy, gen_sdt, gen_rdt, gen_warn
;               gen_camp
;
        ld      hl,ts_stu       ;hl-> time-slot status index
        ld      de,ts_cnt       ;de-> time-slot counter index
        ld      ix,tsi_mem      ;ix-> speech control ram
        LD      IY,TSI_MEM_BAK
        ld      b,max_ts        ;b -> loop counter
anlt1:  ld      a,(de)          ;load a with counter
        and     a               ;if counter= 0 then go to next TS
        jp      nz,anlt2        ;   else mask
        ld      a,(hl)
        and     00011111b
        cp      RBT             ;Compare status with Ring back tone
        call    z,GEN_RBT
        cp      Busy            ;Compare status with BUSY tone
        call    z,GEN_BUSY
        cp      SDT             ;Compare status with Special dial tone
        call    z,GEN_SDT
        cp      RDT             ;Compare status with Reorder tone
        call    z,GEN_RDT
        cp      WT              ;Compare status with Warning tone
        call    z,GEN_WT
        cp      Camp            ;Compare status with Camp on tone
        call    z,Gen_Camp
        and a
        call    nz,Gen_DTMF
anlt2:  inc     hl
        inc     de
        inc     ix
        INC     IY
        djnz    ANLT1
        ret
;-------------------------------------------------------------
Init_8255:    ;  init 8255
;Description :  Set up 8255
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,88h           ;Program 8255
        out     (P8255CMD),a
        out     (P8255CMD),a
        ret
;-------------------------------------------------------------
init_PCM:     ;  Initialize PCM
;Description :  Set up PCM line
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :
;
        ld      a,(pcm_std)
        out     (P8255b),a      ;Set pcm line  to default
```

```
        out     (P8255b),a
        ret
;-------------------------------------------------------------
init_tone:    ;  Initialize Tone Gen
;Description  :  set up timeslot table for each tone
;Parameter    :  none
;Return Value:  none
;Used Reg     :  a,bc,hl
;Call Sub     :  prg_tsac
;
        ld      hl,ts_table
        ld      b,DT            ;Dial tone to ts 0
        ld      c,(hl) -
        call    prg_tsac
        inc     hl
        ld      b,RBT           ;ring back to ts 1
        ld      c,(hl)
        call    prg_tsac
        inc     hl
        ld      b,Busy      ;busy tone to ts 2
        ld      c,(hl)
        call    prg_tsac
        inc     hl
        ld      b,NT            ;null tone to ts 3
        ld      c,(hl)
        call    prg_tsac
        inc     hl
        ld      b,nt+1 .        ;music on hold ts 4
        ld      c,(hl)
        call    prg_tsac
        ret
;-------------------------------------------------------------
init_cnt:     ;  initialize Counter
;Description  :  clear tone counter
;Parameter    :  none
;Return Value:  none
;Used Reg     :  a,b,hl
;Call Sub     :  none
;
        ld      hl,ts_cnt
        ld      a,1
        ld      b,max_ts
initcnt1: ld    (hl),a
        inc     hl
        djnz    initcnt1
        ret
;-------------------------------------------------------------
init_stu:     ;  Initialize status
;Description  :  Set up tone stutus to null tone
;Parameter    :  none
;Return Value:  none
;Used Reg     :  hl,a,b
;Call Sub     :  none
;
        ld      hl,ts_stu
        ld      a,NT
        ld      b,max_ts
initstu1: ld    (hl),a
        inc     hl
        djnz    initstu1
        ret
;-------------------------------------------------------------
init_TSI:     ;  Initialize TSI
;Description  :  Set up Timeslot Interchage
;Parameter    :  none
;Return Value:  none
;Used Reg     :  a,b,hl
;Call Sub     :  none
;

        ld      hl,tsi_mem
        ld      a,(_tsi_nt)
        ld      b,max_ts
inittsi1: ld    (hl),a
        inc     hl
        djnz    inittsi1
        ret
;-------------------------------------------------------------
init_hs:      ;  Initialize Handshake
;Description  :  initialize handshake
```

```
;Parameter    :  none
;Return Value:  none
;Used Reg     :  a
;Call Sub     :
;
        ld      a,0ffh          ;Clear Handshake output
        out     (INF_H_OUT),a
        ld      (H_OUT_ACK),a   ;Clear Handshake output Status flag
        ld      a,nak
        ld      (REQ_ACK),a     ;Clear Request flag
        ret
;--------------------------------------------------------------
clear_buff:  ;  Clear buffer
;Description :  Clear wr_cbuff, rd_cbuff
;Parameter    :  none
;Return Value:  none
;Used Reg     :  none
;Call Sub     :  macro clear_buff, clr_rd_index, clr_wr_index
;
        clr_cbuff   rd_cbuff
        clr_cbuff   wr_cbuff
        call    clr_rd_index
        call    clr_wr_index
        ret
        endf
```

```
;-------------------------------------------------------------
;   TSI-Tone_Gen module
;   Name        : TG099_3.SRC
;   Description : Service Function
;                   - clr_tsi, c_tone, ex_ts, test,
;                   - chk_dev, other
;   Version     : 1.0
;   Date        : January 1991
;-------------------------------------------------------------
Chk_Dev:        ;Check Device Number to MPU
;Description :  Send Device Number to MPU
;Parameter   :  None
;Return Value:  Content in Wr Cbuff
;Used Reg    :  HL,A
;Call Sub    :  none
;
        ld      hl,wr_cbuff
        ld      de,temp_cbuff
        ld      a,(hl)              ;Read byte count
        inc     hl                  ;Point to first command
        cp      0
        jp      z,chkdev1
        ld      b,0
        ld      c,a
        ldir
        ld      hl,temp_cbuff
        ld      de,wr_cbuff+1+3
        ld      b,0
        ld      c,a
        ldir
chkdev1: ld     hl,wr_cbuff+1
        Add_Wr_Cbuff    DEV_NUM
        Add_Wr_Cbuff    Null
        Add_Wr_Cbuff    Null
        inc_bc
        inc_index   wr_index
        Set_Reg
        ld      a,0
        ret
;-------------------------------------------------------------
Clr_TSI:        ;Clear TSI
;Description :  Clear TSI - Set data to null tone
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,b,hl,de
;Call Sub    :  en_nmi, macro inc_index
;
        ld      hl,(rd_index)
        ld      a,(hl)
        cp      clr_tsi
        jr      nz,clrtsie
        inc     hl
        cp      clr_tsi
        jr      nz,clrtsie
        inc     hl
        cp      clr_tsi
        jr      nz,clrtsie
        ld      hl,tsi_mem          ;clear tsi mem
        ld      b,max_ts
        ld      a,(_TSi_NT)
clrtsi1:ld      (hl),a
        inc     hl
        djnz    clrtsi1
        ld      hl,ts_stu           ;Clear status
        ld      de,ts_cnt           ;Clear counter
        ld      a,nt
        ld      b,max_ts
clrtsi2:ld      (hl),a
        ld      (de),a
        inc     hl
        inc     de
        djnz    clrtsi2
clrtsie:
        call    en_nmi
;**     inc_index   rd_index
        ld      a,0
        ret
;-------------------------------------------------------------
C_Tone:         ;  Connect tone
;Description :  Connect a tone to a timeslot
```

```
;Parameter    :  23h #tone #TS
;Return Value:  none
;Used Reg     :  a,hl,bc
;Call Sub     :  inc_index
;
        ld      hl,(rd_index)
        inc     hl
        ld      a,(hl)          ;a<-tone#
                                ;First compare with continuous tone
        inc     hl
        cp      DT
        call    z,Gen_DT
        cp      NT
        call    z,Gen_NT
        cp      SST
        call    z,Gen_SST
        cp      Music
        call    z,Gen_MUS
        ld      bc,ts_stu
        ld      c,(hl)          ;c<-ts#
        ld      (bc),a          ;(ts_stu) <- tone-wait (TONE#)
        ld      bc,ts_cnt
        ld      c,(hl)
        ld      a,1
        ld      (bc),a          ;set counter to 1
        LD      BC,TSI_MEM:
        LD      DE,TSI_MEM_BAK
        LD      C,(HL)
        LD      E,(HL)
        LD      A,(BC)
        LD      (DE),A          ;SAVE OLD TSI DATA TO TSI_MEM_BAK
;**     inc_index rd_index
        ld      a,0
        ret
;-----------------------------------------------------------
Ex_TS:
;Description :  Set speech control ram .connect two timeslot
;Parameter    :  24h,xxh,yyh
;             :  xxh,yyh = XXXX XYYY -> XXXX X =TS#
;             :                      -> YYY    =PCM#
;Return Value:  none
;Used Reg     :
;Call Sub     :
;
;   EXTS- source to destinaton = id (des),source
;         (send souce data to destination)
;
        ld      hl,(Pd_Index)
        inc     hl
        ld      bc,tsi_mem      ;bc=ff00h       (ss)
        ld      a,(hl)          ; a=source tspcm
        inc     hl
        ld      c,(hl)          ; bc->destination TSI speech control ram (dd)
        ld      (bc),a          ;(ffddh)  <- ss
;**     inc_index   rd_index
        ld      a,0
        ret
;-----------------------------------------------------------
Ex_TS2: ;   (OLD VERSION)
;Description :  Set speech control ram .connect two timeslot
;Parameter    :  24h,xxh,yyh
;             :  xxh,yyh = XXXX XYYY -> XXXX X =TS#
;             :.                     -> YYY    =PCM#
;Return Value:  none
;Used Reg     :
;Call Sub     :
;
        ld      hl,(Pd_Index)
        inc     hl
        ld      bc,tsi_mem      ;bc=ff00h
        ld      c,(hl)          ;bc=ffxxh <-ts#xx
        inc     hl
        ld      a,(hl)          ;a        <-ts#yy
        ld      (bc),a          ;(ffxxh)  <- yy
                                ;
        ld      l,c             ;l=c=xx
        ld      c,a             ;c=a=yy
        ld      a,l             ;a=l=xx
                                ;c=yy , a=xx ,bc=ffyy
        ld      (bc),a
```

```
;**      inc_index   rd_inde
        ld      a,0
        ret
;-------------------------------------------------------------
TEST:           ;  TEST
;Description :  Test condition
;Parameter   :  connect tx-ts to rx-ts (the same timeslot)
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  macro   inc
;
        ld      hl,(rd_ince
        ld      a,(hl)
        cp       test
        jr      nz,teste
        inc     hl
        ld      a,(hl)
        cp       test
        jr      nz,teste
        ld      hl,tsi mem
        ld      b,max_ts
        ld      a,0
test1:  ld      (hl),a
        inc     a
        inc     hl
        djnz    test1
        ld      hl,ts_stu
        ld      b,max_ts
        ld      a,nt
test2:  ld      (hl),a
        inc     hl
        djnz    test2
teste:
;**      inc_index   rd_index
        ld      a,0
        ret
;-------------------------------------------------------------
OTHER:          ;  OTHER
;Description :  Send unrecog.. command back to MPU (TEST)
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,de,bc
;Call Sub    :  set_req
;
        push    bc                      Send back last command
        ld      c,max_command
        ld      b,0
        LD      HL,RD_CBUFF              hl) ->Source
        LD      DE,WR_CBUFF              de) ->Destination
        LDIR                             bc) ->Number
        set_req                         Set Request flag
        pop     bc
        RET

        endf
```

```
          list a,e
          nlist m
;-----------------------------------------------
;   8 TDE Module
;   Name        : TDE002.SRC
;   Description : Tone decoder module Control Program
;                   - Detect DTMF tone from ic
;                   - Report to MPU
;   Version     1.0
;   Date        : 24 January 1991
;-----------------------------------------------
ptde            equ 0f0h
P8255           equ 90h
P8253           equ 80h
;
P8255A          equ P8255
P8255B          equ P8255+1
P8255C          equ P8255+2
P8255CMD        equ P8255+3
;
PTSAC_CS        equ P8255A
PTSAC_DI        equ P8255C
PTSAC_DTS       equ P8255C
PRING_EN        equ 0eh
PHOOK_STATUS    equ 0fh
;
MAX_COMMAND     equ 240
MAX_DEVICE      equ 8
;
_TDE_Dev_Num    equ 06h
_Dev_num        equ _TDE_Dev_Num
_Chk_Dev        equ 20h
_Set_TSAC       equ 30h
_Set_PCM        equ 31h
_Rx_digit       equ 60h
;-----------------------------------------
;   Inteface to MPU
;-----------------------------------------
INF_H_OUT       equ 0AH         ;MPU Inteface Handshake output
INF_H_IN        equ 09H         ;MPU Inteface Handshake input
INF_DATA        equ 08H         ;MPU Inteface Data
;-----------------------------------------
;   I/O HANDSHAKE SIGNAL
;   <INPUT>
;-----------------------------------------
;                   VALUE
PM_EN           equ 02H         ;PM enable          BIT 1
REQ_ACK         equ 04H         ;Request acknowledge BIT 2
RST_PM          equ 08H         ;Reset PM           BIT 3
;   <OUTPUT>        BIT#
NAK_ACK         equ 0           ;Nak-Ack - data correct;      BIT 0
DATA_RDY        equ 1           ;Data ready                   BIT 1
REQ             equ 2           ;Request from PM              BIT 2
RST_CNT         equ 3           ;Reset inteface address counter BIT 3
;-----------------------------------------
;   ASCII Control code
;
NULL            equ 00H         ; NULL
ACK             equ 06H         ; Ack
NAK             equ 15H         ; Nak
;*****************************
        FILE    TDE002_0.SRC
;*****************************
        org     0h
        di
        im      1
        ld      sp,stack_pt
        ld      b,0
        djnz    $
        jp      MAIN
;
        int_service
;*************************************************
        org     100h
MAIN:       ; Main
;Description : Main Program
;Parameter   : none
;Return Value: none
;Used Reg    :
;Call Sub    : init,scan,anal,exxec_com
```

```
;
        call    init
main1:  call    scan
        call    anal
        call    exec_com
        jp      main1
;-------------------------------------------------------
        File TDE002_1.src
        File TDE002_2.src
        org     2000h
;-------------------------------------------------------
;   Start of RAM
;
H_IN_ADD:   defs    1
H_OUT_ADD:  defs    1
REQ_ADD:    defs    1
MUX_ADD:    defs    1
WR_Index:   defs    2
Rd_Index:   defs    2
_pcm        defs    1
;
        org     2100h
RD_CBUFF:   defs    max_command
        org     2200h
WR_CBUFF:   defs    max_command
temp_cbuff: defs    max_command
        org     2400h
new0:       defs    max_device
old0:       defs    max_device

        org     3fffh
STACK_PT:   defs    1
;
        end
        end
```

```
;-------------------------------------------------------------
;   Tone_Decoder Module
;   Name        : tde002_0.SRC
;   Description : Bios Macro
;   Version     : 1.0
;   Description : - Macro initialize interrupt service routine
;                 - Macro support communicate with MPU
;                 - Macro support generate tone
;   Date        : 19 September 1990
;                 24 January   1991
;-------------------------------------------------------------
Int_service    Macro
;Description : Interrupt service routine
;            :  - Service all interrupt
;Parameter   : none
;Return Value: none
;Used Reg    : all
;Call Sub    : none
;
RST28:  org     28H
        reti
RST30:  org     30H
        reti
RST38:  org     38H             ; Service communication
        di
        jp      Comm_service
        reti
NMI:    org     66H             ; Scan & analysis SSI status
        jp      Nmi_Service
        retn
        endM
;-----------------------------------------
Reset_Cnt     Macro
;Description : Reset counter in MPU-PPU interface
;Parameter   : none
;Return Value: none
;Used Reg    : hl,b,a
;Call Sub    : none

        ld      hl,H_out_ADD
        res     RST_CNT,(hl)
        out     (INF_H_out),a
        ld      b,0AH           ;20 MICROSEC
        djnz    $
        set     RST_CNT,a
        out     (INF_H_out),a
        ld      (hl),a
        endM

;-----------------------------------------
Dis_H_out     Macro  #H_out
;Description : Disable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none
;
        ld      hl,H_out_ADD
        set     #H_out,(hl)
        ld       a,(hl)
        out     (INF_H_out),a
        endM
;-----------------------------------------
En_H_out      Macro  #H_out
;Description : Enable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none

        ld       hl,H_out_ADD
        res      #H_out,(hl)
        ld       a,(hl)
        out      (INF_H_out),a
        endM
;-----------------------------------------
Wr_Inf          Macro
;Description : Copy C_Buff from PPU to MPU
;Parameter   : none
;Return Value: none
```

```
;Used Reg     : a,b,hl
;Call Sub     :
        LOCAL #WR_INF_1
        ld      hl,WR_CBUFF       ;Byte_count
        ld      b,(hl)
        inc     b                 ; add one byte for Byte count
#WR_INF_1: ld   a,(hl)
        out     (INF_DATA),a
        inc     hl
        djnz    #WR_INF_1
        endM
;-----------------------------------------
Add_Wr_Cbuff  macro #data
;Description : Add a data to wr_cbuff
;Parameter   : #data
;Return Value: none
;Used Reg     : a,hl
;Call Sub     : none
        ld      a,#DATA
        ld      (hl),a
        inc     hl
        endM
;-----------------------------------------
Clr_Cbuff       Macro #Cbuff
;Description : Clear Command Buffer
;Parameter   : #cbuff
;Return Value: none
;Used Reg     : a,b,hl
;Call Sub     : none
        LOCAL   #L1
        ld      hl,#CBUFF
        ld      b,MAX_COMMAND+1
        xor     a
#L1:    ld      (hl),a
        inc     hl
        djnz    #L1
        endM
;-----------------------------------------
Set_Req         Macro
;Description : Activate Request signal
;Parameter   : none
;Return Value: none
;Used Reg     : a
;Call Sub     : En_H_out
        ld        a,ACK
        ld        (Req_add),a
        En_H_out REQ
        endM
;-----------------------------------------
Inc_index       macro #index
;Description : Increment rc_index or Wr_index
;Parameter   : #index
;Return Value: none
;Used Reg     : hl
;Call Sub     : none
        push    hl
        ld      hl,(#index)
        inc     hl
        inc     hl
        inc     hl
        ld      (#index),hl
        pop     hl
        endM
;-----------------------------------------
Inc_BC          Macro
;Description : Increment Byte count
;Parameter   : none
;Return Value: none
;Used Reg     : hl
;Call Sub     : none
        ld      a,(wr_cbuff)
        add     a,1
        ld      (wr_cbuff),a
        endM
;-----------------------------------------------------------
        endf
```

```
;-------------------------------------------------------
;   Tone Decoder Module
;   Name        : TDE002_1.SRC
;   Description : Bios
;   Version     : 1.0
;   Description :
;   Date        : 19 September 1990
;                 24 January   1991
;-------------------------------------------------------

NMI_SERVICE: ; NMI Service
;Description : none
;Parameter   : none
;Return Value: none
;Used Reg    : none
;Call Sub    : none
;
        RETI
;-------------------------------------------------
Prg_TSAC:     ;Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value:  B=>  1=Sucess / 0=Timeout
;Used Reg    :  a,bc
;Call Sub    :  none
;
        inc     b               ;B = (0-7) ,increment b before use
stsacs: ld      a,80h
stsac1: rlca
        djnz    stsac1
        out     (PTSAC_CS),a    ;Set CS
        ld      a,c
        ld      b,08h
        rlc     c
        rlc     c
stsac2: ld      a,c
        or      0fdh            ;1111 1101
        out     (PTSAC_DI),a    ;send data to port c low
        dec     a
        out     (PTSAC_DI),a    ;send data to port c hi
        rlc     c
        djnz    stsac2
        ld      b,0
stsac3: inc     b
        jp      z,stsac_to      ;If b>256 Time out
        in      a,(PTSAC_CTS)
        and     10h             ;CTS
        jp      nz,stsac3
        ld      b,1             ;Success
        jp      stsac_e
stsac_to: ld    b,0
stsac_e:  ld    a,0
        out     (PTSAC_CS),a
        ret
;-------------------------------------------------
Other:        ; Other
;Description : Send Unrecognize command to MPU
;Parameter   : none
;Return Value: none
;Used Reg    : bc,de,hl
;Call Sub    : none
;
          push   bc
          ld     c,max_command
          ld     b,0
          LD     HL,RD_CBUFF        ;(hl) ->Source
          LD     DE,WR_CBUFF        ;(de) ->Destination
          LDIR                      ;(bc) ->Number
          set_req                   ;Set Request flag
          pop    bc
          RET
;-----------------------------------------
Clr_wr_index:;Clear write index
;Description : Clear data in wr_index
;Parameter   : none
;Return Value: none
;Used Reg    : a,hl
;Call Sub    :
           ld      hl,wr_cbuff+1
           ld      (Wr_index),hl
```

```
            ld      a,0
            ld      (wr_cbuff),a
            ret
;---------------------------------------
Clr_rd_index::Clear write index
:Description : set data in wr index point to firt command
:Parameter   : none
;Return Value: none
:Used Reg    : hl
;Call Sub    :
            ld      hl,rd_cbuff+1
            ld      (Rd_index),hl
            ret
;-------------------------------------------------------
Comm_service:; Communication service routine
;Description : Communication routine
:               - communicate with MPU
:Parameter   : h_in,h_out
:Return Value: content in C_Buff
:Used Reg    : none
:Call Sub    : none
;
        exx                         ;Save old content
        ex      af,af'
        in      a,(INF_H_in)        ;Check REQ_ACK before check mode
        and     REQ_ACK             ;mask for REQ_ACK
                                    ;  If a=0 then REQ_ACK active
        cp      00H                 ;After compare with Zero
                                    ;
        jp      nz,RD               ;If not ZERO then NO REQ_ACK ,goto rd
        ld      a,(REQ_ADD)         ;Check Request Flag
        cp      ACK                 ;   If ACK then Request to Send data
        jp      nz,NR_NW            ;If no Request then No read.No write
                                    ;Write command to inteface
WR:     WR_INF                      ;Command write complete
        ld      a,NAK               ;Clear request flag
        ld      (REQ_ADD),a
        Dis_H_Out_REQ               ;Disable request signal
        En_H_Out  DATA_RDY          ;Enable data_rdy signal
        call    clr_wr_index        ;Clear wr_index ,clear byte count
        jp      comms_e
                                    ;Read command from interface
RD:     ld      hl,RD_CBUFF
        in      a,(INF_DATA)        ;Read Byte count
        ld      (hl),a
        ld      b,a
        inc     hl
RD_1:   in      a,(INF_DATA)        ;Read Command/Data
        ld      (hl),a
        inc     hl
        djnz    RD_1                ;Repeat Until Byte count=0
RD_E:   En_H_Out  DATA_RDY          ;Enable DATA_RDY
        call    clr_rd_index        ;Clear rd index
        jp      comms_e
                                    ;No read ,No write
NR_NW:  ld      hl,WR_CBUFF         ; This Case -> REQ_ACK = active
        ADD_WR_CBUFF  3             ;       but    NO REQUEST
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        En_H_Out   DATA_RDY
comms_e:Dis_H_Out  DATA_RDY
        ld      hl,WR_CBUFF+1
        ld      (Wr_Index),hl
        ld      hl,rd_Cbuff+1       ; Skip Byte count
        ld      (Rd_index),hl
        ex      af,af'
        exx
        ei
        RETI
            endf
```

```
;-------------------------------------------------------------
;   Tone Decoder module
;   Name        : TDE002_2.SRC
;   Description : Service Function
;   Version     : 1.0
;   Description :
;   Date        : 24 January 1991
;-------------------------------------------------------------
INIT:       ; Initialize
;Description : Initialize 8 CO Module
;Parameter   : none
;Return Value: none
;Used Reg    :
;Call Sub    :
;
        DI
        ld      a,0ffh          ;Clear Handshake output
        out     (INF_H_OUT),a
        ld      (H_OUT_ADD),a   ;Clear Handshake output Status flag
        CLR_CBUFF  WR_CBUFF     ;Clear Write Command Buffer
        call    clr_wr_index    ;Clear wr_index
        CLR_CBUFF  RD_CBUFF     ;Clear Read  Command Buffer
        call    clr_rd_index    ;Clear rd_index
        ld      a,88h           ;Program 8255
        out     (PS255CMD),a
        ld      a,0
        out     (PS255b),a      ;Set pcm line  to 0  *****
        out     (PTSAC_CS),a    ;Clear tsac_cs to 0  *****
        call    init_tsac       ;Clear tx/rx ts
        EI
        RET
;-------------------------------------------------------------
EXEC_COM:   ; Execute Command
;Description : Execute command received from MPU
;Parameter   : none
;Return Value: none
;Used Reg    :
;Call Sub    : chk_dev,set_tsac,set_pcm
;
        DI
        ld      hl,rd_cbuff     ;Read byte count
        ld      a,(hl)
        cp      0               ;If byte count equal 0 then RET
        jp      z,execee
        ld      b,a             ;   else do until last command
        inc     hl
        ld      (rd_index),hl   ;Read index <- first command
exec1:  ld      hl,(rd_index)
        push    bc              ;Save byte count
        ld      a,(hl)          ;Read command
        cp      chk_dev         ;Compare commad
        call    z,chk_dev       ; Jump if correct else skip
        cp      set_tsac
        call    z,set_tsac
        cp      set_pcm
        call    z,set_pcm
        cp      0
        jp      z,exec2
        call    other
        pop     bc
        jp      exece
exec2:  inc_index rd_index      ;Go to next command
        pop     bc
        dec     b
        dec     b               ;Decrement loop counter
        djnz    exec1           ;Repeat until Last command
exece:  ld      hl,rd_cbuff     ;Set rd_index to first command
        ld      (rd_index),hl
        ld      a,0             ;Clear Byte count
        ld      (hl),a
execee: EI
        RET
;-------------------------------------------------------------
Init_tsac:  ;  Initialize tsac
;Description :  Set Tx/Rx time slot to 0
;Parameter   :  NONE
;Return Value:  NONE
;Used Reg    :  hl,de,a
;Call Sub    :  none
;
```

```
        ld      b,8
        ld      c,0ffh
init_tsac1:
        push    bc
        call    prg_tsac
        pop     bc
        djnz    init_tsac1
        ret
;-------------------------------------------------------------
Chk_Dev:        ;Check Device Number to MPU
;Description :  Send Device Number to MPU
;Parameter   :  None
;Return Value:  Content in Wr_Cbuff
;Used Reg    :  HL,A
;Call Sub    :  none
;
        ld      hl,wr_cbuff
        ld      de,temp_cbuff
        ld      a,(hl)              ;Read byte count
        inc     hl                  ;Point to first command
        cp      0
        jp      z,chkdev1
        ld      b,0
        ld      c,a
        ldir
        ld      hl,temp_cbuff
        ld      de,wr_cbuff+1+3
        ld      b,0
        ld      c,a
        ldir
chkdev1: ld     hl,wr_cbuff+1
        Add_Wr_Cbuff    DEV_NUM
        Add_Wr_Cbuff    Null
        Add_Wr_Cbuff    Null
        inc_bc
        inc_index   wr_index
        Set_Reg
        ld      a,0
        ret
;-------------------------------------------------------------
Set_TSAC:     ;Set Time Slot Assigner Circuit
;Description :  Set -both tx/rx timeslot  (00h)
;                   -only Tx              (40h)
;                   -Only Rx              (80h)
;Parameter   : 30h,xxh,yyh  xx=SSI
;                           yy=TS# or (TSAC Command)
;Return Value:  B-> 1=Sucess / 0=Timeout
;Used Reg    :  a,bc,hl
;Call Sub    :  Prg_TSAC
;
        ld      hl,(Rd_index)   ;Command
        inc     hl
        ld      a,(hl)          ;SSI#
        ld      b,a             ;B <- SSI#  (0-7)
        inc     hl
        ld      a,(hl)          ;TSAC command + TS#
        ld      c,a             ;C<- TSAC com + TS#
        call    prg_Tsac
;       inc_index   rd_index
        ld      a,0
        ret
;-------------------------------------------------------------
Set_PCM:      ;Set Tx/Rx PCM line
;Description :  Set -only Tx
;                   -Only Rx
;Parameter   :  31h,xxh,yyh  00xx0000h xx=00-> both
;                                      xx=01-> tx
;                                      xx=10-> rx
;                           yy=data 00tttrrr
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  none
;
        ld      hl,(Rd_Index)
        inc     hl
        ld      a,(hl)      ; xx000000h  xx=00->Both
                            ;            xx=01->tx
                            ;            xx=10->rx
        and     0f0h        ;If a=0 then Tx
        cp      01000000b   ;if Zero -> Tx
```

```
          jr      z,spcmtx
          cp      10000000b    ;if zero -> Rx
          jr      z,spcmrx
          ld      a,(hl)
both:     jr      spcm2
spcmtx:   ld      a,(hl)       ; 00000tttb ->Tx
          and     00000111b
          rlca
          rlca
          rlca
          ld      b,a          ; 00ttt000b
          ld      a,(_pcm)     00xxxrrrb
          and     07h          ; 00000111b
          or      b            ; 00tttrrrb ->A
          jr      spcm2
spcmrx:   ld      a,(hl)       ; 00010rrrb
          and     07h          ; 00000111b
          ld      b,a          ; 00000rrrb
          ld      a,(_pcm) ; 00tttxxxb
          and     00111000b    ; 00111000b
          or      b            ; 00tttrrrb
spcm2:    ld      (_pcm),a
          out     (P82SSB),a
;         inc_index   rd_index
          ld      a,0
          ret
;-------------------------------------------------------------
SCAN:          ; Scan
;Description : Scan data from TDE module
;Parameter   : none
;Return Value: Contents in buffer NEW
;Used Reg    :
;Call Sub    : none
          ld      hl,new0
          ld      b,8
          ld      c,0f0h
scan1:    in      a,(c)
          and     00011111b
          ld      (hl),a
          inc     c
          inc     hl
          djnz    scan1
          RET
;-------------------------------------------------------------
ANAL:          ; Analysis
;Description : Analysis Tone decoder scan input
;                000X nnnn  X=0  : Release
;                           X=1  : Press
;                           nnnn : Data
;Parameter   : none
;Return Value: contents in wr_cbuff ( if key pressed )
;Used Reg    :
;Call Sub    :
          ld      hl,new0
          ld      de,old0
          ld      b,8
anal1:    ld      a,(hl)          ; Get new value
          ld      c,a             ; c <- new state
          and     00010000b       ; if 1->Press then report to MPU
          jp      z,anal_e        ; if 0->Release jump
          ld      a,(de)
          and     00010000b       ; if old_state=release then write couff
          jp      nz,anal_e       ;   else jump to end
          push    hl
          ld      hl,(wr_index)
          add_wr_cbuff   Rx_Digit ; Write command Rx digit
          ld      a,8
          sub     b
          ld      (hl),a          ; Module#
          inc     hl
          ld      a,c             ; c->new state
          and     0fh
          ld      (hl),a          ; Digit Value
          inc     hl
          inc_bc                  ; increment byte count
          inc_index Wr_index
          Set_REQ
          pop     hl
anal_e:   ld      a,(hl)
          ld      (de),a
```

```
        inc     de
        inc     hl
        djnz    anal1
        RET
;-------------------------------------------------------------------
        endf
```

```
            list    a,e
            nlist   m
;-------------------------------------------------------------
;   OPI-Operator Console Interfce Module
;   Name        : OPI002.SRC
;   Description : OPI Main Control Program
;   Version     : 1.0
;   Description :
;   Date        : 3 Dec 1990
;   Last Edit   : 24 January 1991
;-------------------------------------------------------------
ROM             equ 0           ;Rom start address
RAM             equ 2000h       ;Ram start address
;
P8255           equ 90h         ;Port 8255
P8253           EQU 80h
;
P8253C0         EQU P8253
P8253C1         EQU P8253+1
P8253C2         EQU P8253+2
P8253CMD        EQU P8253+3
;
P8255A          equ P8255
P8255B          equ P8255+1
P8255C          equ P8255+2
P8255CMD        equ P8255+3
;
P8251CMD        equ 1
P8251           equ 0
;
PTSAC_CS        equ P8255A      ;Port Chip select
PTSAC_DI        equ P8255C      ;Port Data input
PTSAC_CTS       equ P8255C      ;Port Clear to send
;
MAX_COMMAND     equ 240         ;Maximum command
;
 OPI_Dev_Num    equ 0Ah         ;TSI device number
 Dev_Num        equ  OPI_Dev_Num
 Chk_Dev        equ 20h         ;Check device number command
 Set_TSAC       equ 30h         ;Set TSAC
 Set_PCM        equ 31h         ;Set PCM
 OPI_ONH        equ 41h         ;OPI onhook
 OPI_OFFH       equ 42h         ;OPI offhook
 OPI_FL         equ 43h         ;OPI Flash
 OPI_DIAL       equ 44h         ;OPI Dial Number
 OPI_ENR        equ 37h         ;OPI Enable Ring
 OPI_DISR       equ 38h         ;OPI Disable Ring
 OPI_CNL        equ 47h         ;OPI Ring Cnacel
 OPI_HOLD       equ 0A0h        ;OPI Hold
 MOD_FAIL       equ 26h         ;Module #xx absent
;-------------------------------------------------------------
;   OPI-OPC command
;
ROFF_CODE:      equ 30h         ;Ring off code
RON_CODE:       equ 38h         ;Ring on  code
DIAL_CODE:      equ 40h         ;Dial code
HOLD_CODE:      equ 0a0h        ;Hold code
ON_HOOK_CODE:   equ 50h         ;Onhook code
OFF_HOOK_CODE:  equ 58h         ;Offhook code
FLASH_CODE:     equ 60h         ;Flash code
CANCEL_CODE:    equ 0b0h        ;Cancel code
;-------------------------------------------------------------
;   Inteface to MPU
;-------------------------------------------------------------
INF_H_OUT       equ 0AH         ;MPU Inteface Handshake output
INF_H_IN        equ 09H         ;MPU Inteface Handshake input
INF_DATA        equ 08H         ;MPU Inteface Data
;-------------------------------------------------------------
;   I/O HANDSHAKE SIGNAL
;   <INPUT>
;-------------------------------------------------------------
;                   VALUE
PM_EN           equ 02H         ;PM enable          BIT 1
REQ_ACK         equ 04H         ;Request acknowleage BIT 2
RST_PM          equ 08H         ;Reset PM           BIT 3
;   <OUTPUT>        BIT#
NAK_ACK         equ 0           ;Nak-Ack ->data correct;      BIT 0
DATA_RDY        equ 1           ;Data ready                   BIT 1
REQ             equ 2           ;Request from PM              BIT 2
RST_CNT         equ 3           ;Reset inteface address counter BIT 3
```

```
;-------------------------------------------------------------
;   COI STATE
;
;ONHOOK_ST      equ     0
;RINGDET_ST     equ     1
;OFFHOOK_ST     equ     2
;FLASH_ST       equ     3
;DIALON_ST      equ     4
;DIALOFF_ST     equ     5
;DINTER_ST      equ     6
;-------------------------------------------------------------
;   ASCII Control code
;
NULL            equ 00H         ;NULL
ACK             equ 06H         ;Ack
NAK             equ 15H         ;Nak
;-------------------------------------------------------------
        FILE    OPI002_0.src    ;BIOS MACRO FILE
;-------------------------------------------------------------
;   START MAIN PROGRAM
;
        org     ROM
        ld      sp,stk_pt       ;Set stack pointer
        di
        im      1
        ld      b,0
        djnz    $
        jp      MAIN
;
        int_service             ;Interrupt service macro
;
        org     100h
;-------------------------------------------------------------
MAIN:           ; MAIN
;Description : Main Program
;Parameter   : none
;Return Value: none
;Used Reg    : none
;Call Sub    :
;
        call    init_hs         ;Initialize handshake
        call    clear_buff      ;Clear command buffer
        call    init_tspcm
        call    init_8255       ;initialize 8255
        call    init_8253
        call    init_8251
        ei
main1:
        halt
        call    ANL_OPC         ;Analyze Command Send from OPC
        call    EXEC_COM        ;Execute command send from MPU
        jr      main1
;-------------------------------------------------------------
;Bios file
        file    OPI002_1.src
;Service Function file
        file    OPI002_2.src
;Service Function file
        file    OPI002_3.src
;-------------------------------------------------------------
;   OPI-OPC convert Table
;
TBL_LEN     equ     8
OPI_TBL:    defb     OPI_DISP, OPI_ENP, OPI_DIAL, OPI_ONH, OPI_OFFH
            defb     OPI_FL, OPI_HOLD, OPI_CNL
OPC_TBL:    defb    ROFF_CODE,RON_CODE,DIAL_CODE,ON_HOOK_CODE
            defb    OFF_HOOK_CODE,FLASH_CODE,HOLD_CODE,CANCEL_CODE
;-------------------------------------------------------------
;   Constant value
;   pcm, ts, tone ts table
;
pcm_std:    defb    00001001b   ;PCM#1
;
ts_std:     defb    0ffh
;-------------------------------------------------------------
;   Define value
;
        org     RAM
H_IN_ADD:   defs    1
H_OUT_ADD:  defs    1
```

```
MUX_ADD:    defs    2
REQ_ADD:    defs    1
WR_Index:   defs    2
Rd_Index:   defs    2
RX_BUFF     defs    8
;-------------------------------------------------------------
_pcm        defs    1       ;pcm line address
ts_table:   defs    1
;-------------------------------------------------------------
;   Command Buffer
;
            org     2500h
RD_CBUFF:   defs    max_command
            org     2500h
WR_CBUFF:   defs    max_command
            org     2700h
temp_cbuff  defs    max_command
;-------------------------------------------------------------
;   Timeslot interchange location
;
            org     ram+1fffh
stk_pt:     defs    1
        end
        end
```

```
;-------------------------------------------------------
;   OPI-Operator Console Interface Module
;   Name        : OPI002_0.SRC
;   Description : Bios Macro
;   Version     : 1.0
;   Description : - Macro initialize interrupt service routine
;                 - Macro support communicate with MPU
;   Date        :
;   Last Edit   : 24 January 1991
;-------------------------------------------------------
Int_service    Macro
;Description : Interrupt service routine
;            :  - Service all interrupt
;Parameter   : none
;Return Value: none
;Used Reg    : all
;Call Sub    : none
;
RST28:  org     28H
        reti
RST30:  org     30H
        reti
RST38:  org     38H             ; Service communication
        di
        jp      Comm_service
        reti
NMI:    org     66H             ; Scan & analysis SSI status
        jp      NMI_Service
        retn
        endM
;-----------------------------------------
Reset_Cnt     Macro
;Description : Reset counter in MPU-PPU interface
;Parameter   : none
;Return Value: none
;Used Reg    : hl,b,a
;Call Sub    : none

        ld      hl,H_out_ADD
        res     RST_CNT,(hl)
        out     (INF_H_out),a
        ld      b,0AH           ;20 MICROSEC
        djnz    $
        set     RST_CNT,a
        out     (INF_H_out),a
        ld      (hl),a
        endM

;-----------------------------------------
Dis_H_out     Macro  #H_out
;Description : Disable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none
;
        ld      hl,H_out_ADD
        set     #H_out,(hl)
        ld       a,(hl)
        out     (INF_H_out),a
        endM
;-----------------------------------------
En_H_out     Macro  #H_out
;Description : Enable h_out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none

        ld       hl,H_out_ADD
        res      #H_out,(hl)
        ld       a,(hl)
        out      (INF_H_out),a
        endM
;-----------------------------------------
Wr_Inf          Macro
;Description : Copy C_Buff from PPU to MPU
;Parameter   : none
;Return Value: none
;Used Reg    : a,b,hl
```

```
;Call Sub    :
        LOCAL $WR_INF_1
        ld      hl,WR_CBUFF       ;Byte count
        ld      b,(hl)
        inc     b                 ; add one byte for Byte count
$WR_INF_1: ld   a,(hl)
        out     (INF_DATA),a
        inc     hl
        djnz    $WR_INF_1
        endM
;---------------------------------------
Add_Wr_Cbuff  Macro #data
;Description : Add a data to wr cbuff
;Parameter   : #data
;Return Value: none
;Used Reg    : a,hl
;Call Sub    : none
        ld      a,#DATA
        ld      (hl),a
        inc     hl
        endM
;---------------------------------------
Clr_Cbuff       Macro #Cbuff
;Description : Clear Command Buffer
;Parameter   : #cbuff
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    : none
        LOCAL   #L1
        ld      hl,#CBUFF
        ld      b,MAX_COMMAND+1
        xor     a
#L1:    ld      (hl),a
        inc     hl
        djnz    #L1
        endM
;---------------------------------------
Set_Req         Macro
;Description : Activate Request signal
;Parameter   : none
;Return Value: none
;Used Reg    : a
;Call Sub    : En_h_out
        ld        a,ACK
        ld        (Req_add),a
        En_H_out_REQ
        endM
;---------------------------------------
Inc_index       Macro #index
;Description : Increment rd index or Wr index
;Parameter   : #index
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        push    hl
        ld      hl,(#index)
        inc     hl
        inc     hl
        inc     hl
        ld      (#index),hl
        pop     hl
        endM
;---------------------------------------
Inc_BC          Macro
;Description : Increment Byte count
;Parameter   : none
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        ld      a,(wr_cbuff)
        add     a,3
        ld      (wr_cbuff),a
        endM
```

```
;-----------------------------------------------------------
;   OPI-Operator Console Interrface Module
;   Name        : OPI002_1.SRC
;   Description : BIOS, Service Function
;   Version     : 1.0
;   Description : - Program tsac
;                 - Clear command index
;                 - Convert form ts/pcm -> tsi-add
;   Date        :
;   Last Edit   : 24 January 1991
;-----------------------------------------------------------
Prg_TSAC:     ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: B-> 1=Sucess / 0=Timeout
;Used Reg    : a,bc
;Call Sub    : none
;
        push    bc
        inc     b               ;B = (0-7) ,increment b before use
sstsacs: ld     a,80h
sstsac1: rlca
        djnz    sstsac1
        out     (PTSAC_CS),a    ;Set CS
        ld      a,c
        ld      b,08h
        rlc     c
        rlc     c
sstsac2: ld     a,c
        or      0fdh            ;1111 1101
        out     (PTSAC_DI),a    ;send data to port c low
        dec     a
        nop
        nop
        out     (PTSAC_DI),a    ;send data to port c hi
        rlc     c
        djnz    sstsac2
        ld      b,0
sstsac3: dec        b
        jp      z,sstsac_to     ;If b>256 Time out
        in      a,(PTSAC_CTS)
        bit     4,a             ;CTS
        jp      nz,sstsac3
        pop     bc
        ld      b,1             ;Success
        jp      sstsac_e
sstsac_to:
        ld      hl,(wr_index)
        add_wr_cbuff  MOD FAIL
        pop     bc
        ld      a,b
        ld      (hl),a
        inc_BC
        inc_index wr_index
        set_req
        ld      b,0
sstsac_e:  ld   a,0
        out     (PTSAC_CS),a
        ret
;-----------------------------------------------------------
;Prg_TSAC:    ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: B-> 1=Sucess / 0=Timeout
;Used Reg    : a,bc
;Call Sub    : none
;
;       inc     b               ;B = (0-7) ,increment b before use
;stsacs:    ld       a,80h
;stsac1: rlca
;       djnz    stsac1
;       out     (PTSAC_CS),a    ;Set CS
;       ld      a,c
;       ld      b,08h
;       rlc     c
;       rlc     c
;stsac2:    ld       a,c
;       or      0fdh            ;1111 1101
;       out     (PTSAC_DI),a    ;send data to port c low
;       dec     a
```

```
;       out     (PTSAC_DI),a    ;send data to port c hi
;       rlc     c
;       djnz    stsac2
;       ld      b,0
;stsac3: inc         b
;       jp      z,stsac_to      ;If b>256 Time out
;       in      a,(PTSAC_CTS)
;       and     10h             ;CTS
;       jp      nz,stsac3
;       ld      b,1             ;Success
;       jp      stsac_e
;stsac_to: ld    b,0
;stsac_e:  ld    a,0
;       out     (PTSAC_CS),a
;       ret
;-------------------------------------------------------------
Clr_wr_index:;  Clear write Index
;Description :  Clear write Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,hl
;Call Sub    :  none
;
        ld      hl,wr_cbuff+1
        ld      (Wr_index),hl
        ld      a,0
        ld      (wr_cbuff),a
        ret
;-------------------------------------------------------------
Clr_rd_index:;  Clear read Index
;Description :  Clear read Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl
;Call Sub    :  none
;
        ld      hl,rd_cbuff+1
        ld      (Rd_index),hl
        ret
;-------------------------------------------------------------
Cnv_TS:         :  Convert timeslot
;Description :  Convert format ts/pcm to tsi add
;Parameter   :  B=ts#  ,  C=PCM#
;Return Value:  a=TSI ADD
;Used Reg    :  A,BC
;Call Sub    :  None
;
        ld      a,0
        sla     a
        sla     a
        sla     a
        or      c
        ret

        ENDF
```

```
;-------------------------------------------------------
;   OPI-Operator Console Interface Module
;   Name        : OPI002 2.SRC
;   Description : Service Function
;                   - Execute command
;                   - NMI Service
;                   - Commuication service
;                   - Initialize
;   Version     : 1.0
;   Date        :
;   Last Edit   : 24 January 1991
;-------------------------------------------------------
Exec_com:     ;  Exec command
;Description :  Execute command which sent from MPU
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  Chk_dev,Clr_tsi,C_tone,Ex_ts,Test
;
        di
        ld      hl,rd_cbuff     ;Read byte count
        ld      a,(hl)
        cp      0               ;If byte count equal 0 then RET
        jp      z,execee
        ld      b,a             ;   else do until last command
        inc     hl
        ld      (rd_index),hl   ;Read index -- first command
exec1:  ld      hl,(rd_index)
        push    bc
        ld      a,(hl)          ;Read command
        cp      _chk_dev        ;Check Device Number
        call    z,chk_dev
        cp      _set_pcm        ;Set PCM
        call    z,set_pcm
        cp      _set_tsac       ;Set TSAC
        call    z,set_tsac
        cp      _opi_dial       ;Set 7 Segment Display
        call    z,opi_dial
        cp      0
        jp      z,exec2
        push    hl
        ld      hl,opi_tbl+tbl_len-1
        ld      de,opc_tbl+tbl_len-1
        ld      b,tbl_len
exe1:   cp      (hl)
        jp      z,exe2
        dec     hl
        dec     de
        djnz    exe1
        pop     hl
        push    hl
        call    command_err
        pop     hl
        jp      exe3
exe2:   pop     hl
        inc     hl
        ld      a,(de)
        and     11111000b
        ld      b,a
        ld      a,(hl)
        or      b
        ld      c,a
sendopc:
        in      a,(p8251cmd)
        bit     0,a
        jp      z,sendopc
        ld      a,c
        out     (P8251),a
exe3:
        cp      0
        jp      z,exec2
        call    other
        pop     bc
        jp      exece
exec2:  inc_index   rd_index    ;Go to next command
        pop     bc
        dec     b
        dec     b               ;Decrement loop counter
        djnz    exec1            ;Repeat until Last command
exece:  ld      hl,rd_cbuff     ;Set rd_index to first command
```

```
        ld      (rd_index),hl
        ld      a,0
        ld      (hl),a
execee: EI
        ret
;-------------------------------------------------------
NMI_Service: ;  Non maskable Interrupt service routine
;Description :  Scan OPC
;Parameter   :  none
;Return Value:  none
;Used Reg    :  all
;Call Sub    :  inc_cnt2, anl_tone
;
        exx
        ex      af,af'
        push    ix
        push    iy
        di
        call    SCAN_OPC2
        pop     iy
        pop     ix
        ex      af,af'
        exx
        ei
        retn
;-------------------------------------------------------
Comm_service:: Communication service routine
;Description : Communication routine
;               - communicate with MPU
;Parameter   : h_in,h_out
;Return Value: content in C_Buff
;Used Reg    : none
;Call Sub    : none
;
        exx                             ;Save old content
        ex      af,af'
        in      a,(INF_H_in)            ;Check REQ_ACK before check mode
        and     REQ_ACK                 ;mask for REQ_ACK
                                        ;  If a=0 then REQ_ACK active
        cp      00H                     ;After compare with Zero
                                        ;
        jp      nz,RD                   ;If not ZERO then NO REQ_ACK ,goto rd
        ld      a,(REQ_ADD)             ;Check Request Flag
        cp      ACK                     ;   If ACK then Request to Send data
        jp      nz,NR_NW                ;If no Request then No_read,no_write
                                        ;Write command to inteface
WR:     WR_INF                          ;Command write complete
        ld      a,NAK                   ;Clear request flag
        ld      (REQ_ADD),a
        Dis_H_Out_REQ                   ;Disable request signal
        En_H_Out  DATA_RDY              ;Enable data_rdy signal
        call    clr_wr_index            ;Clear wr_index ,clear byte count
        jp      comms_e
                                        ;Read command from interface
RD:     ld      hl,RD_CBUFF
        in      a,(INF_DATA)            ;Read Byte count
        ld      (hl),a
        ld      b,a
        inc     hl
RD_1:   in      a,(INF_DATA)            ;Read Command/Data
        ld      (hl),a
        inc     hl
        djnz    RD_1                    ;Repeat Until Byte count=0
RD_E:   En_H_Out  DATA_RDY              ;Enable DATA_RDY
        call    clr_rd_index            ;Clear rd_index
        jp      comms_e
                                        ;No read ,No write
NR_NW:  ld      hl,WR_CBUFF             ; This Case -> REQ_ACK = active
        ADD_WR_CBUFF  3                 ;       but    NO REQUEST
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        En_H_Out   DATA_RDY
comms_e:Dis_H_Out  DATA_RDY
        ld      hl,WR_CBUFF+1
        ld      (Wr_Index),hl
        ld      hl,rd_Cbuff+1           ; Skip Byte count
        ld      (Rd_index),hl

;       call    exec_com
```

```
        SET_REQ
        ld      a,0
        ld      (PX_SUFF),a
anlopce:
        ei
        ret
;-------------------------------------------------------------
Command_ERR:  ;  Command error
;Description :  Command error
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  delay
;
        call    delay
        ret
;-------------------------------------------------------------
Init_8255:    ;  Init 8255
;Description :  Set up 8255
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,88h           ;Program 8255
        out     (PS255CMD),a
        out     (PS255CMD),a
        ld      a,0
        ret
;-------------------------------------------------------------
init_tspcm:   ;  Init TS/PCM
;Description :  Initialize PCM/TSAC
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc
;Call Sub    :  PRG_TSAC
;
        ld      a,(ts_std)
        ld      (ts_table),a
        ld      a,(pcm_std)
        ld      (_pcm),a
        out     (PS255b),a      ;Set pcm line  to default
        out     (PS255b),a
        ld      a,(_pcm)
        out     (PS255B),a      ;Set pcm to default value
        ld      hl,ts_table
        ld      b,0
        ld      c,(hl)          ; c -> TS#
        call    prg_tsac
        ret
;-------------------------------------------------------------
init_hs:      ;  Initialize Handshake
;Description :  initialize handshake
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    : none
;
        ld      a,0ffh          ;Clear Handshake output
        out     (INF_H_OUT),a
        ld      (H_OUT_ADD),a   ;Clear Handshake output Status flag
        ld      a,nak
        ld      (REQ_ADD),a     ;Clear Request flag
        ret
;-------------------------------------------------------------
clear_buff:   ;  Clear buffer
;Description :  Clear wr_cbuff, rd_cbuff
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  macro clear_buff, clr_rd_index, clr_wr_index
;
        clr_cbuff   rd_cbuff
        clr_cbuff   wr_cbuff
        call    clr_rd_index
        call    clr_wr_index
        ret
;-------------------------------------------------------------
init_8253:    ;  Program 8253
;Description :  Program 8253 as Bit rate gen
```

```
;Parameter    :  NONE
;Return Value:  NONE
;Used Reg     :  a
;Call Sub     :  none
;
        ld      a,37h
        out     (P8253cmd),a
        call    delay
        ld      a,8
        out     (P8253c0),a
        call    delay
        ld      a,0
        out     (P8253c0),a
        call    delay
        ret
;-----------------------------------------------------------------
init_8251:    ; Program 8251
;Description :
;Parameter    :  NONE
;Return Value:  NONE
;Used Reg     :  a
;Call Sub     :  none
;
        ld      a,0
        out     (P8251cmd),a
        call    delay
        out     (P8251cmd),a
        call    delay
        out     (P8251cmd),a
        call    delay
        ld      a,40h
        out     (P8251cmd),a
        call    delay
        ld      a,4eh
        out     (p8251cmd),a
        call    delay
        ld      a,37h
        out     (p8251cmd),a
        ld      a,0
        ld      (RX_BUFF),a
        ret
;-----------------------------------------------------------------
delay:        ; DELAY
;Description :  Delay
;Parameter    :  none
;Return Value:  none
;Used Reg     :  b
;Call Sub     :  none
;
        ld      b,0
        djnz    $
        ret
;-----------------------------------------------------------------
        endf
```

```
;-----------------------------------------------------------
;   OPI-Operator Console Interface module
;   Name        : OPI002_3.SRC
;   Description : Service Function
;                   - opi_dial,chk_dev, other
;   Version     : 1.0
;   Description :
;   Date        :
;   Last Edit   : 24 January 1991
;-----------------------------------------------------------
Chk_Dev:        ;Check Device Number to MPU
;Description :  Send Device Number to MPU
;Parameter   :  None
;Return Value:  Content in Wr_Cbuff
;Used Reg    :  HL,A
;Call Sub    :  none
;
        ld      hl,wr_cbuff
        ld      de,temp_cbuff
        ld      a,(hl)              ;Read Data Count
        inc     hl                  ;Point to first command
        cp      0
        jp      z,chkdev1
        ld      b,0
        ld      c,a
        ldir
        ld      hl,temp_cbuff
        ld      de,wr_cbuff+1+3
        ld      b,0
        ld      c,a
        ldir
chkdev1: ld     hl,wr_cbuff+1
        Add Wr_Cbuff    DEV NUM
        Add Wr_Cbuff    Null
        Add Wr_Cbuff    Null
        inc bc
        inc index   wr_index
        Set Req
        ld      a,0
        ret
;-----------------------------------------------------------
Set_TSAC:       ;Set Time Slot Assigner Circuit
;Description :  Set -both tx/rx timeslot  (00h)
;                   -only Tx              (40h)
;                   -Only Rx              (80h)
;Parameter   : 30h,xxh,yyh  xx=SSI
;                           yy=TS# or (TSAC Command)
;Return Value:  B-> 1=Sucess / 0=Timeout
;Used Reg    :  a,bc,hl
;Call Sub    :  Prg_TSAC
;
        ld      hl,(Pd_index)   ;Command
        inc     hl
        ld      a,(hl)          ;SSI#
        ld      b,a             ;B <- SSI#  (0-7)
        inc     hl
        ld      a,(hl)          ;TSAC command + TS#
        ld      c,a             ;C <- TSAC com + TS#
        call    prg_Tsac
;       inc index   rd_index
        ld      a,0
        ret
;-----------------------------------------------------------
Set_PCM:        ;Set Tx/Rx PCM line
;Description :  Set -only Tx
;                   -Only Rx
;Parameter   :  31h,xxh,yyh  xx000000h xx=00-  both
;                                      xx=01-  tx
;                                      xx=10-  rx
;                           yy=data 00ttrrrr
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  none
;
        ld      hl,(Pd_Index)
        inc     hl
        ld      a,(hl)      ; xx000000h  xx=00->both
                            ;            xx=01->tx
                            ;            xx=10->rx
        and     0f0h        ;If a=0 then Tx
```

```
        cp      01000000b       ;if Zero -> Tx
        jr      z,spcmtx
        cp      10000000b       ;if zero -> Rx
        jr      z,spcmrx
        ld      a,(hl)
both:   jr      spcm2
spcmtx: ld      a,(hl)          ; 00000rrrb -> Tx
        and     00000111b
        rlca
        rlca
        rlca
        ld      b,a             ; 00rrr000b
        ld      a,( pcm)    00xxxrrrb
        and     07h             ; 00000111b
        or      b               ; 00rrrrrrb -> A
        jr      spcm2
spcmrx: ld      a,(hl)          ; 00010rrrb
        and     07h             ; 00000111b
        ld      b,a             ; 00000rrrb
        ld      a,( pcm) ; 00tttxxxb
        and     00111000b       ; 00111000b
        or      b               ; 00tttrrrb
spcm2:  ld      ( pcm),a
        out     (PB2SSB),a
;       inc index   rd_index
        ld      a,0
        ret
;-----------------------------------------------------------------
OPI DIAL:       ;
;Description : SET 7 Segment on OPI
;Parameter   :
;Return Value:
;Used Reg    :
;Call Sub    :
;
        ld      hl,(Rd_index)   ;Command
        inc     hl
        ld      a,(hl)          ;OPI#
        ld      b,a             ;B <- OPI#  (0-7)
        inc     hl
        ld      a,(hl)
        ld      c,a             ;C - DIGIT
        ld      a,DIAL_CODE
        or      b
        ld      d,a
        call    delay
        call    delay
        call    delay
txf:
        in      a,(pB2S1cmd)
        bit     0,a
        jp      z,txf
        ld      a,d
        out     (PB2S1),a
        call    delay
        call    delay
        call    delay
txs:
        in      a,(pB2S1cmd)
        bit     0,a
        jp      z,txs
        ld      a,c
        out     (PB2S1),a
        ld      a,0
        ret
;-----------------------------------------------------------------
OTHER:          ; OTHER
;Description :  Send Unrecognize command back to MPU (TEST)
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,de,bc
;Call Sub    :  set_req
;
        push    de
        ld      hl,(wr_index)
        add wr_cbuff 6h
        ld      de,(rd_cbuff)
        ld      b,6
o1:     ld      a,(de)
        ld      (hl),a
```

```
        djnz    o1
        INC BC
        INC INDEX wr index
        set rel
        pop     de
        ret
;-----------------------------------------------------------

        endf
```

```
            list    a.e
            nlist   m
;-----------------------------------------------------------------
;   DCI-Data Card Interface
;   Name        : DCI001.SRC
;   Description : DCI-Data Communication Control Program
;   Version     : 1.0
;   Date        : 3 Dec 1990
;   Last Edit   : 24 January 1991
;-----------------------------------------------------------------
ROM             equ 0           ;Rom start address
RAM             equ 2000h       ;Ram start address
;
P8255           equ 90h         ;Port 8255
P8253           EQU 80h
PNMI            EQU 0FH
;
P8253C0         EQU P8253       ;8253 Channel 1
P8253C1         EQU P8253+1     ;8253 Channel 2
P8253C2         EQU P8253+2     ;8253 Channel 3
P8253CMD        EQU P8253+3     ;8253 Command
;
P8255A          equ P8255
P8255B          equ P8255+1
P8255C          equ P8255+2
P8255CMD        equ P8255+3
;
ASY0_COMM       equ 1           ;8251 Command
ASY0_DATA       equ 0           ;8251 Data
ASY1_COMM       equ 3           ;8251 Command
ASY1_DATA       equ 2           ;8251 Data
;
PPCM0_WR        equ 4           ;PCM WR/RD port
PPCM0_RD        equ 5
PPCM1_WR        equ 6
PPCM1_RD        equ 7
;
PTSAC_CS        equ P8255A      ;Port Chip select
PTSAC_DI        equ P8255C      ;Port Data input
PTSAC_CTS       equ P8255C      ;Port Clear to send
;
MAX_COMMAND     equ 240         ;Maximum command
;
 DCI_Dev_Num    equ 08h         ;TSI device number
 Dev_Num        equ  DCI_Dev_Num
 Chk_Dev        equ 20h         ;Check device number command
 Set_TSAC       equ 30h         ;Set TSAC
 Set_PCM        equ 31h         ;Set PCM
 MOD_FAIL       equ 26h         ;Module #xx absent
;
time_l          equ #24h        ;8253 time const low bit
time_h          equ 0           ;8253 time const hi bit
;-----------------------------------------------------------------
;   Inteface to MPU
;
INF_H_OUT       equ 0AH         ;MPU Inteface Handshake output
INF_H_IN        equ 09H         ;MPU Inteface Handshake input
INF_DATA        equ 0BH         ;MPU Inteface Data
;-----------------------------------------------------------------
;   I/O HANDSHAKE SIGNAL
;   <INPUT>
;
;                   VALUE
PM_EN           equ 02H         ;PM enable              BIT 1
REQ_ACK         equ 04H         ;Request acknowledge BIT 2
RST_PM          equ 08H         ;Reset PM               BIT 3
;   <OUTPUT>        BIT#
NAK_ACK         equ 0           ;Nak-Ack -=data correct;        BIT 0
DATA_RDY        equ 1           ;Data ready                     BIT 1
REQ             equ 2           ;Request from PM                BIT 2
RST_CNT         equ 3           ;Reset inteface address counter BIT 3
;-----------------------------------------------------------------
;   ASCII Control code
;
NULL            equ 00H         ;NULL
ACK             equ 06H         ;Ack
NAK             equ 15H         ;Nak
;-----------------------------------------------------------------
        FILE    DCI001_0.src    ;BIOS MACRO FILE
;-----------------------------------------------------------------
```

```
;   START MAIN PROGRAM
;
        org     ROM
        ld      sp,stk_pt       ;Set stack pointer
        xor     a
        cpl
        out     (PNMI),a
        di
        im      1
        ld      b,0
        djnz    $
        jp      MAIN
;
        int_service             ;Interrupt service macro
;
        org     100h
;-----------------------------------------------------------
MAIN:           ; MAIN
;Description    : main Program
;Parameter      : none
;Return Values  : none
;Used Reg       : none
;Call Sub       :
;
        call    init_hs         ;Initialize handshake
        call    clear_buff      ;Clear command buffer
        call    init_tspcm
        call    init_8255       ;initialize 8255
        call    init_8253
        call    init_8251
        call    set_NMI
        ei
main1:
        halt
        call    exec_com        ;Execute command send from MPU
        jr      main1
;-----------------------------------------------------------
;Bios file
        file    DCI001_1.src
;Service Function file
        file    DCI001_2.src
;Service Function file
        file    DCI001_3.src
;-----------------------------------------------------------
;   Constant value
;   pcm, ts, tone ts table
;
pcm_std:    defb    00010010b   ;PCM#2
;
ts_std:     defb    0ffh,0ffh
;-----------------------------------------------------------
;   Define value
;
        org     RAM
H_IN_ADD:   defs    1
H_OUT_ADD:  defs    1
MUX_ADD:    defs    2
REQ_ADD:    defs    1
WR_Index:   defs    2
Rd_Index:   defs    2
ledcnt      defs    1
ledstu      defs    1
nmi_stu     defs    1
;-----------------------------------------------------------
_pcm        defs    1       ;pcm line address
ts_table:   defs    2
;-----------------------------------------------------------
;   Command Buffer
;
            org     2500h
RD_CBUFF:   defs    max_command
            org     2600h
WR_CBUFF:   defs    max_command
            org     2700h
temp_cbuff  defs    max_command
;-----------------------------------------------------------
;   Timeslot interchange location
;
            org     ram+1fffh
stk_pt:     defs    1
```

```
;-------------------------------------------------------
;   DCI-Data Communication Interface Module
;   Name        : DCI001_0.SRC
;   Description : Bios Macro
;   Version     : 1.0
;   Description : - Macro initialize interrupt service routine
;                 - Macro support communicate with MPU
;   Date        :
;   Last Edit   : 24 January 1991
;-------------------------------------------------------
Int service    Macro
;Description : Interrupt service routine
;            :  - Service all interrupt
;Parameter   : none
;Return Value: none
;Used Reg    : all
;Call Sub    : none
;
RST28:  org     28H
        reti
RST30:  org     30H
        reti
RST38:  org     38H             ; Service communication
        di
        jp      Comm service
        reti
NMI:    org     66H             ; Scan & analysis SSI status
        jp      NMI Service
        retn
        endM
;-------------------------------------
Reset Cnt     macro
;Description : Reset counter in MPU-PPU interface
;Parameter   : none
;Return Value: none
;Used Reg    : hl,b,a
;Call Sub    : none

        ld      hl,H_out ADD
        res     RST CNT,(hl)
        out     (INF H out),a
        ld      b,0AH           ;20 MICROSEC
        djnz    $
        set     RST CNT,a
        out     (INF H out),a
        ld      (hl),a
        endM

;-------------------------------------
Dis H out     macro  #H out
;Description : Disable h out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none
;
        ld      hl,H out ADD
        set     #H out,(hl)
        ld       a,(hl)
        out     (INF H out),a
        endM
;-------------------------------------
En H out     macro  #H out
;Description : Enable h out
;Parameter   : Handshake signal number
;Return Value: none
;Used Reg    : hl,a
;Call Sub    : none

        ld       hl,H out ADD
        res      #H out,(hl)
        ld       a,(hl)
        out      (INF H out),a
        endM
;-------------------------------------
Wr Inf         macro
;Description : Copy C Buff from PPU to MPU
;Parameter   : none
;Return Value: none
;Used Reg    : a,b,hl
```

```
;Call Sub    :
        LOCAL #WR_INF_1
        ld      hl,WR_CBUFF       ;Byte count
        ld      b,(hl)
        inc     b                 ; add one byte for Byte count
#WR_INF_1: ld   a,(hl)
        out     (INF_DATA),a
        inc     hl
        djnz    #WR_INF_1
        endM
;-----------------------------------------
Add_Wr_Cbuff  Macro #data
;Description : Add a data to wr cbuff
;Parameter   : #data
;Return Value: none
;Used Reg    : a,hl
;Call Sub    : none
        ld      a,#DATA
        ld      (hl),a
        inc     hl
        endM
;-----------------------------------------
Clr_Cbuff       macro #Cbuff
;Description : Clear Command Buffer
;Parameter   : #cbuff
;Return Value: none
;Used Reg    : a,b,hl
;Call Sub    : none
        LOCAL   #L1
        ld      hl,#CBUFF
        ld      b,MAX_COMMAND+1
        xor     a
#L1:    ld      (hl),a
        inc     hl
        djnz    #L1
        endM
;-----------------------------------------
Set_Req         macro
;Description : Activate Request signal
;Parameter   : none
;Return Value: none
;Used Reg    : a
;Call Sub    : En_H_out
        ld        a,ACK
        ld        (Req_add),a
        En_H_out REQ
        endM
;-----------------------------------------
Inc_index       Macro #index
;Description : Increment rd index or wr index
;Parameter   : #index
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        push    hl
        ld      hl,(#index)
        inc     hl
        inc     hl
        inc     hl
        ld      (#index),hl
        pop     hl
        endM
;-----------------------------------------
Inc_BC          Macro
;Description : Increment Byte count
;Parameter   : none
;Return Value: none
;Used Reg    : hl
;Call Sub    : none
        ld      a,(wr_cbuff)
        add     a,3
        ld      (wr_cbuff),a
        endM
;-----------------------------------------------------------
        endF
```

```
;-------------------------------------------------------------
;   DCI-Data Communication Interface Module
;   Name        : DCI001 1.SRC
;   Description : BIOS, Service Function
;   Version     : 1.0
;   Description : - Program tsac
;                 - Clear command index
;   Date        :
;   Last Edit   : 24 January 1991
;-------------------------------------------------------------
Prg_TSAC:     ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: S->  1=Sucess / 0=Timeout
;Used Reg    :  a,bc
;Call Sub    :  none
;
        push    bc
        inc     b               ;B = (0-7) ,increment b before use
sstsacs:    ld      a,80h
sstsac1: rlca
        djnz    sstsac1
        out     (PTSAC_CS),a    ;Set CS
        ld      a,c
        ld      b,08h
        rlc     c
        rlc     c
sstsac2:    ld      a,c
        or      0fdh            ;1111 1101
        out     (PTSAC_DI),a    ;send data to port c low
        dec     a
        nop
        nop
        out     (PTSAC_DI),a    ;send data to port c hi
        rlc     c
        djnz    sstsac2
        ld      b,0
sstsac3: dec        b
        jp      z,sstsac_to     ;If b>256 Time out
        in      a,(PTSAC_CTS)
        bit     4,a             ;CTS
        jp      nz,sstsac3
        pop     bc
        ld      b,1             ;Success
        jp      sstsac_e
sstsac_to:
        ld      hl,(wr_index)
        add_wr_cbuff _MOD_FAIL
        pop     bc
        ld      a,b
        ld      (hl),a
        inc_BC
        inc_index wr_index
        set_req
        ld      b,0
sstsac_e:  ld   a,0
        out     (PTSAC_CS),a
        ret
;-------------------------------------------------------------
;Prg_TSAC:     ; Program Time Slot Assigner Circuit
;Description : Shift 8 bit Command to TSAC
;Parameter   : B=SSI# (0-7), C=Tsac command
;Return Value: S->  1=Sucess / 0=Timeout
;Used Reg    :  a,bc
;Call Sub    :  none
;
;       inc     b               ;B = (0-7) ,increment b before use
;stsacs:    ld      a,80h
;stsac1: rlca
;       djnz    stsac1
;       out     (PTSAC_CS),a    ;Set CS
;       ld      a,c
;       ld      b,08h
;       rlc     c
;       rlc     c
;stsac2:    ld      a,c
;       or      0fdh            ;1111 1101
;       out     (PTSAC_DI),a    ;send data to port c low
;       dec     a
;       out     (PTSAC_DI),a    ;send data to port c hi
```

```
;       rlc     c
;       djnz    stsac2
;       ld      b,0
;stsac3: inc         b
;       jp      z,stsac_to      ;If b=256 Time out
;       in      a,(PTSAC_CTS)
;       and     10h             ;CTS
;       jp      nz,stsac3
;       ld      b,1             ;Success
;       jp      stsac_e
;stsac_to: ld    b,0
;stsac_e:  ld    a,0
;       out     (PTSAC_CS),a
;       ret
;-------------------------------------------------------------------
Clr_wr_index::  Clear write Index
;Description :  Clear write Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,hl
;Call Sub    :  none
;
        ld      hl,wr_cbuff+1
        ld      (Wr_index),hl
        ld      a,0
        ld      (wr_cbuff),a
        ret
;-------------------------------------------------------------------
Clr_rd_index::  Clear read Index
;Description :  Clear read Index
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl
;Call Sub    :  none
;
        ld      hl,rd_cbuff+1
        ld      (Rd_index),hl
        ret

        ENDf
```

```
;-------------------------------------------------------
;   DCI-Data Communication Interface module
;   Name        : DCI001 2.SRC
;   Description : Service Function
;                   - Execute command
;                   - NMI Service
;                   - Communication service
;                   - Initialize
;   Version     : 1.0
;   Date        :
;   Last Edit   : 24 January 1991
;-------------------------------------------------------
Exec_com:    ;  Exec command
;Description :  Execute command which sent from MPU
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  Chk_dev,Clr_tsi,C_tone,Ex_ts,Test
;
        di
        ld      hl,rd_cbuff     ;Read byte count
        ld      a,(hl)
        cp      0               ;if byte count equal 0 then RET
        jp      z,execee
        ld      b,a             ;   else do until last command
        inc     hl
        ld      (rd_index),hl   ;Read index  - first command
execi:  ld      hl,(rd_index)
        push    bc
        ld      a,(hl)          ;Read command
        cp      _chk_dev
        call    z,chk_dev
        cp      _set_pcm
        call    z,set_pcm
        cp      _set_tsac
        call    z,set_tsac
        cp      0
        jp      z,exec2
        push    hl
        call    command_err
        pop     hl
exec2:  inc_index   rd_index    ;Go to next command
        pop     bc
        dec     b
        dec     b               ;Decrement loop counter
        djnz    execi            ;Repeat until last command
exece:  ld      hl,rd_cbuff     ;Set rd_index to first command
        ld      (rd_index),hl
        ld      a,0
        ld      (hl),a
execee: EI
        ret
;-------------------------------------------------------
NMI_Service: :  Non Maskable Interrupt service routine
:Description :  Receive/Transmit data From ASY1/ASY2
;Parameter   :  none
;Return Value:  none
;Used Reg    :  all
;Call Sub    :  inc_cnt2, ani_tone
;
        di
        push    af
        push    bc
        push    de
        push    hl
        xor     a
        cpl
        out     (PNMI),a
PCM0_ASY0:
        in      a,(PPCM0_RD)
        cp      0ffh
        jr      z,PCM1_ASY1
;******
        out     (0ffh),a
;******
        out     (ASY0_DATA),a
PCM1_ASY1:
        in      a,(PPCM1_RD)
        cp      0ffh
        jr      z,ASY0_PCM0
```

```
        out     (ASY1_DATA),a
ASY0_PCM0:
        in      a,(ASY0_COmm)
        bit     1,a
        ld      a,0ffh
        jr      z,NO_RX0
        in      a,(ASY0_DATA)
no_rx0: out     (PPCm0_WR),a
ASY1_PCM1:
        in      a,(ASY1_COmm)
        bit     1,a
        ld      a,0ffh
        jr      z,NO_RX1
        in      a,(ASY1_DATA)
no_rx1: out     (PPCM1_WR),a

;       ld      a,(LEDcnt)
;       dec     a
;       ld      (ledcnt),a
;       cp      0
;       jp      nz,lle
;       ld      a,(ledstu)
;       inc     a
;       ld      (ledstu),a
;       out     (0FFh),a
;       out     (ASY0_DATA),a

lle:
        xor     a
        out     (PNMI),a
        pop     hl
        pop     de
        pop     bc
        pop     af
        ei
        reti
;---------------------------------------------------------
Comm_service:: Communication service routine
;Description : Communication routine
;              - communicate with MPU
;Parameter   : h_in,h_out
;Return Value: content in C_Buff
;Used Reg    : none
;Call Sub    : none
;
        exx                             ;Save old content
        ex      af,af'
        xor     a
        cpl
        out     (PNMI),a

        in      a,(INF_H_in)            ;Check REQ_ACK before check mode
        and     REQ_ACK                 ;Mask for REQ_ACK
                                        ;  If a=0 then REQ_ACK active
        cp      00H                     ;After compare with Zero
                                        ;
        jp      nz,RD                   ;If not ZERO then NO REQ_ACK ,goto rd
        ld      a,(REQ_ADD)             ;Check Request Flag
        cp      ACK                     ;   If ACK then Request to Send data
        jp      nz,NR_NW                ;If no Request then No_read,No write
                                        ;Write command to inteface
WR:     WR_INF                          ;Command write complete
        ld      a,NAK                   ;Clear request flag
        ld      (REQ_ADD),a
        Dis_H_Out_REQ                   ;Disable request signal
        En_H_Out_DATA_RDY               ;Enable data rdy signal
        call    clr_wr_index            ;Clear wr_index ,clear byte count
        jp      comms_e
                                        ;Read command from interface
RD:     ld      hl,RD_CBUFF
        in      a,(INF_DATA)            ;Read Byte count
        ld      (hl),a
        ld      b,a
        inc     hl
RD_1:   in      a,(INF_DATA)            ;Read Command/Data
        ld      (hl),a
        inc     hl
        djnz    RD_1                    ;Repeat Until Byte count=0
RD_E:   En_H_Out_DATA_RDY               ;Enable DATA RDY
        call    clr_rd_index            ;Clear rd_index
```

```
        JP      comms_e
                                        ;No read ,No write
NR_NW:  ld      hl,WR_CBUFF             ; This Case -> REQ_ACK = active
        ADD_WR_CBUFF  3                 ;       but     NO REQUEST
        ADD_WR_CBUFF  NULL
        ADD_WR_CBUFF  NULL
        ADD_WR_CSUFF  NULL
        En_H_Out   DATA_RDY
comms_e:Dis_H_Out  DATA_RDY
        ld      hl,WR_CBUFF+1
        ld      (Wr_Index),hl
        ld      hl,rd_Cbuff+1           ; Skip Byte count
        ld      (Rd_index),hl
        xor     a
        out     (PNMI),a
        ex      af,af'
        exx
        ei
        RETI
;-----------------------------------------------------------
;ASY_PCM:    ;
;Description :
;Parameter   :
;Return Value:
;Used Reg    :
;Call Sub    :
;
;
        ret
;-----------------------------------------------------------
Command_ERR: ;  Command error
;Description :  Command error
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  delay
;
        call    delay
        ret
;-----------------------------------------------------------
Init_8255:   ;  init 8255
;Description :  Set up 8255
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,88h           ;Program 8255
        out     (P8255CMD),a
        out     (P8255CMD),a
        ld      a,0
        ret
;-----------------------------------------------------------
init_tspcm:  ;  Init TS/PCM
;Description :  Initialize TS/PCM
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  prg_tsac
;
        ld      a,(ts_std)
        ld      (ts_table),a
        ld      a,(ts_std+1)
        ld      (ts_table+1),a
        ld      a,(pcm_std)
        ld      (_pcm),a
        out     (P8255a),a      ;Set pcm line  to default
        out     (P8255b),a
        ld      a,(_pcm)
        out     (P8255B),a      ;Set pcm to default value
        ld      hl,ts_table
        ld      b,0
        ld      c,(hl)          ; c -> TS#
        call    prg_tsac
        call    delay
        inc     hl
        ld      b,1
        ld      c,(hl)
        call    prg_tsac
        ret
```

```
;-------------------------------------------------------
init_hs:      ;  Initialize Handshake
;Description :  initialize handshake
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,0ffh          ;Clear Handshake output
        out     (INF_H_OUT),a
        ld      (H_OUT_ADD),a   ;Clear Handshake output Status flag
        ld      a,nak
      . ld      (REQ_ADD),a     ;Clear Request flag
        ld      a,0
        ld      (ledcnt),a
        ld      (ledstu),a
        out     (0ffh),a
        ret
;-------------------------------------------------------
clear_buff:   ;  Clear buffer
;Description :  Clear wr_cbuff, rd_cbuff
;Parameter   :  none
;Return Value:  none
;Used Reg    :  none
;Call Sub    :  macro clear_cuff, clr_rd_index, clr_wr_index
;
        clr_cbuff   rd_cbuff
        clr_cbuff   wr_cbuff
        call    clr_rd_index
        call    clr_wr_index
        ret
;-------------------------------------------------------
init_8253:    ; Program 8253
;Description :  Program 8253 as Baudrate gen.
;Parameter   :  NONE
;Return Value:  NONE
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,37h
        out     (P8253cmd),a
        call    delay
        ld      a,time_l
        out     (P8253c0),a
        call    delay
        ld      a,time_h
        out     (P8253c0),a
        call    delay
        ret
;-------------------------------------------------------
init_8251:    : Program 8251
;Description :  Initialize ASY PORT
;Parameter   :  NONE
;Return Value:  NONE
;Used Reg    :  a
;Call Sub    :  none
;
        ld      a,0
        out     (ASY0_COMM),a
        call    delay
        out     (ASY0_COMM),a
        call    delay
        out     (ASY0_COMM),a
        call    delay
        ld      a,40h
        out     (ASY0_COMM),a
        call    delay
        ld      a,6eh
        out     (ASY0_COMM),a
        call    delay
        ld      a,37h
        out     (ASY0_COMM),a
;
        ld      a,0
        out     (ASY1_COMM),a
        call    delay
        out     (ASY1_COMM),a
        call    delay
        out     (ASY1_COMM),a
        call    delay
```

```
        ld      a,40h
        out     (ASY1_COMM),a
        call    delay
        ld      a,5eh
        out     (ASY1_COMM),a
        call    delay
        ld      a,37h
        out     (ASY1_COMM),a
        ret
;---------------------------------------------------------------
SET_NMI:        ;  SET NMI
;Description :  Enable NMI
;Parameter   :  none
;Return Value:  none
;Used Reg    :  a
;Call Sub    :  none
;
        xor     a
        out     (PNMI),a
        ret
;---------------------------------------------------------------
delay:          ;  Delay
;Description :  Delay
;Parameter   :  none
;Return Value:  none
;Used Reg    :  b
;Call Sub    :  none
;
        ld      b,0
        djnz    $
        ret
;---------------------------------------------------------------
        endf
```

```
;-------------------------------------------------------
;   DCI-Data Communication Interface Module
;   Name        : DCI001 3.SRC
;   Description : Service Function
;                   - chk dev, set pcm, set tsac ,other
;   Version     : 1.0
;   Date        :
;   Last Edit   : 24 January 1991
;-------------------------------------------------------
Chk_Dev:    ;  Check Device Number to MPU
;Description :  Send Device Number to MPU
;Parameter   :  None
;Return Value:  Content in Wr_Cbuff
;Used Reg    :  HL,A
;Call Sub    :  none
;
        ld      hl,wr_cbuff
        ld      de,temp_cbuff
        ld      a,(hl)              ;Read byte count
        inc     hl                  ;Point to first command
        cp      0
        jp      z,chkdev1
        ld      b,0
        ld      c,a
        ldir
        ld      hl,temp_cbuff
        ld      de,wr_cbuff+1+3
        ld      b,0
        ld      c,a
        ldir
chkdev1: ld     hl,wr_cbuff+1
        Add_Wr_Cbuff    _DEV_NUM
        Add_Wr_Cbuff    Null
        Add_Wr_Cbuff    Null
        inc_bc
        inc index   wr_index
        Set_Req
        ld      a,0
        ret
;-------------------------------------------------------
Set_TSAC:   ;Set Time Slot Assigner Circuit
;Description :  Set -both tx/rx timeslot  (00h)
;                   -only Tx              (40h)
;                   -Only Rx              (80h)
;Parameter   : 30h,xxh,yyh  xx=SSI
;                           yy=TS# or (TSAC Command)
;Return Value:  B-> 1=Sucess / 0=Timeout
;Used Reg    :  a,bc,hl
;Call Sub    :  Prg_TSAC
;
        ld      hl,(Rd_index)   ;Command
        inc     hl
        ld      a,(hl)          ;SSI#
        ld      b,a             ;B <- SSI#  (0-7)
        inc     hl
        ld      a,(hl)          ;TSAC command + TS#
        ld      c,a             ;C<- TSAC com + TS#
        call    prg_Tsac
;       inc index   rd_index
        ld      a,0
        ret
;-------------------------------------------------------
Set_PCM:    ;Set Tx/Rx PCM line
;Description :  Set -only Tx
;                   -Only Rx
;Parameter   :  31h,xxh,yyh  xx000000h xx=00-> both
;                                      xx=01-> tx
;                                      xx=10-> rx
;                           yy=data 00ttrrrr
;Return Value:  none
;Used Reg    :  a,bc,hl
;Call Sub    :  none
;
        ld      hl,(Rd_Index)
        inc     hl
        ld      a,(hl)      ; xx000000h  xx=00->Both
                            ;            xx=01->tx
                            ;            xx=10->rx
        and     0f0h        ;If a=0 then Tx
        cp      01000000b   ;if Zero -> Tx
```

```
        jr      z,spcmtx
        cp      10000000b    ;if zero -> Rx
        jr      z,spcmrx
        ld      a,(hl)
both:   jr      spcm2
spcmtx: ld      a,(hl)       ; 00000ttth -> Tx
        and     00000111b
        rlca
        rlca
        rlca
        ld      b,a          ; 00ttt000h
        ld      a,( pcm)     00xxxrrrh
        and     07h          ; 00000111h
        or      b            ; 00tttrrrh ->A
        jr      spcm2
spcmrx: ld      a,(hl)       ; 00010rrrh
        and     07h          ; 00000111h
        ld      b,a          ; 00000rrrh
        ld      a,( pcm) ; 00tttxxxh
        and     00111000b    ; 00111000h
        or      b            ; 00tttrrrh
spcm2:  ld      ( pcm),a
        out     (P82C55),a
;       inc index   rd index
        ld      a,0
        ret
;---------------------------------------------------------------
OTHER:          ; OTHER
;Description :  Send Unrecognize command back to MPU (TEST)
;Parameter   :  none
;Return Value:  none
;Used Reg    :  hl,de,bc
;Call Sub    :  set req
;
        push    de
        ld      hl,(wr_index)
        add wr_cbuff 6h
        ld      de,(rd_cbuff)
        ld      b,6
o1:     ld      a,(de)
        ld      (hl),a
        djnz    o1
        INC_BC
        INC_INDEX wr_index
        set req
        pop     de
        ret
;---------------------------------------------------------------
        endf
```

ภาคผนวก ง

คำสั่งและตัวแปรที่ใช้งานในโปรแกรมควบคุม

ง-1 คำสั่งใช้ในการติดต่อระหว่างหน่วยควบคุมย่อยและหน่วยควบคุมหลัก

ระหว่างการทำงานจะมีการส่งคำสั่งไปมาระหว่างหน่วยควบคุมย่อยและหน่วยควบคุมหลัก คำสั่งนี้จะถูกส่งผ่านไปทางหน่วยเชื่อมโยงหน่วยควบคุม ลักษณะของคำสั่งที่ใช้จะประกอบด้วยข้อมูลจำนวน 3 ไบต์ต่อ 1 ชุด โดยไบต์แรกเป็นคำสั่ง ไบต์ที่สองและสามเป็นข้อมูล จำนวนคำสั่งที่รับส่งได้ใน 1 ครั้งคือ 80 ชุด หรือ 240 ไบต์ คำสั่งเหล่านี้จะถูกเขียนลงในหน่วยความจำร่วมที่มีอยู่ในหน่วยเชื่อมโยงหน่วยควบคุมส่วนที่อยู่กับหน่วยควบคุมหลัก คำสั่งเหล่านี้ถูกกำหนดไว้ใน Header file ชื่อว่า SSICMD.H ซึ่งแสดงไว้ในภาคผนวก ค

คำสั่งต่างๆ มีรายละเอียดดังต่อไปนี้

1. คำสั่งทั่วไป ได้แก่คำสั่งที่จะต้องมีในหน่วยควบคุมย่อยทุกหน่วยเพื่อควบคุมการทำงานทั่วไป

- _chk_dev (20h) : เป็นคำสั่งที่ใช้สอบถามหมายเลขอุปกรณ์
- _set_tsac (30h) : คำสั่งในการโปรแกรมการทำงานของ TSAC ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล ไบต์ที่สามเป็นค่าที่จะโปรแกรม TSAC
- _set_pcm (31h) : เป็นคำสั่งในการโปรแกรมการใช้งานเส้นสัญญาณร่วม

2. คำสั่งสำหรับหน่วยเชื่อมโยงโทรศัพท์สายภายใน ได้แก่

- _ssi_offh (32h) : คำสั่งรายงานการยกหูโทรศัพท์สายภายในที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล
- _ssi_onh (33h) : คำสั่งรายงานการวางหูโทรศัพท์สายภายในที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล
- _ssi_fl (34h) : คำสั่งรายงานการแฟลชโทรศัพท์สายภายในที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล
- _ssi_dp (35h) : คำสั่งรายงานการหมุนหมายเลขโทรศัพท์สายภายในที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล
- _ssi_enr (37h) : คำสั่งเปิดสัญญาณกระดิ่งโทรศัพท์สายภายในที่ส่งมา

จากหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _ssi_disr (38h) : คำสั่งปิดสัญญาณกระดิ่งโทรศัพท์สายภายในที่ส่งมาจากหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _ssi_click (27h) : คำสั่งปิดสัญญาณกระดิ่งโทรศัพท์สายภายในสั้นๆ หนึ่งครั้งที่ส่งมาจากหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

3. คำสั่งสำหรับหน่วยเชื่อมโยงโทรศัพท์สายภายนอก ได้แก่

- _coi_onh (41h) : คำสั่งยกหูโทรศัพท์สายภายนอกที่ส่งมาจากหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _coi_offh (42h) : คำสั่งวางหูโทรศัพท์สายภายนอกที่ส่งมาจากหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _coi_fl (43h) : คำสั่งแฟลชโทรศัพท์สายภายนอกที่ส่งมาจากหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _coi_dial (44h) : คำสั่งให้ส่งพัลส์การหมุนหมายเลขโทรศัพท์สายภายนอกที่ส่งมาจากหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล ไบต์ที่สามเป็นหมายเลขที่จะหมุน

- _coi_ring (45h) : รายงานการตรวจพบสัญญาณกระดิ่งของโทรศัพท์สายภายนอกที่ส่งไปยังหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _coi_rcnl (47h) : รายงานการตรวจพบการยกเลิกสัญญาณกระดิ่งของโทรศัพท์สายภายนอกที่ส่งไปยังหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _coi_drdy (46h) : รายงานการเสร็จสิ้นการหมุนหมายเลขของโทรศัพท์สายภายนอกที่ส่งไปยังหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

4. คำสั่งสำหรับหน่วยถอดรหัสสัญญาณDTMF มีเพียงคำสั่งเดียวคือ _dtmf_digit (60h) ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล ไบต์ที่สามเป็นค่าที่ถอดรหัสได้

5. คำสั่งสำหรับหน่วยแลกเปลี่ยนช่องเวลา ได้แก่

- _tsi_ctone (23h) : เป็นคำสั่งให้ส่งสัญญาณโทนแบบต่างๆ ไปยังช่องเวลาที่กำหนด โดยไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขช่องเวลา ไบต์ที่สามเป็นชนิดของเสียงโทน

- _tsi_exts (24h) : คำสั่งให้ทำการสลับช่องเวลา ไบต์แรกเป็นคำสั่ง ไบต์ที่สองและสามเป็นหมายเลขช่องเวลา

6. คำสั่งสำหรับหน่วยเชื่อมโยงโอเปอเรเตอร์ ได้แก่

- _opi_enr (37h) : คำสั่งเปิดสัญญาณกระดิ่งของโอเปอเรเตอร์ที่ส่งมาจากหน่วยควบคุมหลักไปหน่วยควบคุมย่อย ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _opi_disr (38h) : คำสั่งปิดสัญญาณกระดิ่งของโอเปอเรเตอร์ที่ส่งมาจากหน่วยควบคุมหลักไปหน่วยควบคุมย่อย ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _opi_onh (41h) : คำสั่งรายงานการวางสายของโอเปอเรเตอร์ที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _opi_offh (42h) : คำสั่งรายงานการยกหูของโอเปอเรเตอร์ที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _opi_fl (43h) : คำสั่งรายงานการแฟลชวางสายของโอเปอเรเตอร์ที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

- _opi_dial (44h) : คำสั่งรายงานการหมุนหมายเลขของโอเปอเรเตอร์ที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล ไบต์ที่สามเป็นหมายเลขหมุน

- _opi_hold (0Ah) : คำสั่งรายงานการพักสายของโอเปอเรเตอร์ที่ส่งมาจากหน่วยควบคุมย่อยไปหน่วยควบคุมหลัก ไบต์แรกเป็นคำสั่ง ไบต์ที่สองเป็นหมายเลขมอดูล

ง-2 ตัวแปรที่ใช้ในโปรควบคุมหน่วยควบคุมหลัก

ในโปรแกรมควบคุมหน่วยควบคุมหลักมีตัวแปรที่ใช้เก็บค่าสภาวะต่างๆ ซึ่งจะเป็นตัวแปรแบบชนิดต่าง ๆ ที่ใช้ในภาษา C ตัวแปรต่างๆ เหล่านี้กำหนดไว้ในแฟ้มข้อมูลชื่อว่า SSIVAR.H

ตัวแปรต่างๆที่ใช้ในโปรแกรมควบคุมหน่วยควบคุมหลักมีรายละเอียดดังต่อไปนี้

1. DEV [] เป็นอาเรย์ของตัวแปรโครงสร้างที่มีจำนวนสมาชิกภายในอาเรย์เท่ากับจำนวนสูงสุดของช่องเวลา คือ 256 พอร์ต โครงสร้างนี้กำหนดขึ้นเพื่อเก็บค่าสถานะของอุปกรณ์สื่อสารทุกชนิดที่ต่อเข้ากับระบบ โครงสร้าง DEV จะมีสมาชิกดังต่อไปนี้ (รายละเอียดแสดงในไฟล์ SSIVAR.H)

- dev_typ บอกชนิดของอุปกรณ์
- pos บอกตำแหน่งสล็อตและมอดูลของอุปกรณ์

- tspcm ตัวแปรที่บอกตำแหน่งการใช้งานช่องเวลาและสายสัญญาณร่วม
- state ตัวแปรที่บอกสถานะในขณะนี้ของอุปกรณ์สื่อสาร
- ostate ตัวแปรที่เก็บค่าสถานะเดิมของอุปกรณ์สื่อสาร
- down_cnt ตัวแปรที่ใช้ในการนับคาบเวลาต่างๆ
- a เป็นตัวแปรที่เก็บสภาวะการยกหูวางหูของโทรศัพท์สายภายใน โทรสัพท์สายภายนอก
- b เป็นตัวแปรที่เก็บค่าการแฟลชของโทรศัพท์สายภายใน หรือเก็บค่าสภาวะการมีสัญญาณกระดิ่งของโทรศัพท์สายภายนอก
- c เป็นตัวแปรที่เก็บค่าหมายเลขการหมุนของโทรศัพท์
- opi_hold เป็นตัวแปรที่เก็บสภาวะการพักสายของโอเปอเรเตอร์
- *pdial,dial_cnt,dial[16] เป็นตัวแปรที่เป็นตัวชี้และตัวแปรที่เก็บค่าหมายเลขที่ทำการหมุนของอุปกรณ์ทั้งหมด
- *predi,redi_cnt,redi[16] เป็นตัวแปรที่เป็นตัวชี้และตัวแปรที่เก็บค่าหมายเลขที่จะทำการหมุนซ้ำของอุปกรณ์ทั้งหมด เมื่อได้รับคำสั่งหมุนซ้ำ (Redial)
- dev_called เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ปลายทางที่อุปกรณ์นี้ทำการติดต่อด้วย
- dev_hold เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ที่อุปกรณ์ติดต่อด้วยแล้วทำการพักสายเอาไว้
- dev_dtmf เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ถอดรหัสสัญญาณ DTMF ที่อุปกรณ์นี้ทำการติดต่อด้วย
- dev_co เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ของโทรศัพท์สายภายนอกที่ทำการติดต่อด้วย
- dev_callb เป็นตัวแปรที่เก็บค่าหมายเลขของโทรศัพท์สายภายในที่ทำการจองสายอุปกรณ์นี้ไว้
- dev_callbdes เป็นตัวแปรเก็บที่หมายเลขอุปกรณ์โทรศัพท์สายภายในปลายทางที่อุปกรณ์นี้ทำการจองสายไว้
- dev_callf เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ที่ฝากสายไว้
- dev_callfbusy เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ที่ฝากสายเมื่อสายไม่ว่างไว้
- dev_callfno เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ที่ฝากสายเมื่อไม่มีผู้รับสาย

- dev_callwait เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ที่รอสายอยู่
- hot เป็นตัวแปรที่เก็บหมายเลขอุปกรณ์ปลายทางที่ตั้งบริการสายด่วนไว้
- hunt เป็นตัวแปรที่เก็บหมายเลขของกลุ่มการใช้ (Hunting group)
- svc เป็นตัวแปรที่เก็บระดับการใช้บริการของอุปกรณ์
- stn เป็นตัวแปรที่เก็บค่าหมายเลขโทรศัพท์สายภายใน สายภายนอก หรือหมายเลขอุปกรณ์สื่อสารแบบอะซิงโครนัส
- fea,src,tmp เป็นตัวแปรสำรองที่ใช้เก็บค่าต่างๆระหว่างการใช้งาน
- start เป็นตัวแปรที่เก็บค่าเวลาเริ่มการใช้งานสายภายนอก

2. Slot_tbl[] เป็นตัวแปรอาเรย์ที่เก็บชนิดของอุปกรณ์ในตำแหน่งสล็อตต่างๆ
3. Code_set[] เป็นตัวแปรอาเรย์ที่เก็บรหัสการใช้งานของบริการพิเศษ
4. Code_cnl[] เป็นตัวแปรอาเรย์ที่เก็บรหัสยกเลิกการใช้งานของบริการพิเศษ
5. Speed_tbl[] เป็นตัวแปรอาเรย์ที่เก็บหมายเลขย่อ
6. Tsi_add เป็นตัวแปรที่เก็บตำแหน่งสล็อตของหน่วยแลกเปลี่ยนช่องเวลา
7. Tsi_data[] เป็นตัวแปรอาเรย์ที่เก็บค่าของการสลับช่องเวลา
8. Night[],Night_flag เป็นตัวแปรที่เก็บค่าเกี่ยวกับ Night Connection
9. Dtmf_add[] เป็นตัวแปรที่เก็บตำแหน่งอุปกรณ์ของหน่วยถอดรหัสสัญญาณ DTMF
10. Opi_add[] เป็นตัวแปรที่เก็บตำแหน่งอุปกรณ์ของหน่วยเชื่อมโยงโอเปอเรเตอร์
11. Qdev[],Qcnt เป็นตัวแปรที่เก็บค่าเกี่ยวกับอุปกรณ์ที่มีการเปลี่ยนแปลงเกิดขึ้น
12. coQ[] เป็นตัวแปรที่เก็บค่าหมายเลขอุปกรณ์ที่ทำการจองสายภายนอกไว้

ประวัติผู้เขียน

นายสุรศักดิ์ อุทโยภาศ เกิดเมื่อวันที่ 22 มิถุนายน 2506 ที่ กรุงเทพมหานคร ได้รับปริญญาวิศกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า จากมหาวิทยาลัยเชียงใหม่ เมื่อปี 2529 และเข้าศึกษาต่อปริญญามหาบัณฑิต ที่ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย ในปีเดียวกัน