เอกสารอ้างอิง

1. IBM,"IBM PC 3270 Emulation Program","System Planner's and User's Guide",International Business Machines Corporation. 1986
2. IBM,"IBM PC 3270 Emulation Program","User's Reference", Internation Business Machines Corporation. 1986
3. AST RESEARCH, Inc.,"AST-SNA IBM 3270 Emulation for the IBM Personal Computer, IBM PC-XT and other IBM-Compatible System",AST RESEARCH,Inc. 1985
4. Editor-in-chief,Data Communication,"Linking Microcomputer", McGraw-Hill,Inc. 1985
5. Frank J.Derfler,"The Micro-to-Mainframe Decision Matrix", PC MAGAZINE. May 1986
6. IBM,"Resource Definition Referance for the IBM 3705", International Business Machines Corparation. 1983
7. IBM,"Customer Information Control System/Virtual Storage (CICS/VS) Version 1 Release 6 ","Resource Definition Guide",International Business Machines Corparation. 1984
8. Peter Abel,"Assembler for the IBM PC and PC-XT",Reston Publishing Company,Inc. 1984
9. Borland International,Inc,"TURBO C","Referance Guide","User's Guide", Borland International, Inc. 1987
10. Borland International,Inc,"TURBO BASIC","Owner's Handbook", Borland International, Inc. 1987
11. บุญเลิศ เอียมทัศนา,ยืน ภู่วรวรรณ,สมนึก คีรีโต,"โปรแกรมคอมพิวเตอร์ภาษาซี", บริษัท ซีเอ็ดยูเคชั่น จำกัด,พิมพ์ครั้งที่ 1 พ.ศ. 2529

ภาคผนวก ก.

การเตรียมฮาร์ดแวร์และซอฟต์แวร์บนไมโครคอมพิวเตอร์

ภาคผนวก ก.

การเตรียมฮาร์ดแวร์และซอฟต์แวร์บนไมโครคอมพิวเตอร์

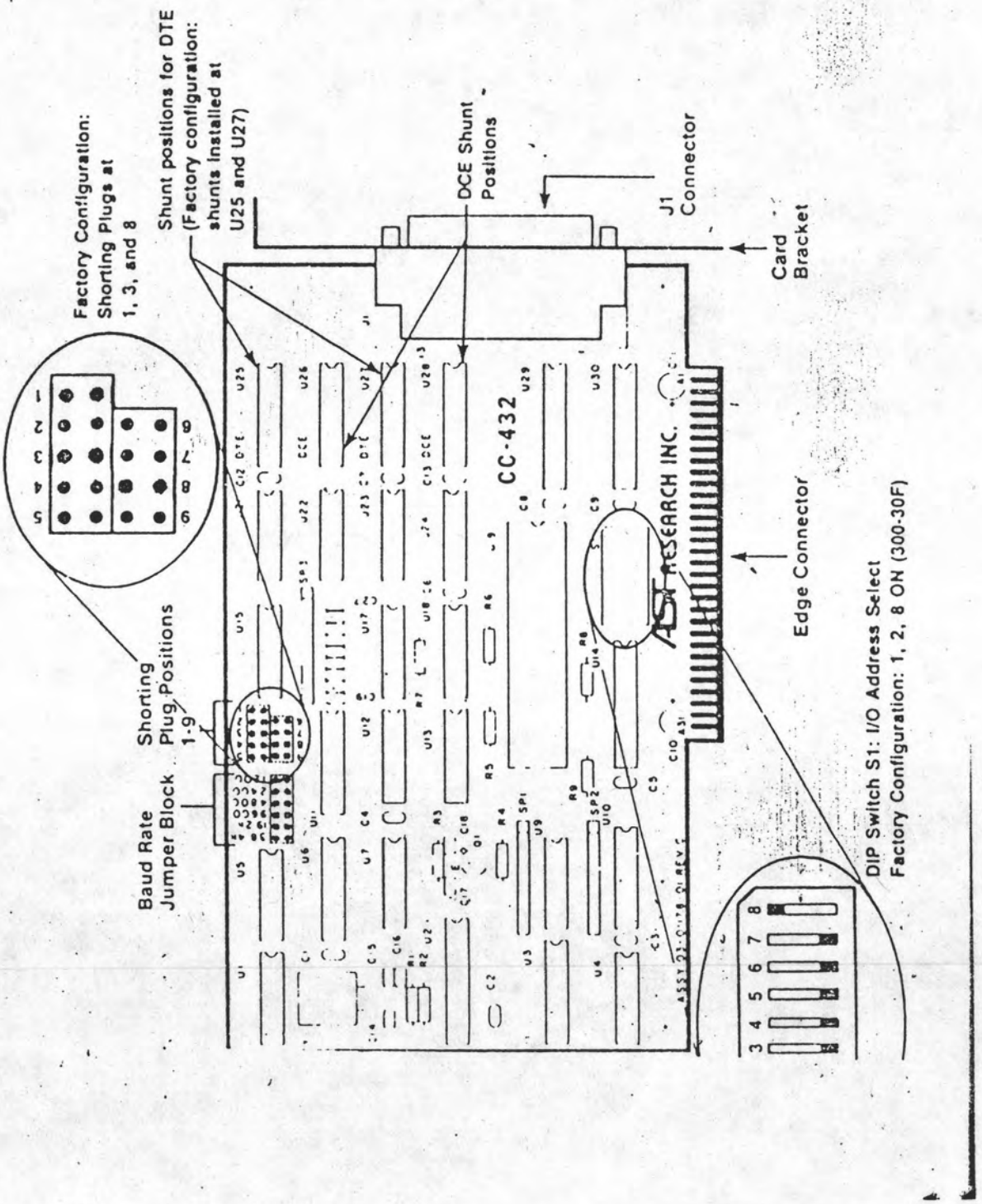
การนำคาร์ดเอเอสที มาติดตั้งบนไมโครคอมพิวเตอร์ เพื่ออิมูเลตเป็นเทอร์มินอลของเมนเฟรมนั้น จะต้องมีขั้นตอนที่สำคัญ 3 ส่วน ดังรายละเอียดในแต่ละหัวข้อ ดังนี้ [3]

1. Hardware Configuration

คาร์ดเอเอสทีได้ถูกออกแบบมาเพื่อให้ใช้กับโปรโตคอลได้หลายแบบ ดังนั้น การจะนำคาร์ดมาใช้งานจำเป็นต้องปรับให้เข้ากับลักษณะของโปรโตคอลแต่ละชนิดที่จะนำไปใช้ และเราสามารถเปลี่ยนค่าของแอดเดสของอินพุทเอาท์พุทอินเทอรัพท์ (I/O Address Interrupt) จากค่าที่กำหนดมาจากโรงงาน ถ้าหากว่าค่าที่กำหนดนั้นไปตรงกับค่าที่ใช้อยู่ในไมโครคอมพิวเตอร์ การเปลี่ยนแปลงค่าดูได้จากตาราง ก.1 และรูป ก.1 โดยเปลี่ยนแปลงสวิทช์ S1

S1 Position 1	S1 Position 2	Hexadecimal I/O Address
ON	ON	300-30F
OFF	ON	320-32F
ON	OFF	340-34F
OFF	OFF	360-36F

ตารางที่ ก.1 ตารางกำหนดแอดเดสของอินพุทเอาท์พุทอินเทอร์รัพท์บนเอเอสทีคาร์ด



รูปที่ ก.1 แสดงสวิทช์และส่วนต่าง ๆ ของเอเอสทีคาร์ด

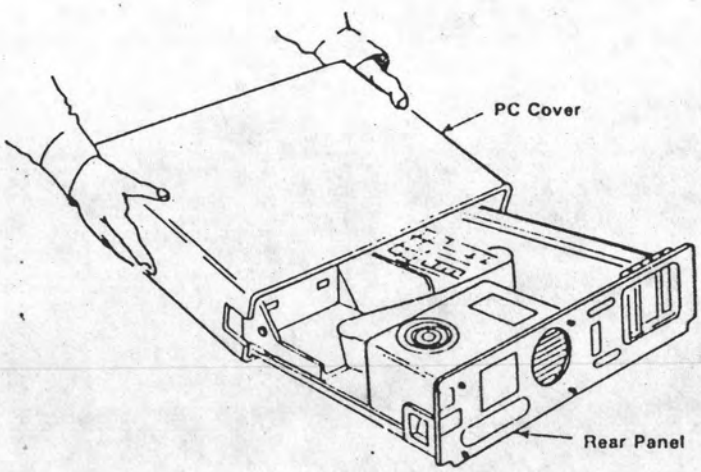
ซึ่งค่าที่กำหนดมาจากโรงงานคือ 300-30F และการติดต่อของไมโครคอมพิวเตอร์กับเมนเฟรมนั้น ไมโครคอมพิวเตอร์จะเป็นเทอร์มินอล ดังนั้นเราต้องเปลี่ยนแปลงสวิทช์ S3 ตำแหน่งที่ 1,3 และ 8 ให้มีสภาวะเปิด (ON)

สำหรับ โมเด็ม ที่ใช้ทำการแปลงสัญญาณรับ-ส่งจากไมโครคอมพิวเตอร์ไปยังเมนเฟรม จะใช้โมเด็มแบบซิงโครนัท (Synchronous Modem) โดยสามารถเลือกอัตราการส่งได้ตั้งแต่ 2,400 ถึง 9,600 บิทต่อวินาที (2,400-9,600 bps)

2. การติดตั้งคาร์ดเอเอสทีในไมโครคอมพิวเตอร์

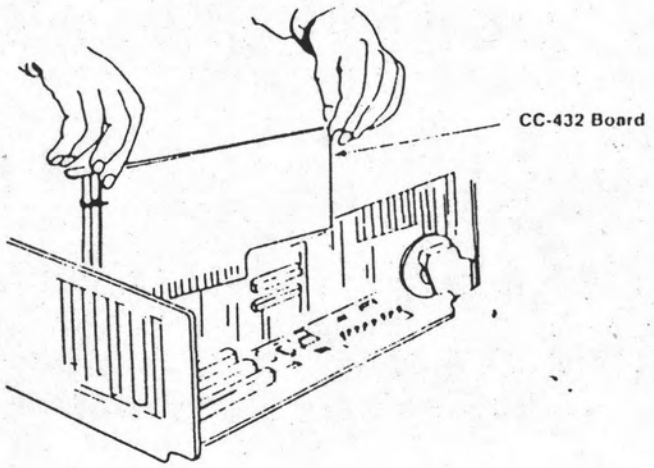
การนำคาร์ดเอเอสทีมาติดตั้งในไมโครคอมพิวเตอร์มี 3 ขั้นตอนดังนี้

2.1 ถอดฝาครอบตัวไมโครคอมพิวเตอร์ ดังรูปที่ ก.2



รูปที่ ก.2 แสดงวิธีการถอดฝาครอบเครื่องไมโครคอมพิวเตอร์

2.2 เสียบคาร์ดเอเอสทีบนสลอทที่ว่าง (expansion slot) ดังรูปที่ ก.3



รูปที่ ก.3 แสดงวิธีการใส่คาร์ดเอเอสทีบนไมโครคอมพิวเตอร์

2.3 ปิดฝาครอบเครื่อง

3. Software Configuration บนไมโครคอมพิวเตอร์

หลังจากที่เรานำคาร์ดเอเอสทีใส่ในเครื่องแล้ว ต้องใช้โปรแกรมของเอเอสทีเพื่อให้ไมโครคอมพิวเตอร์สามารถอิมูเลตเป็นเทอร์มินอลได้

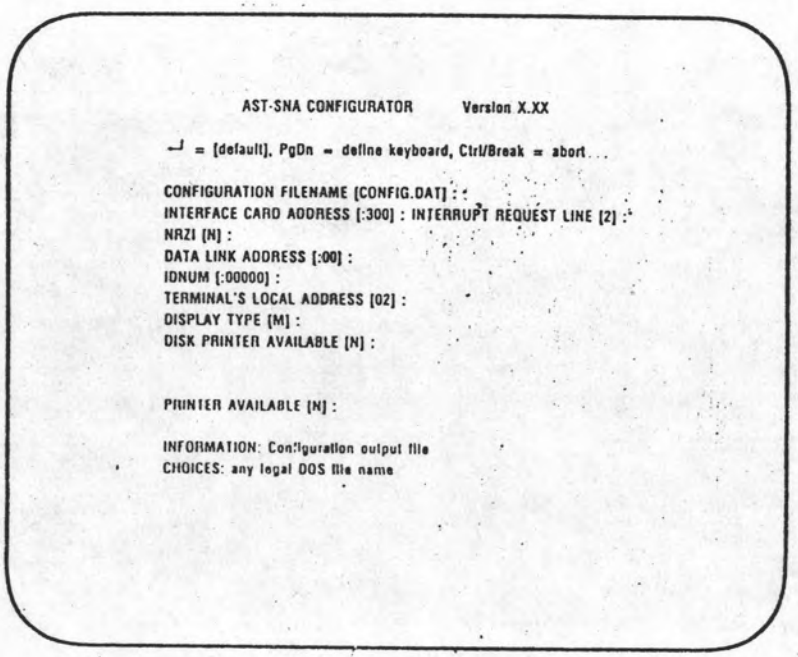
แต่ก่อนที่จะนำซอฟต์แวร์มาใช้งานนั้นเราต้องกำหนดค่าของพารามิเตอร์ต่างๆ ให้ตรงกับค่าที่กำหนดไว้บนเมนเฟรม ซึ่งมีขั้นตอนในการจัดเตรียมดังนี้

สร้าง Configuration File

เมื่อเปิดเครื่องและโหลดดอสแล้วให้ใส่แผ่นโปรแกรมของเอเอสทีที่ติดมากับคาร์ดไว้ที่ไดร์ว A แล้วใส่คำสั่งต่อไปนี้ เพื่อทำการเซ็ทค่าให้ตรงกับความต้องการ

A>SNACFG [filename] <Enter>

โดยที่ filename จะเป็นชื่อ Configution File ที่จะสร้างหรือเปลี่ยนแปลง หลังจากกดปุ่ม <Enter> แล้วบนจอมอนิเตอร์จะขึ้นข้อความดังรูป ก.4

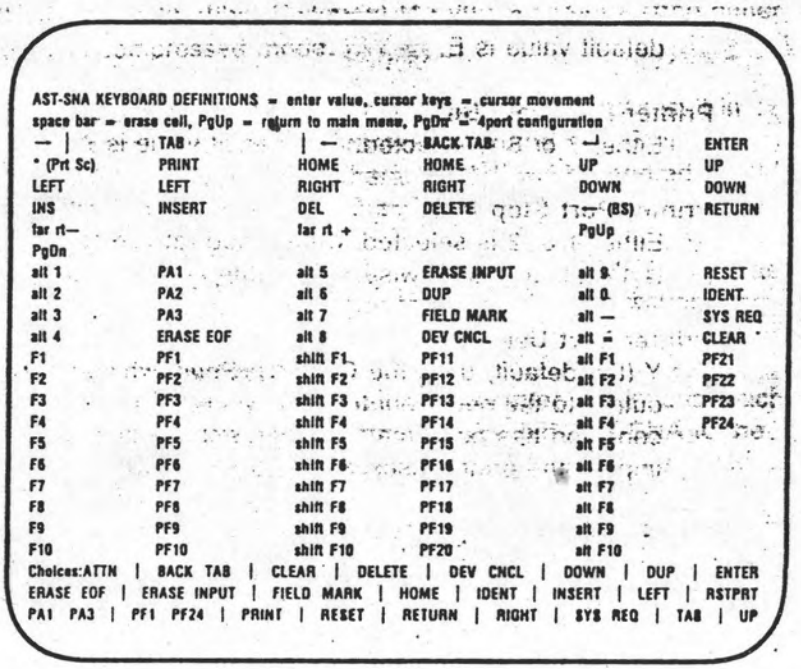


รูปที่ ก.4 แสดงผลบนจอภาพสำหรับใช้ในการสร้าง Configuration File

พารามิเตอร์แต่ละตัวบนจอมีความหมายดังรายละเอียดต่อไปนี้

1. CONFIGURATION FILENAME	1. ชื่อของ Configuration File ที่จะนำไปใช้งาน
2. INTERFACE CARD ADDRESS และ INTERUPT REQUEST LINE	2. แอดเดรสนี้ต้องตรงกับที่เซ็ทบนคาร์ด
3. NRZI	3. เป็นพารามิเตอร์ที่ต้องกำหนดให้ตรงกับพารามิเตอร์ที่เซ็ทค่าไว้บนเมนเฟรม
4. DATALINK ADDRESS	4. ไม่ใช้ (ใส่ 0)
5. IDNUM	5. ไม่ใช้ (ใส่ 0)
6. TERMINAL LOCAL ADDRESS	6. เป็นแอดเดรสของเทอร์มินอล ต้องมีค่าตรงกับที่เซ็ทไว้บนเมนเฟรม
7. DISPLAY TYPE	7. ใช้ C สำหรับจอสี M สำหรับโมโนโครม
8. DISK PRINTER AVAILABLE	8. ไม่ใช้ (ใส่ N)
9. PRINTER AVAILABLE	9. ไม่ใช้ (ใส่ N)

การเปลี่ยนแป้นอักษรบนไมโครคอมพิวเตอร์ ให้ตรงกับแป้นตัวอักษรของเทอร์มินอลของเมนเฟรม สามารถทำต่อจากจอของการสร้าง Configuration File ได้เลย โดยการกดปุ่ม <PgDn> ก็จะมีข้อความปรากฏบนจอ ดังรูป ก.5



รูปที่ ก.5 แสดงผลบนจอภาพสำหรับการแก้ไข เปลี่ยนแปลงปุ่มบนแป้นพิมพ์

ซึ่งสามารถ เปลี่ยนแปลงปุ่มของไมโครคอมพิวเตอร์ แต่ละตำแหน่งให้ตรงกับแป้นเทอร์มินอล 3278/79 ของ IBM ได้ เมื่อแก้ไขให้ตรงกับความต้องการแล้วให้กลับไปที่หน้าแรก และกดปุ่ม <Esc> เพื่อทำการบันทึก (Save) Configuration File ไว้ หรือกดปุ่ม <Ctrl> <Break> เพื่อยกเลิกการแก้ไข Configuration File

ภาคผนวก ข.

การเตรียมฮาร์ดแวร์และซอฟต์แวร์บนเมนเฟรม

ภาคผนวก ข.

การเตรียมฮาร์ดแวร์และซอฟต์แวร์บนเมนเฟรม

การจัดเตรียมฮาร์ดแวร์บนเมนเฟรมมีอุปกรณ์ที่ต้องใช้คือ [3]

1. เครื่องคอมพิวเตอร์เมนเฟรม IBM 43XX
2. คอมมูนิเคชั่นคอนโทรลยูนิท (Communication Control Unit) IBM 37X5
3. โมเด็ม (Modem) ซึ่งเซ็ทค่าตรงกับโมเด็มทางด้านไมโครคอมพิวเตอร์

การจัดเตรียมซอฟต์แวร์บนเมนเฟรม เพื่อให้ไมโครคอมพิวเตอร์ติดต่อเข้ามาได้ ต้องมีการเตรียมซอฟต์แวร์ทางด้านคอมมูนิเคชั่น (NCP/VTAM) และทางด้านออนไลน์คือ ซีไอซีเอส (CICS) ดังนี้

<u>การเตรียมซอฟต์แวร์ทางด้านคอมมูนิเคชั่น</u>

ในการเตรียม VTAM/NCP สำหรับ port ของไมโครคอมพิวเตอร์ที่จะนำมาต่อนั้น จะต้องมีการกำหนดค่าพารามิเตอร์หลายตัว เพื่อบอกให้ VTAM/NCP รู้ว่าไมโครคอมพิวเตอร์ที่อิมูเลตต้องการจะติดต่อนั้น มีลักษณะอย่างไร ซึ่งจะมีพารามิเตอร์ที่สำคัญต้องกำหนดในแมโคร LINE, PU และ LU ดังนี้

```
LINE  ADDRESS={line addr}
      ,SPEED=(rate)
      [,CLOCKING=EXT]
      [,NRZI={NO/YES}]
      [,DUPLEX=(HALF/FULL)]
      [,POLLED=YES]
```

คำอธิบาย

ADDRESS={line addr}	เป็น port แอดเดรสที่จะใช้สำหรับติดต่อกับไมโครคอมพิวเตอร์
,SPEED=(rate)	เป็นอัตราความเร็วที่ใช้ในการส่งข้อมูลระหว่างไมโครคอมพิวเตอร์และเมนเฟรม ขึ้นอยู่กับชนิดของ โมเด็ม แต่ต้องไม่เกิน 9600 บิทต่อวินาที
[,CLOCKING=EXT]	ใช้ CLOCK ของโมเด็ม
[,NRZI={NO/YES}]	จะใช้ NO หรือ YES ก็ได้ แต่จะต้องตรงกับที่กำหนดในไมโครคอมพิวเตอร์
[,DUPLEX=(HALF/FULL)]	ลักษณะการส่งข้อมูลจะส่งแบบ HALF หรือ FULL ก็ขึ้นกับขนาดของข้อมูลว่ามีมากน้อยแค่ไหน และจะใช้สายโทรศัพท์ 1 คู่ หรือ 2 คู่
[,POLLED=YES]	ตัวไมโครคอมพิวเตอร์ จะถูก poll และ address

```
PU      [ADDR=nn]
        [,MAXDATA=265]
        [,MAXOUT=7]
        [,MAXLU=nn]
        [,PUTYPE=2]
```

คำอธิบาย

[ADDR=nn] เป็นแอดเดสของอุปกรณ์ (ไมโครคอมพิวเตอร์ที่อิมูเลตเป็นเทอร์มินอล) มีค่าได้ตั้งแต่ X'01' ถึง X'FE' และค่าที่กำหนดสำหรับแอดเดสนี้จะต้องตรงกับค่าที่กำหนดใน Configuration File

[,MAXDATA=265] กำหนดขนาดของเอสดีแอลซีเฟรมเป็น 256 ไบท์ สามารถกำหนดต่ำกว่านี้ได้ แต่ขนาด 256 ไบท์จะทำให้มีประสิทธิภาพดีกว่า

[,MAXOUT=7] กำหนดขนาดของจำนวนเอสดีแอลซีเฟรมซึ่งจะถูกส่งโดย NCP ก่อนได้รับการตอบสนองจากข่ายเชื่อมโยงข้อมูล ขนาดสูงสุดที่กำหนดได้คือ 7 และเป็นขนาดที่ให้ประสิทธิภาพสูงสุด

[,MAXLU=nn] กำหนดขนาดสูงสุดของหน่วยเชิงตรรก (Logical Unit) สำหรับการใช้งานที่เป็นจอภาพและเครื่องพิมพ์เราจะกำหนดให้ค่าเท่ากับ 2

LU [LOCADDR=nn]

[,BATCH=NO]

คำอธิบาย

[LOCADDR=nn]	กำหนดแอดเดสของหน่วยเชิงตรรก (Logical Unit Address) ซึ่งจะต้องตรงกับค่าที่กำหนดใน TERMINAL'S LOCAL ADDRESS ของ Configulation File.
[,BATCH=NO]	กำหนดค่าเป็น NO เพื่อให้มีค่าลำดับความสำคัญสูง

การเตรียมซอฟต์แวร์ทางด้านซีไอซีเอส

ซีไอซีเอสเป็นซอฟต์แวร์ฐานข้อมูลและการสื่อสารข้อมูล (Data Base / Data Communication) ซึ่งจะช่วยให้โปรแกรมเมอร์สามารถพัฒนาระบบงานได้สะดวกและรวดเร็วยิ่งขึ้น โดยไม่ต้องคำนึงถึงวิธีการเข้าถึงข้อมูลและการติดต่อสื่อสารข้อมูลกับเทอร์มินอลต่าง ๆ ภายใต้ซีไอซีเอส แต่ผู้ใช้จะต้องแจ้งให้นักโปรแกรมระบบ (System Programmer) ทราบด้วยว่าต้องการใช้แฟ้มข้อมูลแบบใดและต้องการใช้เทอร์มินอลด้วย โดยมีวิธีการกำหนดความต้องการนี้ในเทอร์มินอลคอลโทรนเทเบิล (Terminal Control Table) และไฟล์คอลโทรนเทเบิล (File Control Table) มีรายละเอียดการกำหนดพารามิเตอร์ดังข้างล่างนี้

การกำหนดพารามิเตอร์ในเทอร์มินอลคอลโทรนเทเบิล

```
DFHTCT  TYPE=TERMINAL
        ,TRMTYPE=LUTYPE2
        [,DEFSCRN=(24,80)]
        [,BUFFER=1536]
        [,RUSIZE=256]
        [,CLASS={CONV/VIDEO}]
        [,CHNASSY=YES]
        [,TRMSTAT=TRANSCEIVE]
        [,FEATURE=AUDALARM]
        [,ACCMETH=VTAM]
```

คำอธิบาย

TYPE=TERMINAL	แสดงว่าการกำหนดนี้ใช้เป็นเทอร์มินอล
,TRMTYPE=LUTYPE2	แสดงว่าเทอร์มินอลนี้เป็นหน่วยเชิงตรรกชนิดที่ 2 (Logical unit type 2)
[,DEFSCRN=(24,80)]	ขนาดของจอภาพมี 24 บรรทัด และ 80 สดมภ์
[,BUFFER=1536]	ขนาดของบัฟเฟอร์หน่วยเชิงตรรกใช้ 1536 ไบท์

[,RUSIZE=256]	ขนาดสูงสุดของรีเควชยูนิท เท่ากับ 256 ไบท์
[,CLASS=CONV]	อุปกรณ์ใช้สำหรับสนทนา (Conversation)
[,CLASS=VIDEO]	อุปกรณ์ใช้สำหรับแสดงผล
[,CHNASSY=YES]	กำหนดสำหรับหน่วยแสดงผลแบบ เอส เอ็น เอซึ่งมีหน่วย เชิงตรรก เป็น LUTYPE2
[,TRMSTAT=TRANCEIVE]	กำหนดว่าข้อมูลจะถูกส่งอัตโนมัติไปยัง เทอร์มินอลโดยผู้ใช้
[,ACCMETH=VTAM]	การติดต่อสื่อสารข้อมูลใช้วิธีการ เข้าถึงแบบ VTAM

ภาคผนวก ค.

โปรแกรมเชื่อมโยงบนไมโครคอมพิวเตอร์สำหรับภาษาซี

```
;  ************************************
;  *  Interface Program for C language  *
;  ************************************
           page

_DATA      SEGMENT BYTE PUBLIC 'DATA'
_DATA      ENDS

_BSS       SEGMENT BYTE PUBLIC 'BSS'
_BSS       ENDS

DGROUP     GROUP   _DATA, _BSS

_TEXT      SEGMENT BYTE PUBLIC 'CODE'
           PUBLIC  _API_C
           ASSUME  CS:_TEXT, DS:DGROUP

_API_C     PROC    near
           push    bp                  ; save location of bp
           mov     bp,sp               ; move stack pointer to bp

;
;---------- Allocate device no.2 ----------------------
;
           mov     al,01h              ; Allocate device function.
           mov     ah,10000010b        ; Device number 2
           int     59h                 ; Interupt to emulation kernel

;
;----------- Get buffer address -----------------------
;

           mov     al,03h              ; Get buffer address function.
           mov     ah,02h              ; Device number 2
           int     59h                 ; Interupt to emulation kernel.
           cmp     al,00h              ; Function successful ?
           je      buffer_ok           ;    yes, ... go next step.
error_1:
           mov     al,31h              ;    no, .. then
           mov     bx,[bp + 10]        ;           move error code
           mov     [bx],al             ;           to parm4
           pop     bp                  ; restore bp
           ret                         ; return to calling program.
;
buffer_ok:
           mov     cs:buff_addr,es     ; save screen buffer address
           mov     cs:buff_disp,di     ;   from es:di to work area.

;------------- Clear screen ----------------------
;

           mov     al,07h              ; control key entry function
           mov     ah,02h              ; device number 2
           mov     bh,2dh              ; CLEAR key
           int     59h                 ; interupt to emulation kernel
```

```
;
;----------------- Move data to screen buffer --------------------
;
move_tranid:                            ; Trans_id = HAPI

            mov     dx,cs:buff_addr     ;
            mov     di,0000h            ;

            push    ds                  ; save ds
            push    dx                  ;
            pop     ds                  ; move dx to ds
;
            mov     al,48h              ; H       48
            mov     [di],al
            inc     di
            mov     al,41h              ; A       41
            mov     [di],al
            inc     di
            mov     al,50h              ; P       50
            mov     [di],al
            inc     di
            mov     al,49h              ; I       49
            mov     [di],al
            inc     di
            pop     ds

;
move_program:                           ; Host user program name

            mov     bx,[bp + 4]         ; get first parameter
            mov     cx,0008h            ; parameter length = 8
            mov     si,bx               ; si point to parm1
            mov     dx,cs:buff_addr     ; move screen buffer address to dx
            mov     di,0004h            ; skip 4 bytes

trdata1:
            mov     al,[si]             ; move
            push    ds                  ;
            push    dx                  ;   parm1
            pop     ds                  ;
            mov     [di],al             ;        to
            inc     si                  ;
            inc     di                  ;           screen buffer.
            pop     ds
            loop    trdata1             ;

;
move_content:                               ; User Data

            mov     bx,[bp + 12]        ; get 5's paramater (length of
            mov     cx,[bx]             ; parameter 2)
            mov     bx,[bp + 6]         ; get second parameter
            mov     si,bx               ; si point to 2's parameter
            mov     dx,cs:buff_addr     ; move screen buffer address to dx
            mov     di,000ch            ; skip 12 bytes
```

```
trdata2:
            mov     al,[si]             ;  move
            push    ds                  ;
            push    dx                  ;      user data
            pop     ds                  ;
            mov     [di],al             ;           to
            inc     si                  ;
            inc     di                  ;              screen buffer.
            pop     ds                  ;
            loop    trdata2             ;
;
;  --- set cursor
            mov     al,06h              ; set cursor position function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            jne     error_1             ;    no, ... go error routine.
;
; --- check status before send data to host.
;
check_status:
            mov     cx,0ff0h            ; set time-out/loop counter.
loop_status:
            mov     al,08h              ; get device status function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            jne     end_st              ;    no, ... jump to loop again
            mov     ah,bh               ; move return code to ah
            and     ah,11110000B        ;
            cmp     ah,00h              ; connection to host ready ?
            je      error_2             ;    no, ... go error routine.
            mov     ah,bh               ;
            and     ah,00001111B        ;
            cmp     ah,00h              ; normal ready state ?
            je      send_key            ;    yes, ... go next step
end_st:
            loop    loop_status

error_2:
            mov     al,32h              ; error code 2
            mov     bx,[bp + 10]        ; get address of parm4
            mov     [bx],al             ; move error code to parm4
            pop     bp                  ; restore bp
            ret                         ; return to calling program
;
;------------ Send enter key --------------------
;
send_key:
            mov     bl,00h
            mov     al,07h              ; control key function.
            mov     ah,02h              ; device number 2
            mov     bh,3dh              ; ENTER key code
            int     59h                 ; interupt to emulation kernel
;
            mov     cx,0ff0h
```

```
wait_host:
            mov     al,08h          ;   get device status
            mov     ah,02h          ;       for this terminal (02)
            int     59h
            cmp     al,00h          ;   function sucessfull ?
            jne     end_wait        ;       no, .. loop again.
            mov     ah,bh           ;
            and     ah,11110000B    ;
            cmp     ah,00h          ;   communication lost ?
            je      error_2         ;      yes, ... go error routine.
            mov     ah,bh           ;   get return code
            and     ah,00001111B    ;
            cmp     ah,00h          ;   Transmission or system state ?
            jne     end_wait        ;       yes, .. loop again.
            mov     ah,bl           ;   Get flag
            and     ah,01000000B
            cmp     ah,40h          ;   Host writtten buffer ?
            je      update_screen   ;       yes, .. go get data.
end_wait:
            loop    wait_host       ;   end loop

;
; Time out error routine
;
error_3:
            mov     al,33h          ;   set error code 3
            mov     bx,[bp + 10]    ;   get address of parm4
            mov     [bx],al         ;   move error code to parm4
            pop     bp              ;   restore bp
            ret                     ;   return to calling program
;
;------------ Update screen buffer ---------------
;
update_screen:
            mov     al,07h          ;   control key function.
            mov     ah,02h          ;   device number 2
            mov     bh,16h          ;   screen buffer modify key
            int     59h             ;   interupt to emulation kernel
            cmp     al,00h          ;   Modify screen buffer O.K. ?
            je      receive         ;       yes, .. go get data.
            jmp     error_1

;
;----------------- Receive data ----------------------
;
receive:
            mov     bx,[bp + 14]        ; get length of parameter 3
            mov     cx,[bx]
            mov     bx,[bp + 8]         ; get third parameter
            mov     di,bx               ; di point to 3'nd parameter
            mov     si,0003h            ; skip 3 bytes to data area
            mov     dx,cs:buff_addr     ; move screen buffer address to dx

movdata:
            push    ds                  ;
            mov     ds,dx               ; move
```

```
            mov     al,[si]             ;     output data
            pop     ds                  ;        from host
            mov     [di],al             ;
            inc     si                  ;        to output area
            inc     di                  ;           in main program
            loop    movdata             ;

;
back:
;
;--------------- Transaction completed ----------------------
;
            mov     al,30h              ; move o to return code byte
            mov     bx,[bp + 10]        ; ..
            mov     [bx],al             ; ..

return:
            pop     bp
            ret

_API_C      ENDP
;-------------------------------------------------------------------
;                    Data and Parameter area
;-------------------------------------------------------------------
;
;---------- Buffer Address
;
buff_addr       dw      0
buff_disp       dw      0
stack_seg       dw      0
bp_save         dw      0
save_bp         dw      0
;
;=============================================================================
last        db      0
_TEXT       ENDS
            end
```

ภาคผนวก ง.

โปรแกรมเชื่อมโยงบนไมโครคอมพิวเตอร์สำหรับภาษาโคบอล

```
          include userseg.mac
          page
DYNS      STRUC
          DW      ?
          DD      ?
parm6     DW      ?
parm5     DW      ?
parm4     DW      ?
parm3     DW      ?
parm2     DW      ?
parm1     DW      ?
DYNS      ENDS
;
;
START_CSEG API_COB

          push    bp                ; save location of bp
          mov     bp,sp             ; move stack pointer to bp

;
;---------- Allocate device no.2 ----------------------
;
          mov     al,01h            ; Allocate device function.
          mov     ah,10000010b      ; Device number 2
          int     59h               ; Interupt to emulation kernel

;
;----------- Get buffer address -----------------------
;

          mov     al,03h            ; Get buffer address function.
          mov     ah,02h            ; Device number 2
          int     59h               ; Interupt to emulation kernel.
          cmp     al,00h            ; Function successful ?
          je      buffer_ok         ;    yes, ... go next step.
error_1:
          mov     al,31h            ;    no, .. then
          mov     bx,[bp].parm4     ;           move error code
          mov     [bx],al           ;           to parm4
          pop     bp                ; restore bp
          ret                       ; return to calling program.
;
buffer_ok:
          mov     cs:buff_addr,es   ; save screen buffer address
          mov     cs:buff_disp,di   ;   from es:di to work area.

;------------- Clear screen ----------------------
;

          mov     al,07h            ; control key entry function
          mov     ah,02h            ; device number 2
          mov     bh,2dh            ; CLEAR key
          int     59h               ; interupt to emulation kernel

;
;----------------- Move data to screen buffer ----------------------
```

```
;
move_tranid:                            ; Trans_id = HAPI

            mov     dx,cs:buff_addr     ;
            mov     di,0000h            ;

            push    ds                  ; save ds
            push    dx                  ;
            pop     ds                  ; move dx to ds
;
            mov     al,48h              ; H       48
            mov     [di],al
            inc     di
            mov     al,41h              ; A       41
            mov     [di],al
            inc     di
            mov     al,50h              ; P       50
            mov     [di],al
            inc     di
            mov     al,49h              ; I       49
            mov     [di],al
            inc     di
            pop     ds

;
move_program:                           ; Host user program name

            mov     bx,[bp].parm1       ; get first parameter
            mov     cx,0008h            ; parameter length = 8
            mov     si,bx               ; si point to parm1
            mov     dx,cs:buff_addr     ; move screen buffer address to dx
            mov     di,0004h            ; skip 4 bytes

trdata1:
            mov     al,[si]             ; move
            push    ds                  ;
            push    dx                  ;   parm1
            pop     ds                  ;
            mov     [di],al             ;       to
            inc     si                  ;
            inc     di                  ;           screen buffer.
            pop     ds
            loop    trdata1             ;

;
move_content:                               ; User Data

            mov     bx,[bp].parm5
            mov     cx,[bx]
            xchg    ch,cl
            mov     bx,[bp].parm2       ; get second parameter
            mov     si,bx               ; si point to 2's parameter
            mov     dx,cs:buff_addr     ; move screen buffer address to dx
            mov     di,000ch            ; skip 12 bytes

trdata2:
```

```
            mov     al,[si]             ;  move
            push    ds                  ;
            push    dx                  ;      user data
            pop     ds                  ;
            mov     [di],al             ;           to
            inc     si                  ;
            inc     di                  ;              screen buffer.
            pop     ds                  ;
            loop    trdata2             ;
;
;  --- set cursor
            mov     al,06h              ; set cursor position function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            je      check_status        ;    no, ... go error routine.
            mov     al,31h
            mov     bx,[bp].parm4
            mov     [bx],al
            pop     bp
            ret
;
; --- check status before send data to host.
;
check_status:
            mov     cx,0ff0h            ; set time-out/loop counter.
loop_status:
            mov     al,08h              ; get device status function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            jne     end_st              ;    no, ... jump to loop again
            mov     ah,bh               ; move return code to ah
            and     ah,11110000B        ;
            cmp     ah,00h              ; connection to host ready ?
            je      error_2             ;    no, ... go error routine.
            mov     ah,bh               ;
            and     ah,00001111B        ;
            cmp     ah,00h              ; normal ready state ?
            je      send_key            ;    yes, ... go next step
end_st:
            loop    loop_status

error_2:
            mov     al,32h              ; error code 2
            mov     bx,[bp].parm4       ; get address of parm4
            mov     [bx],al             ; move error code to parm4
            pop     bp                  ; restore bp
            ret                         ; return to calling program
;
;------------ Send enter key --------------------
;
send_key:
            mov     bl,00h
            mov     al,07h              ; control key function.
            mov     ah,02h              ; device number 2
```

```
                mov     bh,3dh              ; ENTER key code
                int     59h                 ; interupt to emulation kernel
;
                mov     cx,0ff0h
wait_host:
                mov     al,08h              ;   get device status
                mov     ah,02h              ;       for this terminal (02)
                int     59h
                cmp     al,00h              ;   function sucessfull ?
                jne     end_wait            ;       no, .. loop again.
                mov     ah,bh               ;
                and     ah,11110000B        ;
                cmp     ah,00h              ;   communication lost ?
                je      error_2             ;      yes, ... go error routine.
                mov     ah,bh               ;   get return code
                and     ah,00001111B        ;
                cmp     ah,00h              ;   Transmission or system state ?
                jne     end_wait            ;       yes, .. loop again.
                mov     ah,bl               ;   Get flag
                and     ah,01000000B
                cmp     ah,40h              ;   Host writtten buffer ?
                je      update_screen       ;       yes, .. go get data.
end_wait:
                loop    wait_host           ;   end loop

;
; Time out error routine
;
error_3:
                mov     al,33h              ;  set error code 3
                mov     bx,[bp].parm4       ;  get address of parm4
                mov     [bx],al             ;  move error code to parm4
                pop     bp                  ;  restore bp
                ret                         ;  return to calling program
;
;------------ Update screen buffer ---------------
;
update_screen:
                mov     al,07h              ;  control key function.
                mov     ah,02h              ;  device number 2
                mov     bh,16h              ;  screen buffer modify key
                int     59h                 ;  interupt to emulation kernel
                cmp     al,00h              ;  Modify screen buffer O.K. ?
                je      receive             ;      yes, .. go get data.
                jmp     error_1

;
;----------------- Receive data -----------------------
;
receive:
                mov     bx,[bp].parm6
                mov     cx,[bx]
                xchg    ch,cl
                mov     bx,[bp].parm3       ; get third parameter
;               mov     cx,00h              ;
;               mov     cl,132              ;
```

```
            mov     di,bx               ; di point to 3'nd parameter
            mov     si,0002h            ; skip 3 bytes to data area
            mov     dx,cs:buff_addr     ; move screen buffer address to dx

movdata:
            push    ds                  ;
            mov     ds,dx               ; move
            mov     al,[si]             ;     output data
            pop     ds                  ;        from host
            mov     [di],al             ;
            inc     si                  ;        to output area
            inc     di                  ;           in main program
            loop    movdata             ;

;
back:
;
;--------------- Transaction completed ----------------------
;
            mov     al,30h              ; move o to return code byte
            mov     bx,[bp].parm4       ; ..
            mov     [bx],al             ; ..

return:
            pop     bp
            ret

;-------------------------------------------------------------
;                   Data and Parameter area
;-------------------------------------------------------------
;
;---------- Buffer Address
;
buff_addr       dw      0
buff_disp       dw      0
stack_seg       dw      0
bp_save         dw      0
save_bp         dw      0
;
;------------------- Error messages ---------------------------

buffer_error    db      'Buffer Address unavailable $'
buffer_found    db      'Buffer Address allocate $'
apinstal_ok     db      'AST API installed $'
cursor_error    db      'Set cursor error $'
send_error      db      'Send data error $'
;=============================================================================


last            db      0
END_CSEG        API_COB
                end
```

ภาคผนวก จ.

โปรแกรม เชื่อมโยงบนไมโครคอมพิว เตอร์สำหรับภาษา เบสิค

```
          page


TEXT      SEGMENT                       ; begin program segment
          ASSUME  CS:TEXT

          org     100h

          push    bp                    ; save location of bp
          mov     bp,sp                 ; move stack pointer to bp
          push    es
          push    ds
;
;---------- Allocate device no.2 ----------------------
;
          mov     al,01h                ; Allocate device function.
          mov     ah,10000010b          ; Device number 2
          int     59h                   ; Interupt to emulation kernel

;
;----------- Get buffer address -----------------------
;

          mov     al,03h                ; Get buffer address function.
          mov     ah,02h                ; Device number 2
          int     59h                   ; Interupt to emulation kernel.
          cmp     al,00h                ; Function successful ?
          je      buffer_ok             ;    yes, ... go next step.
error_1:
          mov     cx,1                  ;    no, .. then
          lds     di,[bp + 6h]          ;           move error code
          mov     ds:[di],cx            ;           to parm4
          pop     ds                    ; restore ds
          pop     es                    ; restore es
          pop     bp                    ; restore bp
          ret                           ; return to calling program.
;
buffer_ok:
          mov     cs:buff_addr,es       ; save screen buffer address
          mov     cs:buff_disp,di       ;   from es:di to work area.

;------------- Clear screen ----------------------
;

          mov     al,07h                ; control key entry function
          mov     ah,02h                ; device number 2
          mov     bh,2dh                ; CLEAR key
          int     59h                   ; interupt to emulation kernel

;
;----------------- Move data to screen buffer ---------------------
;
move_tranid:                            ; Trans_id = HAPI

          mov     dx,cs:buff_addr       ;
          mov     di,0000h              ;
```

```
                push    ds                      ; save ds
                push    dx                      ;
                pop     ds                      ; move dx to ds
;
                mov     al,48h                  ; H       48
                mov     [di],al
                inc     di
                mov     al,41h                  ; A       41
                mov     [di],al
                inc     di
                mov     al,50h                  ; P       50
                mov     [di],al
                inc     di
                mov     al,49h                  ; I       49
                mov     [di],al
                inc     di
                pop     ds

;
move_program:                                   ; Host user program name

                push    ds
                push    es
                les     di,[bp + 12h]           ; load pointer of string descriptor
                                                ; into es:di (parm1)
                mov     dx,ds:[0]               ; get begining of string segment
                push    dx                      ; save dx
                pop     ds                      ; make ds point to string segment
                mov     cx,0008h                ; parameter 1 length = 8
                mov     si,es:[di + 2]          ; get offset of string segment
                mov     es,cs:buff_addr         ; move screen buffer address to dx
                mov     di,0004h                ; skip 4 bytes on screen buffer

trdata1:
                mov     al,[si]                 ; move
                mov     es:[di],al              ;       to
                inc     si                      ;
                inc     di                      ;          screen buffer.
                loop    trdata1                 ;
                pop     es
                pop     ds

;
move_content:                                   ; User Data

                push    ds
                push    es
                les     di,[bp + 0eh]           ; load pointer of string descripter
                                                ; into es:di
                mov     dx,ds:[0]
                push    dx
                pop     ds
                mov     cx,es:[di]              ; get length of parm2
                mov     si,es:[di + 2]          ; si point to 2's parameter
                mov     es,cs:buff_addr         ; move screen buffer address to dx
```

```
            mov     di,000ch            ; skip 12 bytes on screen buffer

trdata2:
            mov     al,[si]             ;  move
            mov     es:[di],al          ;          to
            inc     si                  ;
            inc     di                  ;             screen buffer.
            loop    trdata2             ;
            pop     es
            pop     ds
;
;  --- set cursor
            mov     al,06h              ; set cursor position function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            je      check_status        ;    no, ... go error routine.
            mov     cx,1
            lds     di,[bp + 6]
            mov     es:[di],cx
            pop     ds
            pop     es
            ret
;
; --- check status before send data to host.
;
check_status:
            mov     cx,0ff0h            ; set time-out/loop counter.
loop_status:
            mov     al,08h              ; get device status function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            jne     end_st              ;    no, ... jump to loop again
            mov     ah,bh               ; move return code to ah
            and     ah,11110000B        ;
            cmp     ah,00h              ; connection to host ready ?
            je      error_2             ;    no, ... go error routine.
            mov     ah,bh               ;
            and     ah,00001111B        ;
            cmp     ah,00h              ; normal ready state ?
            je      send_key            ;    yes, ... go next step
end_st:
            loop    loop_status

error_2:
            mov     cx,2                ; error code 2
            lds     di,[bp + 6h]        ; get address of parm4
            mov     ds:[di],cx          ; move error code to parm4
            pop     ds                  ; restore ds
            pop     es                  ; restore es
            pop     bp                  ; restore bp
            ret                         ; return to calling program
;
;------------ Send enter key --------------------
;
```

```
send_key:
            mov     bl,00h
            mov     al,07h              ; control key function.
            mov     ah,02h              ; device number 2
            mov     bh,3dh              ; ENTER key code
            int     59h                 ; interupt to emulation kernel
;
            mov     cx,0ff0h
wait_host:
            mov     al,08h          ;   get device status
            mov     ah,02h          ;       for this terminal (02)
            int     59h
            cmp     al,00h          ;   function sucessfull ?
            jne     end_wait        ;       no, .. loop again.
            mov     ah,bh           ;
            and     ah,11110000B    ;
            cmp     ah,00h          ;   communication lost ?
            je      error_2         ;      yes, ... go error routine.
            mov     ah,bh           ;   get return code
            and     ah,00001111B    ;
            cmp     ah,00h          ;   Transmission or system state ?
            jne     end_wait        ;       yes, .. loop again.
            mov     ah,bl           ;   Get flag
            and     ah,01000000B
            cmp     ah,40h          ;   Host writtten buffer ?
            je      update_screen   ;       yes, .. go get data.
end_wait:
            loop    wait_host       ;   end loop

;
; Time out error routine
;
error_3:
            mov     cx,3            ;  set error code 3
            lds     di,[bp + 6h]    ;  get address of parm4
            mov     ds:[di],cx      ;  move error code to parm4
            pop     ds              ;  restore ds
            pop     es              ;  restore es
            pop     bp              ;  restore bp
            ret                     ;  return to calling program
;
;------------ Update screen buffer ---------------
;
update_screen:
            mov     al,07h          ;  control key function.
            mov     ah,02h          ;  device number 2
            mov     bh,16h          ;  screen buffer modify key
            int     59h             ;  interupt to emulation kernel
            cmp     al,00h          ;  Modify screen buffer O.K. ?
            je      receive         ;      yes, .. go get data.
            mov     cx,1
            lds     di,[bp + 6h]
            mov     ds:[di],cx
            pop     ds
            pop     es
            pop     bp
```

```
;
;----------------- Receive data ----------------------
;
receive:
            push    ds
            push    es
            les     di,[bp + 0ah]       ; get pointer to string descriptor
            mov     dx,ds:[0]
            push    dx
            pop     ds
            mov     cx,es:[di]          ; get length of parm3
            mov     di,es:[di + 2]      ; di point to 3'nd parameter
            mov     si,0002h            ; skip 3 bytes to data area
            mov     es,cs:buff_addr     ; move screen buffer address to dx

movdata:
            mov     al,es:[si]          ;     output data
            mov     [di],al             ;
            inc     si                  ;        to output area
            inc     di                  ;           in main program
            loop    movdata             ;
            pop     es
            pop     ds
;
back:
;
;--------------- Transaction completed ----------------------
;
            mov     cx,0                ; move o to return code byte
            lds     di,[bp + 6h]        ; get 4's parameter
            mov     ds:[di],cx          ; move completed code 0 to parm4

return:
            pop     ds                  ; restore ds
            pop     es                  ; restore es
            pop     bp                  ; restore bp
            ret                         ; return to calling program

;-------------------------------------------------------------
;                   Data and Parameter area
;-------------------------------------------------------------
;
;---------- Buffer Address
;
buff_addr       dw      0
buff_disp       dw      0
stack_seg       dw      0
bp_save         dw      0
save_bp         dw      0
;
;=============================================================================
last        db      0
TEXT        ENDS
            end
```

ภาคผนวก ฉ.

โปรแกรมเชื่อมโยงบนไมโครคอมพิวเตอร์สำหรับภาษาแอสเซมบลี

```
; ------------------------------------------------
; This is the Interface Program for assembly language
; ------------------------------------------------
          EXTRN   parm1:byte,parm2:byte,parm3:byte
          EXTRN   parm4:byte,parm5:word,parm6:word
;
CODESEG   SEGMENT PARA PUBLIC 'CODE'
APIASM    PROC    FAR
          ASSUME  CS:CODESEG
          PUBLIC  APIASM
;
          push    bp                    ; save location of bp
          mov     bp,sp                 ; move stack pointer to bp
          push    es
;
;---------- Allocate device no.2 ----------------------
;
          mov     al,01h                ; Allocate device function.
          mov     ah,10000010b          ; Device number 2
          int     59h                   ; Interupt to emulation kernel

;
;----------- Get buffer address -----------------------
;

          mov     al,03h                ; Get buffer address function.
          mov     ah,02h                ; Device number 2
          int     59h                   ; Interupt to emulation kernel.
          cmp     al,00h                ; Function successful ?
          je      buffer_ok             ;    yes, ... go next step.
error_1:
          mov     al,1                  ;    no, .. then
          mov     parm4,al              ;           to parm4
          pop     es
          pop     bp                    ; restore bp
          ret                           ; return to calling program.
;
buffer_ok:
          mov     cs:buff_addr,es       ; save screen buffer address
          mov     cs:buff_disp,di       ;   from es:di to work area.

;------------- Clear screen ----------------------
;

          mov     al,07h                ; control key entry function
          mov     ah,02h                ; device number 2
          mov     bh,2dh                ; CLEAR key
          int     59h                   ; interupt to emulation kernel

;
;----------------- Move data to screen buffer --------------------
;
move_tranid:                            ; Trans_id = HAPI

           mov     es,cs:buff_addr      ;
           mov     di,0000h             ;
```

```
            mov     al,48h              ; H       48
            mov     es:[di],al
            inc     di
            mov     al,4Fh              ; O       4F
            mov     es:[di],al
            inc     di
            mov     al,53h              ; S       53
            mov     es:[di],al
            inc     di
            mov     al,54h              ; T       54
            mov     es:[di],al
            inc     di

;
move_program:                           ; Host user program name

            lea     bx,parm1            ; get first parameter
            mov     cx,0008h            ; parameter length = 8
            mov     si,bx               ; si point to parm1
            mov     es,cs:buff_addr     ; move screen buffer address to dx
            mov     di,0004h            ; skip 4 bytes

trdata1:
            mov     al,[si]             ; move
            mov     es:[di],al            ;       to
            inc     si                  ;
            inc     di                  ;           screen buffer.
            loop    trdata1             ;

;
move_content:                             ; User Data

            lea     bx,parm2            ; get address parameter 2
            mov     cx,parm5            ; get data length
            mov     si,bx               ; si point to 2's parameter
            mov     es,cs:buff_addr     ; move screen buffer address to dx
            mov     di,000ch            ; skip 12 bytes

trdata2:
            mov     al,[si]             ;  move
            mov     es:[di],al            ;         to
            inc     si                  ;
            inc     di                  ;             screen buffer.
            loop    trdata2             ;
;
;  --- set cursor
            mov     al,06h              ; set cursor position function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            je      check_status        ;    no, ... go error routine.
            mov     al,1
            mov     parm4,al
            pop     es
            pop     bp
```

```
            ret
;
; --- check status before send data to host.
;
check_status:
            mov     cx,0ff0h            ; set time-out/loop counter.
loop_status:
            mov     al,08h              ; get device status function
            mov     ah,02h              ; device number 2
            int     59h                 ; interupt to emulation kernel
            cmp     al,00h              ; function successfull ?
            jne     end_st              ;    no, ... jump to loop again
            mov     ah,bh               ; move return code to ah
            and     ah,11110000B        ;
            cmp     ah,00h              ; connection to host ready ?
            je      error_2             ;    no, ... go error routine.
            mov     ah,bh               ;
            and     ah,00001111B        ;
            cmp     ah,00h              ; normal ready state ?
            je      send_key            ;    yes, ... go next step
end_st:
            loop    loop_status

error_2:
            mov     al,2                ; error code 2
            mov     parm4,al            ; get address of parm4
            pop     es
            pop     bp                  ; restore bp
            ret                         ; return to calling program
;
;------------ Send enter key --------------------
;
send_key:
            mov     bl,00h
            mov     al,07h              ; control key function.
            mov     ah,02h              ; device number 2
            mov     bh,3dh              ; ENTER key code
            int     59h                 ; interupt to emulation kernel
;
            mov     cx,0ff0h
wait_host:
            mov     al,08h          ;   get device status
            mov     ah,02h          ;       for this terminal (02)
            int     59h
            cmp     al,00h          ;   function sucessfull ?
            jne     end_wait        ;       no, .. loop again.
            mov     ah,bh           ;
            and     ah,11110000B    ;
            cmp     ah,00h          ;   communication lost ?
            je      error_2         ;      yes, ... go error routine.
            mov     ah,bh           ;   get return code
            and     ah,00001111B    ;
            cmp     ah,00h          ;   Transmission or system state ?
            jne     end_wait        ;       yes, .. loop again.
            mov     ah,bl           ;   Get flag
            and     ah,01000000B
```

```
            cmp     ah,40h          ;   Host writtten buffer ?
            je      update_screen   ;       yes, .. go get data.
end_wait:
            loop    wait_host       ;   end loop

;
; Time out error routine
;
error_3:
            mov     al,3            ;  set error code 3
            mov     parm4,al        ;  get address of parm4
            pop     es
            pop     bp              ;  restore bp
            ret                     ;  return to calling program
;
;------------ Update screen buffer ---------------
;
update_screen:
            mov     al,07h          ;  control key function.
            mov     ah,02h          ;  device number 2
            mov     bh,16h          ;  screen buffer modify key
            int     59h             ;  interupt to emulation kernel
            cmp     al,00h          ;  Modify screen buffer O.K. ?
            je      receive         ;      yes, .. go get data.
            jmp     error_1

;
;----------------- Receive data ----------------------
;
receive:
            mov     cx,parm6            ; get length of parm3
            lea     bx,parm3            ; get 3's parameter
            mov     di,bx               ; di point to 3'nd parameter
            mov     si,0001h            ; skip 3 bytes to data area
            mov     es,cs:buff_addr     ; move screen buffer address to dx

movdata:
            mov     al,es:[si]          ;     output data
            mov     [di],al             ;
            inc     si                  ;        to output area
            inc     di                  ;           in main program
            loop    movdata             ;

;
back:
;
;--------------- Transaction completed ----------------------
;
            mov     al,0                ; move o to return code byte
            mov     parm4,al            ; ..
;           mov     [bx],cx             ; ..

return:
            pop     es
            pop     bp
            ret
```

```
APIASM      ENDP
;-------------------------------------------------------------
;                    Data and Parameter area
;-------------------------------------------------------------
;
;---------- Buffer Address
;
buff_addr       dw      0
buff_disp       dw      0
stack_seg       dw      0
bp_save         dw      0
save_bp         dw      0
;
;=============================================================================
last            db      0
CODESEG     ENDS
            end
```

ภาคผนวก ช.

โปรแกรม เชื่อมโยงบน เมน เฟรม

```
         PRINT ON,NOGEN
DFHEISTG DSECT
         EJECT
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*   INPUT BUFFER AREA FOR RECEIVE DATA FROM TERMINAL    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
INAREA   DS    0CL1920
TRANID   DS    CL4             TRANSACTION ID
PROGID   DS    CL8             HOST APPLICATION PROGRAM NAME
INDATA   DS    CL1908          INPUT DATA FOR APP. PROG. ON HOST
* * * * * *  * * * * * * * * * * * * * * * * * * * * * * * *
*   OUTPUT BUFFER AREA FOR RECEIVE DATA FROM TERMINAL    *
* * * * * *  * * * * * * * * * * * * * * * * * * * * * * * *
OUTAREA  DS    0CL1920
RTCODE   DS    CL1           RETURN CODE TO PC
OUTDATA  DS    CL1919        MAX. OUTPUT SEND TO PC
INLEN    DS    H             MAX. LENGTH OF INPUT FROM PC
COMMLEN  DS    H             COMMUNICATION AREA LENGTH
*
FLEN     EQU   *-INAREA
         DS    CL3830
* * * * * *  * * * * * * * * * * * * * * * * * * * * * * * *
*   COMMUNICATION AREA FOR SEND & RECEIVE DATA BETWEEN   *
*                MAIN PROGRAM AND SUBPROGRAM             *
* * * * * *  * * * * * * * * * * * * * * * * * * * * * * * *
*COMMAREA DS    0CL3827
COMMAREA DSECT
         USING COMMAREA,R4
COMMIN   DS    CL1908        INPUT TO SUBPROGRAM
COMMOUT  DS    CL1919        OUTPUT FROM SUBPROGRAM
OUTLEN   DS    H             LENGTH OF OUTPUT
*
*
HOSTAPI  DFHEIENT DATAREG(R13)
*
         LA    R4,INAREA
         LA    R4,FLEN(R4)
         MVI   RTCODE,C'A'              SET ACCEPT RETURN CODE
         MVC   INLEN,=H'1920'           MAX. INPUT LENGTH = 1920 BYTES
* ------------------------------------------------------------- *
```

```
*   SET HANDLE CONDITION COMMAND FOR CHECK                          *
*       1. INPUT LENGTH FROM PC NOT OVER 1920 BYTES                 *
*       2. HOST PROGRAM NAME THAT SEND FROM PC ALREADY ON CICS/VS   *
* ------------------------------------------------------------------ *
*
        EXEC CICS HANDLE CONDITION                                   X
              LENGERR(LENGERR)                                       X
              PGMIDERR(NOPROG)
*
* ------------------------------------------------------------------ *
*   RECEIVE DATA FROM PC AND PUT DATA INTO WORK AREA (INAREA)        *
* ------------------------------------------------------------------ *
*
        EXEC CICS RECEIVE INTO(INAREA)                               X
              LENGTH(INLEN)
*
*       MVC   COMMIN,INDATA          MOVE INPUT DATA TO COMM. INPUT
        LA    R6,COMMIN
        L     R7,=F'1908'
        LA    R8,INDATA
        L     R9,=F'1908'
        MVCL  R6,R8
        MVC   COMMLEN,=H'3827'       SET COMM. AREA LENGTH = 3827
*
* ------------------------------------------------------------------ *
*   CALL SUBROUTINE - NAME OF SUBROUTINE IS VALUE OF HOST PROGRAM    *
*                     NAME THAT SET BY PROGRAM ON PC                 *
* ------------------------------------------------------------------ *
        EXEC CICS LINK PROGRAM(PROGID)                               X
              COMMAREA(COMMAREA)                                     X
              LENGTH(COMMLEN)
*
        XR    R5,R5
        LH    R5,OUTLEN
        LA    R5,1(R5)
        STH   R5,OUTLEN
*       MVC   OUTDATA,COMMOUT        MOVE COMM. OUTPUT TO OUTPUT AREA
        LA    R6,OUTDATA
        L     R7,=F'1919'
        LA    R8,COMMOUT
        L     R9,=F'1919'
        MVCL  R6,R8
*
RETURNPC DS   0H
*
* ------------------------------------------------------------------ *
*   SEND DATA THAT RECEIVE FROM SUBPROGRAM TO PC                     *
* ------------------------------------------------------------------ *
        EXEC CICS SEND FROM(OUTAREA)                                 X
              LENGTH(OUTLEN)  ERASE
*
        EXEC CICS RETURN
* * * * * *  * * * * * * * * * *
*   THIS IS ERROR ROUTINE      *
* * * * * *  * * * * * * * * * *
LENGERR  DS   0H
```

```
         MVC   OUTDATA(4),=C'E900'
         MVC   OUTLEN,=H'6'
         B     RETURNPC
NOPROG   DS    0H
         MVC   OUTDATA(4),=C'E901'
         MVC   OUTLEN,=H'6'
         B     RETURNPC
*
         LTORG
         END
```

ภาคผนวก ช.

ตัวอย่างของ Network Control Program บน IBM 3705

```
NCP0A0     PCCU CUADDR=0A0,                                              X
                AUTODMP=NO,                                              X
                AUTOIPL=YES,                                             X
                AUTOSYN=YES,                                             X
                DUMPDS=SYS005,                                           X
                MAXDATA=3000,                                            X
                NCPLUB=SYS012,                                           X
                OWNER=SSCP00,                                            X
                INITEST=NO,                                              X
                SUBAREA=13
NCPTEST    BUILD  ABEND=YES,                                             X
                MAXSUBA=25,                                              X
                MEMSIZE=256,                                             X
                SUBAREA=12,                                              X
                TYPGEN=NCP,                                              X
                ANS=YES,                                                 X
                ASMXREF=NO,                                              X
                BFRS=128,                                                X
                CA=(TYPE4),                                              X
                NCPCA=(ACTIVE),                                          X
                CATRACE=(YES,10),                                        X
                ITEXTTO=60,                                              X
                JOBCARD=YES,                                             X
                MODEL=3705-2,                                            X
                NEWNAME=NCP05M,                                          X
                PARTIAL=NO,                                              X
                PRTGEN=(NOGEN,NOGEN),                                    X
                SLODOWN=12,                                              X
                TRACE=(YES,10),                                          X
                TYPSYS=DOS
       SYSCNTRL OPTIONS=(ENDCALL,MODE,RCNTRL,RIMM,NAKLIM,                X
                 BHSASSC,RECMD,RCOND,SESSION,SSPAUSE,XMTLMT,STORDSP)
       HOST     INBFRS=24,                                               X
                 MAXBFRU=40,                                             X
                 UNITSZ=216,                                             X
                 BFRPAD=0,                                               X
                 SUBAREA=13
       CSB      SPEED=(134),                                             X
                 MOD=0,                                                  X
                 TYPE=TYPE2
*
* ------------------------------------ *
* ***  FOR 4700,ATM,PC - SDLC LINE *** *
* ------------------------------------ *
*
NCPSDLC    GROUP  LNCTL=SDLC,                                            X
                REPLYTO=5,                                               *
                PACING=(1,1),                                            *
                VPACING=(2,1)
* ------------------------------------ *
* ***  AST-SNA 3270  -       SDLC LINE  *** *
* ------------------------------------ *
PC01    LINE        ADDRESS=038,                                         *
                SPEED=2400,                                              *
                CLOCKNG=EXT,                                             *
                DUPLEX=HALF,                                             *
```

```
                NRZI=NO,                                              *
                POLLED=YES
        SERVICE ORDER=(PUPC)
PUPC        PU  ADDR=C1,                                              *
                MAXDATA=265,                                          *
                PACING=0,                                             *
                VPACING=0,                                            *
                ANS=CONTINUE,                                         *
                SSCPFM=USSSCS,                                        *
                DISCNT=NO,                                            *
                MAXOUT=7,                                             *
                PASSLIM=8,                                            *
                PUTYPE=2,                                             *
                RETRIES=(,4,5),                                       *
                ISTATUS=INACTIVE
LUPC01      LU  LOCADDR=2,                                            *
                LOGAPPL=DBDCCICS,                                     *
                ISTATUS=INACTIVE
*
            GENEND
```

ภาคผนวก ฌ.

ตัวอย่าง Terminal Control Program (TCT)

```
TCT      TITLE 'CICS/VS TERMINAL CONTROL TABLE  (TCT) '
         PRINT NOGEN
         DFHTCT TYPE=INITIAL,              REQUIRED OPERAND                *
               ACCMETH=(NONVTAM,VTAM),     ACCESS METHODS USED             *
               APPLID=UBSAFE,              APPLICATION ID                  *
               ERRATT=NO,                  FOR 3270 ERROR MESS.AT CUR.CURSO
               GMTEXT='READY-TO-ONLINE',   SIGN ON MESSAGE (GMMSG=YES)     *
               RAMAX=256,                  RECV ANY I/O AREA SIZE, BYTES   *
               RAMIN=80,                   DATA LEN TO XFER TO NEW TIOA    *
               RAPOOL=12,                  NO. OF FIXED RPLS IN TCT PREFIX*
               RATIMES=4,                  MAX NO. OF TIOA'S PER MSG       *
               SYSIDNT=UBHO,               MAX NO. OF TIOA'S PER MSG       *
               RESP=FME,                   RESPONSE REQUESTED (FOR 3600)   *
               SUFFIX=J1
*  **********************
*    FOR CONSOLE
*  **********************
         DFHTCT TYPE=SDSCI,DEVICE=CONSOLE
         DFHTCT TYPE=LINE,ACCMETH=SEQUENTIAL,INAREAL=80,                    *
               TRMTYPE=CONSOLE
         DFHTCT TYPE=TERMINAL,TRMIDNT=CNSL,FEATURE=(UCTRAN),                *
               TRMSTAT=TRANSCEIVE,TIOAL=256
*  **********************
*       FOR LOCAL TER.
*  **********************
         DFHTCT TYPE=TERMINAL,                                              X
               TCTUAL=255,                                                  X
               TRMIDNT=L1T1,                                                X
               NETNAME=UBTER01,                                             X
               CONNECT=AUTO,                                                X
               ACCMETH=VTAM,                                                X
               TRMTYPE=L3277,                                               X
               TRMMODL=2,                                                   X
               TIOAL=270,                                                   X
               TRMSTAT=TRANSCEIVE,                                          X
               RELREQ=(YES,YES),                                            X
               FEATURE=(COPY,DCKYBD,SELCTPEN,UCTRAN,AUDALARM),              X
               PGESIZE=(24,80),                                             X
               PRINTTO=(P42C),                                              X
               PGESTAT=PAGE
*
*  ------------------------------------------------
*  FOR PC TERMINAL USING AST-SNA CARD (SDLC PROTOCOL)
*  ------------------------------------------------
         DFHTCT TYPE=TERMINAL,                                              X
               BUFFER=1536,                                                 X
               RUSIZE=256,                                                  X
               TRMIDNT=PC01,                                                X
               NETNAME=LUPC01,                                              X
               CONNECT=AUTO,                                                X
               ACCMETH=VTAM,                                                X
               TRMTYPE=LUTYPE2,                                             X
               TRMMODL=2,                                                   X
               CLASS=CONV,                                                  X
               TRMSTAT=TRANSCEIVE,                                          X
               FEATURE=AUDALARM,                                            X
```

```
         CHNASSY=YES,                                          X
         TIOAL=(1024,4096)                                     X
         DEFSCRN=(24,80)
DFHTCT TYPE=FINAL
END    DFHTCTBA
```

ภาคผนวก ญ.

ตัวอย่าง Program Control Table (PCT)

```
PCT      TITLE 'DFHPCT CICS/VS PROGRAM CONTROL TABLE '
         PRINT  NOGEN
         DFHPCT TYPE=INITIAL,                                          X
               DTB=NO,                  DYNAMIC TRANS. BACKOUT FAC.    X
               EXTSEC=NO,               EXTERNAL SECURITY FAC.         X
               FDUMP=(ASRA,ASRB),       FORMATTED DUMP                 X
               INDEX=YES,               INDEXING TO BE USED            X
               SCRNSZE=DEFAULT,         RUN IN DEFAULT SCREEN SIZE MODE X
               SUFFIX=J1                SUFFIX__    **********
* ------------------------------------------------------------------- *
*  DEFINE TRANSACTION ID FOR NAME OF INTERFACE PROGRAM ON MAINFRAME   *
* ------------------------------------------------------------------- *
         DFHPCT TYPE=ENTRY,PROGRAM=HOSTAPI,TRANSID=HAPI
* -------------------------------------------------------------------
*
* ------------------------------------------------------------------- *
*  DEFINE SYSTEM GROUP OF CICS/VS
* ------------------------------------------------------------------- *
INTER    DFHPCT TYPE=GROUP,                                            X
               FN=INTERPRETER
HARDCOPY DFHPCT TYPE=GROUP,                                            X
               FN=HARDCOPY              3270 PRINT FUN. TRANSID=CSPP
SIGNON   DFHPCT TYPE=GROUP,                                            X
               FN=SIGNON                SIGN ON/OFF
CONSOLE  DFHPCT TYPE=GROUP,                                            X
               FN=CONSOLE               WRITE TO CPU CONSOLE
MSWITCH  DFHPCT TYPE=GROUP,                                            X
               FN=MSWITCH               MSG SW TRANSACTION
BMS      DFHPCT TYPE=GROUP,                                            X
               FN=BMS,                  TERM PAGE RETRIEVAL
FE       DFHPCT TYPE=GROUP,                                            X
               FN=FE                    FE TERMINAL TEST
MASTTERM DFHPCT TYPE=GROUP,                                            X
               FN=MASTTERM              OPERATOR TERMINAL
JOURNAL  DFHPCT TYPE=GROUP,                                            X
               FN=JOURNAL               LOG/JOURNAL TRANSACTION
AKP      DFHPCT TYPE=GROUP,                                            X
               FN=AKP                   ACTIVITY KEYPOINT
AUTOSTAT DFHPCT TYPE=GROUP,                                            X
               FN=AUTOSTAT              AUTO STATISTICS
STANDARD DFHPCT TYPE=GROUP,                                            X
               FN=STANDARD              STANDARD ENTRIES
TIME     DFHPCT TYPE=GROUP,                                            X
               FN=TIME                  TIME ADJUST
NUMERICS DFHPCT TYPE=GROUP,                                            *
               FN=NUMERICS              NUMERICS SIGNON GROUP
ATP      DFHPCT TYPE=GROUP,                                            *
               FN=ATP                   ASYNCHRONOUS PROC GROUP
VTAMPRT  DFHPCT TYPE=GROUP,                                            *
               FN=VTAMPRT               PRINT REQUEST
RESPLOG  DFHPCT TYPE=GROUP,                                            *
               FN=RESPLOG               RESPONSE LOGGER
VTAM     DFHPCT TYPE=GROUP,                                            *
               FN=VTAM                  VTAM GROUP
RESEND   DFHPCT TYPE=GROUP,                                            *
               FN=RESEND                RESEND PGM, FOR SOME VTAM
```

```
ISC      DFHPCT TYPE=GROUP,                                    *
               FN=ISC                  INTER SYSTEMS COUPLING GROUP
OPERATOR DFHPCT TYPE=GROUP,                                    *
               FN=OPERATORS            ..........................
EDF      DFHPCT TYPE=GROUP,                                    *
               FN=EDF                  ..........................
         DFHPCT TYPE=FINAL
         END DFHPCTBA
```

ภาคผนวก ฎ.

ตัวอย่าง Processing Program Table (PPT)

```
PPT      TITLE 'DFHPPT CICS/VS PROCESSING PROGRAM TABLE '
        PRINT  NOGEN
         DFHPPT TYPE=INITIAL,                                          X
               SUFFIX=J1,                SUFFIX__     ************     X
               INDEX=YES                 IS ALPHA. ORDERED LIST
* ------------------------------------------------------------------ *
*     DEFINE GROUP OF APPLICATION PROGRAM THAT CONTAIN OF             *
*           1. NAME OF INTERFACE PROGRAM (ASSEMBLY LANGUAGE)          *
*                    == HOSTAPI ==                                    *
*           2. NAME OF APPLICATION PROGRAM FOR DEMO (ASSEMBLY)        *
*                    == HINQASM ==                                    *
*           3. NAME OF APPLICATION PROGRAM FOR DEMO (COBOL)           *
*                    == HINQCOB ==                                    *
* ------------------------------------------------------------------ *
         DFHPPT TYPE=ENTRY,PROGRAM=HOSTAPI
         DFHPPT TYPE=ENTRY,PROGRAM=HINQASM
         DFHPPT TYPE=ENTRY,PROGRAM=HINQCOB,PGMLANG=COBOL
*
* ------------------------------------------------------------------ *
*     DEFINE GROUP OF SYSTEM PROGRAM (CICS/VS)                        *
* ------------------------------------------------------------------ *
DFHTDWT$ DFHPPT TYPE=ENTRY,             OPTNL TD AUTO INIT TASK       X
               PROGRAM=DFHTDWT$         WRITE TO L3277 PRTR
INTERPRE DFHPPT TYPE=GROUP,                                          X
               FN=INTERPRETER           3270 PRINT ALLOCA. PROGRAM
HARDCOPY DFHPPT TYPE=GROUP,                                          X
               FN=HARDCOPY              3270 PRINT ALLOCA. PROGRAM
CONSOLE  DFHPPT TYPE=GROUP,                                          X
               FN=CONSOLE               WRITE TO CPU CONSOLE
SIGNON   DFHPPT TYPE=GROUP,                                          X
               FN=SIGNON                SIGNON TABLE
FE       DFHPPT TYPE=GROUP,                                          X
               FN=FE                    FE TERMINAL TEST PGM
DFHPEP   DFHPPT TYPE=ENTRY,                                          X
               PROGRAM=DFHPEP           USER PROGRAM ERROR PGM
DFHTEPT  DFHPPT TYPE=ENTRY,                                          X
               PROGRAM=DFHTEPT          TERM ERROR TABLE
OPENCLSE DFHPPT TYPE=GROUP,                                          X
               FN=OPENCLSE              DYNAMIC OPEN/CLOSE PGM
MASTTERM DFHPPT TYPE=GROUP,                                          X
               FN=MASTTERM              MASTER TERMINAL PGM
* REQUIRED ENTRIES FOR BMS PAGING AND MSG SWITCHING
MSWITCH  DFHPPT TYPE=GROUP,                                          X
               FN=MSWITCH               MSG SWITCH PGM
BMS      DFHPPT TYPE=GROUP,                                          X
               FN=BMS                   TERM PAGE CLEAN-UP
* REQUIRED SYSTEM ENTRIES FOR LOGGING/JOURNALLING
JOURNAL  DFHPPT TYPE=GROUP,                                          X
               FN=JOURNAL               JOURNAL CONTROL KICK-OFF
RECOVERY DFHPPT TYPE=GROUP,                                          X
               FN=RECOVERY              TRANS BACKOUT PGM
* ENTRY MAY BE REQUIRED FOR DFHUAKP USER KEY-POINT PGM
AKP      DFHPPT TYPE=GROUP,                                          X
               FN=AKP                   ACTIVITY KEYPOINT PGM
STANDARD DFHPPT TYPE=GROUP,                                          X
               FN=STANDARD              STANDARD ENTRIES
```

```
TIME     DFHPPT TYPE=GROUP,                                          X
               FN=TIME                TIME ADJUSTMENT PGM
DFHTLT1$ DFHPPT TYPE=ENTRY,                                          X
               PROGRAM=DFHTLT1$       TERM LIST TABLE
DFHXLT1$ DFHPPT TYPE=ENTRY,           OPTNL USED FOR SHUTDOWN       X
               PROGRAM=DFHXLT1$       FOR CICS ONLY
DFHPLT1$ DFHPPT TYPE=ENTRY,           OPTNL USED FOR SHUTDOWN       X
               PROGRAM=DFHPLT1$       FOR CICS ONLY
DFHSDR$  DFHPPT TYPE=ENTRY,           OPTNL. LOGS BTAM ERRS TO      X
               PROGRAM=DFHSDR$        DOS IJSYSRC FILE FOR CE
AUTOSTAT DFHPPT TYPE=GROUP,                                          X
               FN=AUTOSTAT            AUTO STATS SUMMARY PGM
EDF      DFHPPT TYPE=GROUP,                                          X
               FN=EDF
ATP      DFHPPT TYPE=GROUP,                                          *
               FN=ATP                 ASYNCH TRANS PROC GROUP
VTAMPRT  DFHPPT TYPE=GROUP,                                          *
               FN=VTAMPRT             VTAM PRINT FN.
RESPLOG  DFHPPT TYPE=GROUP,                                          *
               FN=RESPLOG             RESPONSE LOGGER
VTAM     DFHPPT TYPE=GROUP,                                          *
               FN=VTAM                VTAM GROUP
RESEND   DFHPPT TYPE=GROUP,                                          *
               FN=RESEND              REQUIRED FOR SOME VTAM SYSS
BACKOUT  DFHPPT TYPE=GROUP,                                          *
               FN=BACKOUT             DYNAMIC BACKOUT PROGRAM
ISC      DFHPPT TYPE=GROUP,                                          *
               FN=ISC                 INTER SYSTEMS COUPLING
OPERATOR DFHPPT TYPE=GROUP,                                          *
               FN=OPERATORS           OPERATOR        GROUP%%%%%%%%%
RTY      DFHPPT TYPE=ENTRY,          OPTNL 'RETRY' EXIT PROGRAM    X
               PROGRAM=DFHRTY
*  ************************
         DFHPPT TYPE=FINAL
         END    DFHPPTBA
```

ภาคผนวก ฎ.

ตัวอย่าง File Control Table (FCT)

```
FCT      TITLE 'CICS/VS FILE CONTROL TABLE  (FCT) '
         PRINT NOGEN
         DFHFCT TYPE=INITIAL,           REQUIRED OPERAND              *
               SUFFIX=J1                SUFFIX
*        ..........................................................
         DFHFCT TYPE=DATASET,                                         *
               DATASET=FILEIN,                                        *
               ACCMETH=(VSAM,KSDS),                                   *
               SERVREQ=(GET,BROWSE,PUT,UPDATE,NEWREC,DELETE),         *
               RECFORM=(FIXED,UNBLOCKED),                             *
               BUFND=2,                                               *
               BUFNI=1,                                               *
               OPEN=INITIAL,                                          *
               STRNO=2
*        ..........................................................
         DFHFCT TYPE=FINAL
         END   DFHFCTBA
```

ประวัติผู้เขียน

นายศิริพงษ์ โลหะศิริกุล เกิดวันที่ 27 กรกฎาคม 2503 สำเร็จการศึกษาพาณิชยศาสตร์บัณฑิต สาขาวิชาการจัดการเชิงปริมาณ จากคณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2525 เข้าศึกษาระดับปริญญามหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ ในปี พ.ศ. 2526

ปัจจุบันทำงานในตำแหน่งวิศวกรระบบ ส่วนประสานงานและวางระบบ ฝ่ายคอมพิวเตอร์ ธนาคารสหธนาคาร จำกัด