การควบคุมเชิงตัวเลขด้วยคอมพิวเตอร์

COMPUTERIZE NUMERICAL CONTROL

โดย

นาย จักรกฤษณ์	ลิมปีระกุล	34162145
น.ส. ชมัยพร	วงศ์เมือง	34162148
นาย วิสุทธิ์	จงประเสริฐศักดิ์	34162170
นาย เศกสรรค์	วจนะคุณากร	34162172

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

สาขาวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

การควบคุมเชิงตัวเลขด้วยคอมพิวเตอร์

COMPUTERIZE NUMERICAL CONTROL

นาย จักรกฤษณ์ ลิมปีระกุล 34162145

น.ส. ชมัยพร วงศ์เมือง 34162148

นาย วิสุทธิ์ จงประเสริฐศักดิ์ 34162170

นาย เศกสรรค์ วจนะคุณากร 34162172

อาจารย์ที่ปรึกษา

ภากร หุตะสังกาศ

Advisor

Phakorn Huthasankarn

ปริญญานิพนธ์สำหรับปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาคอมพิวเตอร์อุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขาวิชา เทคโนโลยีคอมพิวเตอร์อุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมเชิงเลขด้วยคอมพิวเตอร์

COMPUTERIZE NUMERICAL CONTROL

ผู้จัดทำ

1. นาย	จักรกฤษณ์	ลิมป์ปีระกุล	34162145
2. น.ส.	ชมัยพร	วงศ์เมือง	34162148
3. นาย	วิสุทธิ์	จงประเสริฐศักดิ์	34162170
4. นาย	เศกสรรค์	วจนะคุมากร	34162172

.....อาจารย์ที่ปรึกษา

(อาจารย์ ภากร หุตะสิงกาศ)

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการนำคอมพิวเตอร์ (386 SX-16) ใช้ในการควบคุมงานในทางอุตสาหกรรม โดยการทำงานของเครื่องจะใช้ระบบการควบคุมเชิงตัวเลขด้วยคอมพิวเตอร์ (COMPUTERIZE NUMERICAL CONTROL) ซึ่งหลักการข้างต้น นั้นจะมาทำการจำลองแบบและประยุกต์เป็นเครื่องพับเหล็ก ซึ่งจะใช้หลักการควบคุมระยะทางการพับของเหล็กแบบแกนเดียวโดยใช้ Motor AC ขับเฟืองทดกำลัง และใช้ Encode ในการอ่านค่าระยะทาง การทำงานในส่วนนี้จะมีทั้งทางด้านฮาร์ดแวร์และ ซอฟแวร์ ระบบซอฟแวร์ของเครื่องคำสั่งภายในทั้งหมดจะถูกเขียนขึ้นจากภาษาปาสคาล

Abstract

Main gist of this is availability computer (386 sx-16) in industry control work with computerize numerical control. The prinuple is duplication and adaptation of stul folded machine. Direction control of fold stul,single axis,can be done by selecting of ball screw. The process covered both hardware and software. All commands in software was created by pascal which as a high level language and popular in the present.

สารบัญ

บทนำ

ในปัจจุบัน ไมโครคอมพิวเตอร์ ได้เข้ามามีบทบาทมาก ในวงการธุรกิจ วิทยาศาสตร์ และอุตสาหกรรม เพราะเป็นส่วนที่ช่วยในการเพิ่มปริมาณงาน เพิ่มประสิทธิภาพของงาน มีความรวดเร็วกว่าการใช้แรงงานของมนุษย์ เช่น การผลิตแผ่นวงจรพิมพ์ การกัดเหล็ก ฯลฯ โดยนำเอาไมโครคอมพิวเตอร์มาใช้ในส่วนการควบคุมเครื่อง CNC (COMPUTERIZE NUMERICAL CONTROL) ทำให้งานที่ผลิตออกมามีความเที่ยงตรงสูง และจะช่วยในการประหยัดค่าใช้จ่ายเนื่องจาก ในขณะนี้เครื่องไมโครคอมพิวเตอร์มีราคาถูกลงกว่าสมัยก่อน

ปริญญานิพนธ์นี้จะเป็นการวิจัย และการออกแบบควบคุมเครื่องกัดเหล็ก ซึ่งจะใช้ CNC ควบคุมการทำงาน โดยจะทำทั้งในส่วน ฮาร์ดแวร์ และ ซอฟแวร์ การวิจัยส่วนนี้จะนำไปใช้เป็นเครื่องต้นแบบ และใช้งานจริง ทำให้ลดค่าใช้จ่ายในส่วนต้นทุนลงมาก จึงเหมาะสมกับการนำไปศึกษา และใช้ในงานจริงได้ซึ่งมีประสิทธิภาพเท่าเทียมกับเครื่องต้นแบบ

วัตถุประสงค์และขอบเขตของปริญญานิพนธ์

การวิจัยโครงงานนี้จะกำหนดเป้าหมายไว้ให้ทำการออกแบบและสร้างเครื่องต้นแบบที่ควบคุมการทำงานด้วยไมโครคอมพิวเตอร์ โดยการ INTERFACE วงจรควบคุมการเคลื่อนที่ วงขับเหล็กเข้ากับเครื่องไมโครคอมพิวเตอร์

ในการทำเครื่องขับเหล็กนี้จะใช้หลักการของ CNC แกนเดียว คือทางแกนนอน (Z-AXIS) ซึ่งในเทอมนี้จะมีการพัฒนาทั้งทางด้าน ฮาร์ดแวร์(A/D CONVERTER, DECODER) และซอฟแวร์ โดยนำโปรแกรมจากเทอมแรก มาทำการพัฒนาให้มีการใช้งานที่สมบูรณ์แบบขึ้นกว่าเดิม โดยการพัฒนาส่วนเครื่องขับเหล็กทั้งหมดนั้นจะมีด้วยกัน 3 ส่วนคือ

1. ส่วนควบคุม
2. ส่วนของ STEPPING MOTOR
3. ส่วนของการควบคุมระบบโดยใช้ซอฟแวร์

ซึ่งในที่นี้จะกล่าวเพียงในส่วนที่ 3 เท่านั้น การทำงานของระบบ จะใช้ภาษาระดับสูงในการเขียนโปรแกรมคือ ภาษาปาสคาล เพราะภาษานี้เป็นที่นิยมในปัจจุบัน

บทที่ 1

การใช้งาน 8255

1.1 ลักษณะทั่วไปของ 8255 PPI

8255 PPI(PROGRAMMABLE PERIPHERAL INTERFACE) เป็น LSI ขนาด 40 ขาทำหน้าที่ INTERFACE ระหว่าง MICROPROCESSOR กับอุปกรณ์ภายนอก 8255 ถูกออกแบบมาใช้กับไมโครโปรเซสเซอร์เบอร์ 8080

บล็อกไดอะแกรมของ 8255 แสดงได้ดังรูปที่ 1. ซึ่งมีส่วนที่ใช้ติดต่อกับอุปกรณ์ภายนอก 4 กลุ่ม คือ PA0-PA7,PB0-PB7,PC0-PC3 และ PC4-PC7 กลุ่มของสัญญาณควบคุมมี 2 กลุ่มคือ GROUP A CONTROL และ GROUP B CONTROL ซึ่งเป็นส่วนควบคุมการทำงานของทั้ง 3 พอร์ท DATA BUS BUFFER และ READ/WRITE CONTROL LOGIC ใช้สำหรับติดต่อกับไมโครโปรเซสเซอร์ทาง บัสข้อมูลและสัญญาณควบคุมการอ่านและเขียนข้อมูลกับรีจีสเตอร์ที่อยู่ภายใน 8255

1.2 สัญญาณต่างๆ ของ 8255

หน้าที่ของสัญญาณต่างๆของ 8255 เป็นดังนี้

D0-D7 : เป็นขาข้อมูลที่ใช้ต่อกับไมโครโปรเซสเซอร์

CS (CHIP SELECT INPUT) : เมื่อขานี้มีค่า ลอจิก 0 CPU สามารถติดต่อกับ 8255 ได้

RD (READ INPUT) : เมื่อขานี้มีค่า ลอจิก 0 พร้อมกับ CS 8255 จะส่งข้อมูลออกมาทางบัสข้อมูล

WR (WRITE INPUT) : เมื่อขานี้มีค่า ลอจิก 0 พร้อมกับ CS ข้อมูลที่อยู่บนบัสข้อมูลของระบบจะถูกเขียนลงไปใน 8255

A0-A1 (ADDRESS INPUT) : ใช้สำหรับชี้ตำแหน่งของ รีจิสเตอร์ภายใน 8255 ที่ CPU ต้องการติดต่อ

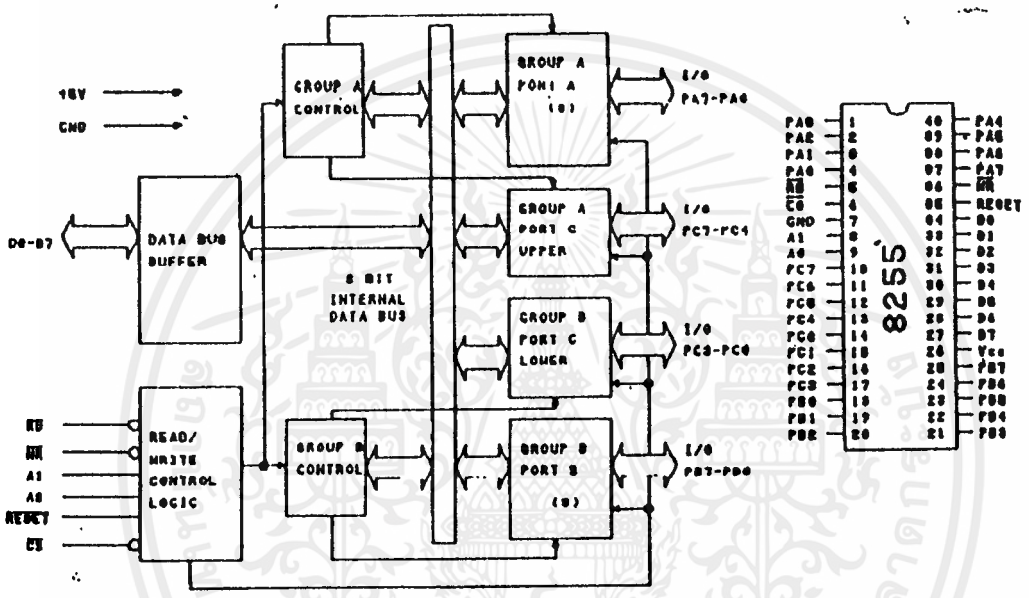
RESET : เมื่อขานี้มีค่า ลอจิก 1 8255 จะอยู่ในช่วง RESET พอร์ทุกพอร์ทจะอยู่

ในโหมดของ อินพุทพอร์ท

PA0-PA7 : เป็นพอร์ทข้อมูลที่ใช้สำหรับติดต่ออุปกรณ์ภายนอก

PB0-PB7 : เป็นพอร์ทข้อมูลที่ใช้สำหรับติดต่ออุปกรณ์ภายนอก

PC0-PC7 : เป็นพอร์ทข้อมูลที่ใช้สำหรับติดต่ออุปกรณ์ภายนอก



รูปที่ 1. แสดงบล็อกไดอะแกรมของ 8255

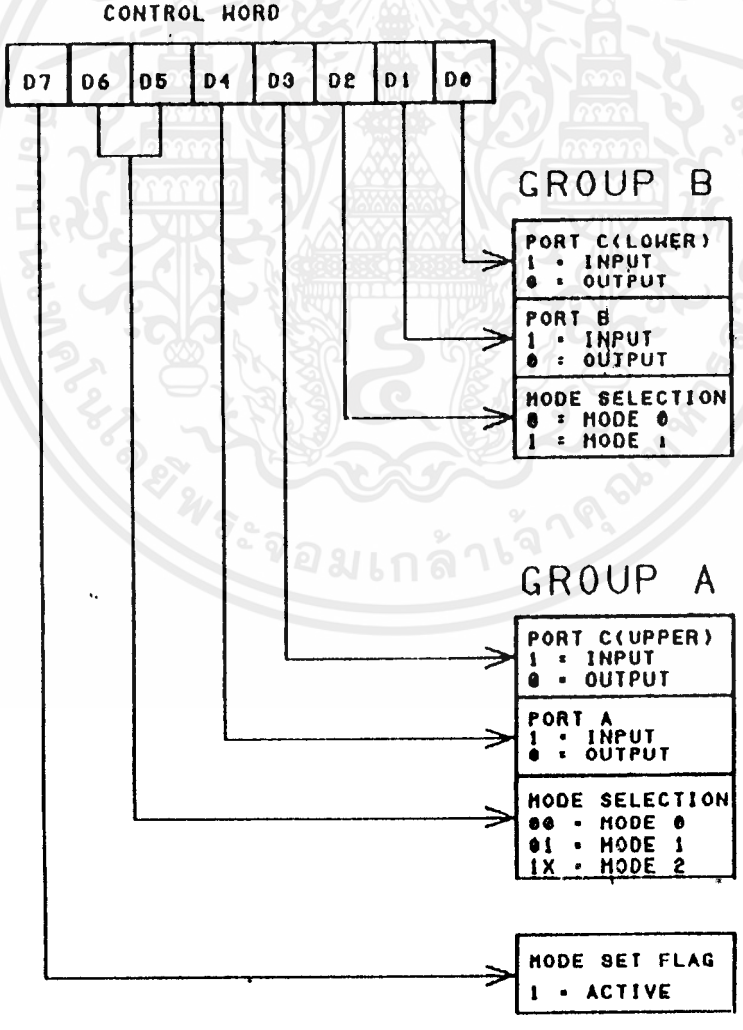
1.3 <u>การติดต่อกับพอร์ทต่างๆ ของ 8255</u>

ภายใน 8255 มีพอร์ทภายในอยู่ 4 พอร์ทซึ่งเราสามารถติดต่อกับพอร์ทต่างๆได้ดังนี้

DEVICE PINS				PORT NAME
RD	WR	A1	A0	

1	0	0	0	WRITE PORT A DATA
0	1	0	0	READ PORT A DATA
1	0	0	1	WRITE PORT B DATA
0	1	0	1	READ PORT B DATA
1	0	1	0	WRITE PORT C DATA
0	1	1	0	READ PORT C DATA
1	0	1	1	WRITE CONTROL WORD
0	1	1	1	ILLEGAL READ REGISTER

การทำงานของพอร์ท A,B,C จะกำหนดโดยข้อมูลที่ส่งไปยังพอร์ทควบคุมโดย แต่ละบิทจะมีความหมายดังในรูป 2. ซึ่งสามารถกำหนดการทำงานของ 8255 ได้ 3 โหมด



รูปที่ 2. แสดง CONTROL WORD ของ 8255

1.4 การใช้งาน 8255 ในโหมด 0

การทำงานของ 8255 ในโหมด 0 จะเป็นพอร์ทอินพุท หรือ พอร์ทเอาท์พุทธรรมดา เราสามารถกำหนดให้ 8255 ทำงานในโหมด 0 ได้โดยส่ง CONTROL WORD ไปยังพอร์ทควบคุม มีค่าต่อไปนี้

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0

จากค่าสั่งควบคุมในรูป 2. เราสามารถอธิบายความหมายของบิทต่างๆ ได้ดังนี้

D7 = 1 กำหนดให้ข้อมูลบนเป็น CONTROL WORD

D6,D5 = 0 กำหนดให้พอร์ท A ใน 8255 ทำงานในโหมด 0

D4 = 0 กำหนดให้พอร์ท A เป็นเอาท์พุท

D3 = 0 กำหนด 4 บิทบนของพอร์ท C เป็นเอาท์พุท

D2 = 0 กำหนดพอร์ท B ทำงานในโหมด 0

D1 = 0 กำหนดพอร์ท B เป็นเอาท์พุท

D0 = 0 กำหนด 4 บิทล่างของพอร์ท C เป็นเอาท์พุท

จาก CONTROL WORD ที่ส่งออกไปกำหนดให้พอร์ท A, B, C เป็นเอาท์พุทพอร์ททั้งหมด ซึ่งเราสามารถต่อกับอุปกรณ์ภายนอกทั้งหมดได้ 24 บิท สำหรับคำสั่งของ Turbo Pascal ที่จะกำหนดให้ 8255 ที่มีการต่อตามวงจรในรูปทำงานในโหมด 0 จะเป็นดังนี้

```
Add        := $1B3  ;Set Address
Port[Add] := $80    ;out control ($80) word to 8255
```

เมื่อ 8255 ถูกโปรแกรมแล้ว เราสามารถส่งข้อมูลไปที่พอร์ทตามที่ต้องการได้โดยใช้คำสั่ง Port[Add] เช่นเราต้องการส่งข้อมูล $23 ออกไปที่พอร์ท A ,$41 ออกไปที่พอร์ท B และ $73 ออกไปที่พอร์ท C เราสามารถทำได้โดยใช้โปรแกรมดังนี้

```
Port[$1B0]  :=   $12; output data to port A
Port[$1B1]  :=   $41; output data to port B
Port[$1B2]  :=   $73; output data to port C
```

หลังจากที่โปรแกรมถูก EXCUTUTE แล้วที่ A,B และ C จะมีค่าตามที่กำหนดการทำงานในโหมดนี้จะทำให้เราได้เอาท์พุทพอร์ท 3 พอร์ท ที่ประกอบอยู่ใน 8255

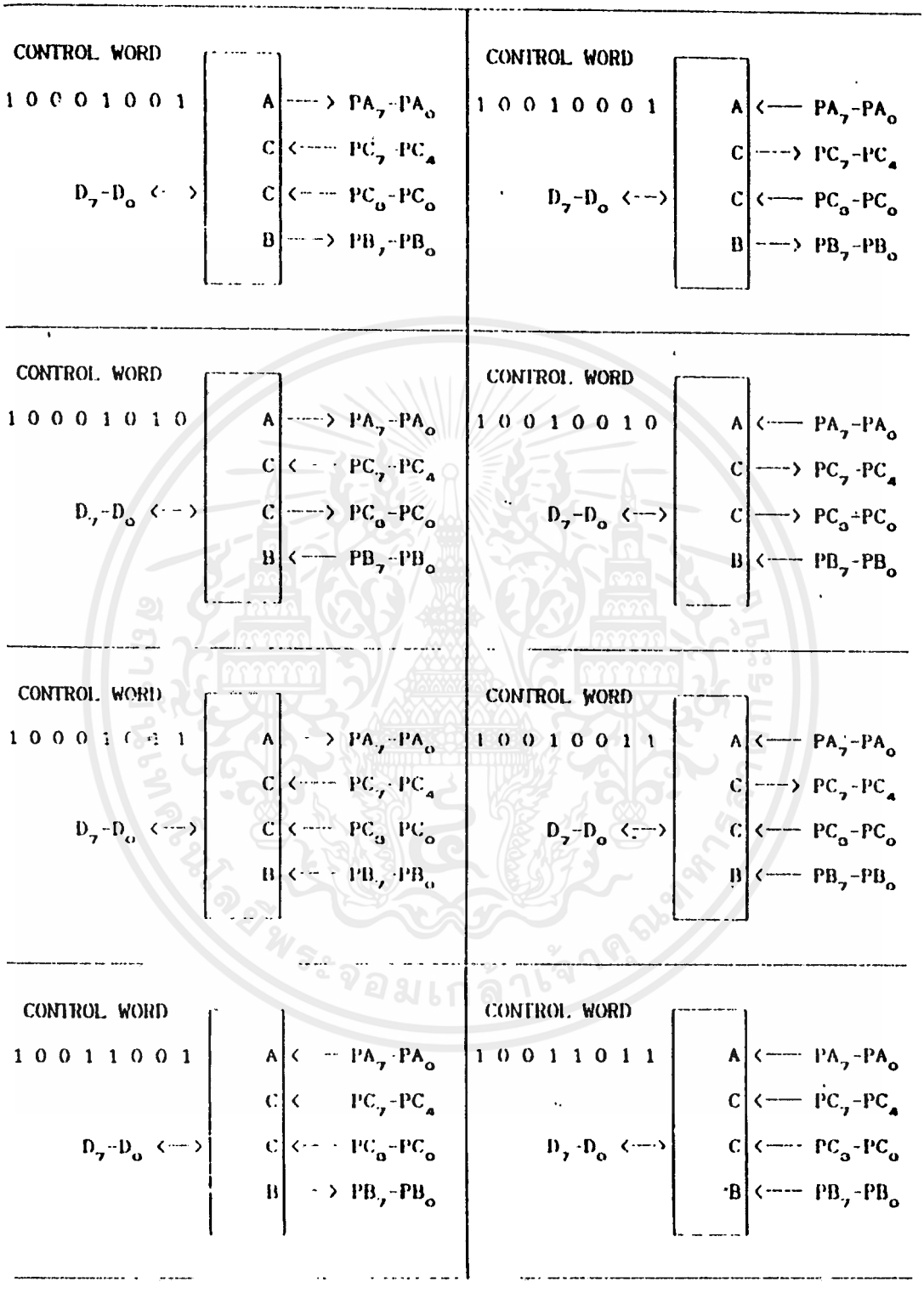
เราสามารถโปรแกรมให้ 8255 ทำงานในลักษณะของอินพุท และเอาท์พุทก็ได้ เช่นต้องการให้พอร์ท A และ C เป็นเอาท์พุทพอร์ท และ พอร์ท B เป็นอินพุทพอร์ทเราสามารถเขียน CONTROL WORD ได้ดังนี้

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	1	0

เมื่อ CONTROL WORD ถูกส่งไปยัง 8255 การทำงานของ 8255 จะเป็นตามที่เราต้องการซึ่งเราสามารถอ่านข้อมูลเข้ามายังพอร์ท B ได้โดยใช้คำสั่ง IN ดังนี้

```
datain  :=  port[$1b1];    ;  Read data from port B
```

เราสามารถกำหนดการทำงานของ 8255 ในโหมด 0 ได้หลายแบบแตกต่างกันดังแสดงในรูปที่ 3.



รูปที่ 3. การทำงานของ 8255 แบบต่างๆ ในโหมด 0

CONTROL WORD

1 0 0 0 0 0 0 0

D_7-D_0 <—>

A ----> PA_7-PA_0

C ----> PC_7-PC_4

C ----> PC_3-PC_0

B ----> PB_7-PB_0

CONTROL WORD

1 0 0 0 0 0 1 0

D_7-D_0 <—>

A ----> PA_7-PA_0

C ----> PC_7-PC_4

C ----> PC_3-PC_0

B <---- PB_7-PB_0

CONTROL WORD

1 0 0 0 0 0 0 1

D_7-D_0 <—>

A ----> PA_7-PA_0

C ----> PC_7-PC_4

C <---- PC_3-PC_0

B ----> PB_7-PB_0

CONTROL WORD

1 0 0 0 0 0 1 1

D_7-D_0 <—>

A ----> PA_7-PA_0

C ----> PC_7-PC_4

C <---- PC_3-PC_0

B <---- PB_7-PB_0

CONTROL WORD

1 0 0 0 1 0 0 0

D_7-D_0 <—>

A ----> PA_7-PA_0

C <---- PC_7-PC_4

C ----> PC_3-PC_0

B ----> PB_7-PB_0

CONTROL WORD

1 0 0 1 0 0 0 0

D_7-D_0 <—>

A <---- PA_7-PA_0

C ----> PC_7-PC_4

C ----> PC_3-PC_0

B ----> PB_7-PB_0

CONTROL WORD

1 0 0 1 1 0 0 0

D_7-D_0 <—>

A <---- PA_7-PA_0

C <---- PC_7-PC_4

C ----> PC_3-PC_0

B ----> PB_7-PB_0

CONTROL WORD

1 0 0 1 1 0 1 0

D_7-D_0 <—>

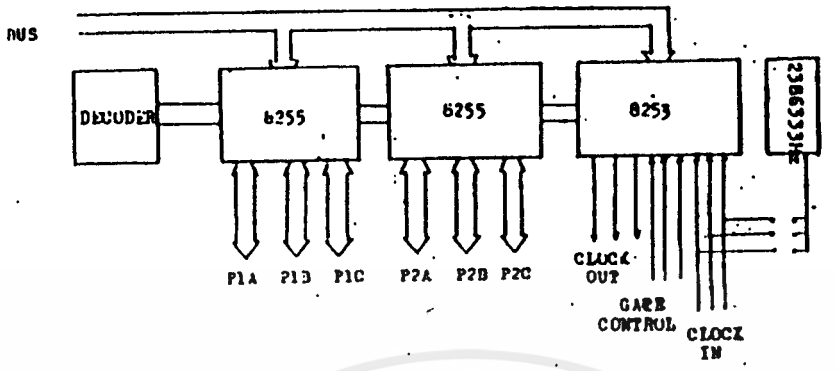
A <---- PA_7-PA_0

C <---- PC_7-PC_4

C ----> PC_3-PC_0

B <---- PB_7-PB_0

1.5 ส่วนของการ์ด 8255



รูปที่ 4. BLOCK DIAGRAM

SPECIFICATION

- 48 PROGRAMMABLE I/O CONTROL LINE
- 3 INDEPENDENT 16 BIT COUNTER
- 16 LED I/O DISPLAY

1.6 รายละเอียดของการ์ด 8255

1. PORT ADDRESS

   SW4 = OFF SW5 = ON Select$1B0 - $1BF

   SW4 = ON SW5 = OFF Select$1F0 - $1FF

2. SYSTEM BOARD DEFAULT $1B0 - $1BF

   $1B0 : port iA read/weite buffer

   $iB1 : port 1B read/weite buffer

$1B2 : port 1C read/weite buffer

$1B3 : port 1 control register(8255)*

$1B4 : port 2A read/write buffer

$1B5 : port 2B read/write buffer

$1B6 : port 2C read/write buffer

$1B7 : port 2 control register(8255)

$1B8 : counter 0 read/write buffer

$1B9 : counter 1 read/write buffer

$1BA : counter 2 read/write buffer

$1BB : counter chaip 8253 control register

3. SW1 =ON Select internal clock(1.19MHz)to counter 0

SW1 =OFF Counter 2 used external clock

SW2 =ON Select internal clock(1.19MHz) to counter 1

SW2 =OFF Counter 1 used external clock

SW3 =ON Select internal clock(2386333Hz) to counter 0

SW3 =OFF Counter 0 used external clock

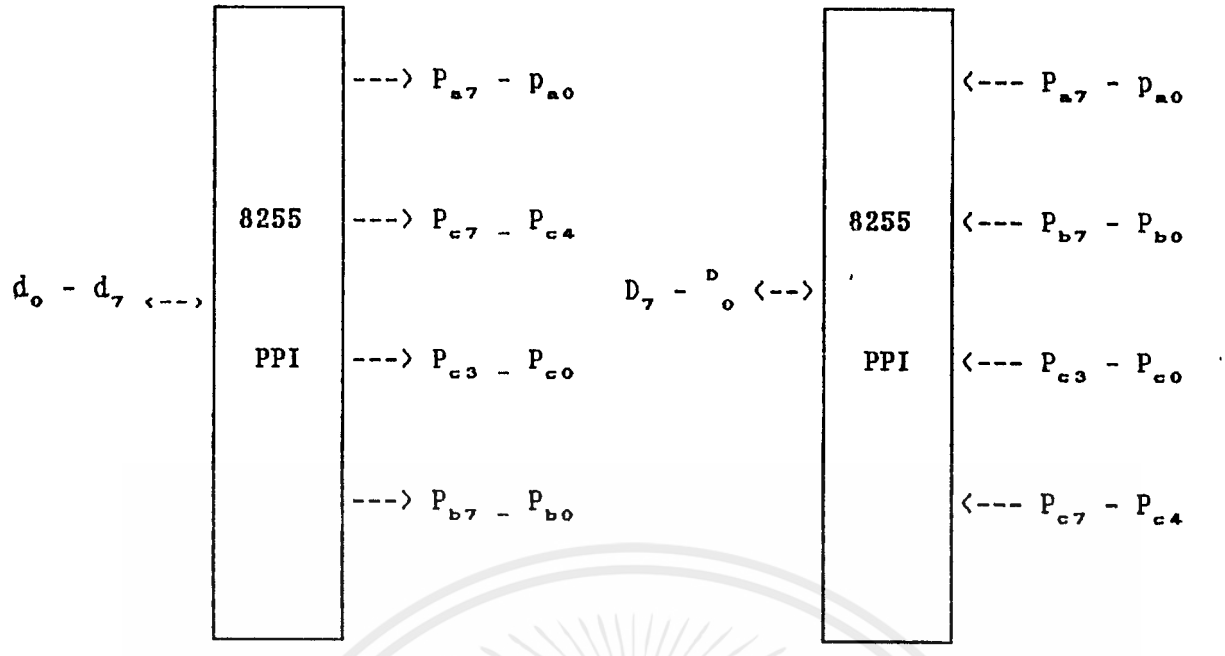
PORT รายละเอียดที่ใช้ 8255 ควบคุม D/A

$1B7 : PORT 2 CONTROL REGISTOR

$1B4 : PORT 2A READ/WRITE BUFFER

-$1B7 : ใช้เป็น CONTROL WORD ให้ค่าเท่ากับ $80 (1000 0000)

ให้ทุก port เป็น output



รูปที่ 5. การทำงานของ 8255 เมื่อให้ control word

-$1B4 : คือ PORT OUTPUT ($PA_7 - PA_0$) ที่นำไปควบคุม วงจร D/A

เช่นถ้า ส่งค่า $FF OUT ออกที่ PORT $1B4 ค่าที่ออกจากเอาท์พุทของ D/A = 5V

Port รายละเอียดที่ใช้ 8255 ควบคุมการอ่านค่าตำแหน่งจากวงจร counter

$1B3 : Port 1 counter register

$1b0 : Port 1a input 8 bit from Counter

$1b1 : Port 1b input 8 bit from Counter

$1b2 : Port 1c input 4 bit from Counter

- $1B3 : ใช้เป็น control word ให้ค่าเท่ากับ $93 (1001 0011)
ให้ port A , B และ C low เป็น input และ port C
high เป็น output

1.7 <u>การใช้การ์ด 8255 ในการอ่านค่าตำแหน่งจากวงจร Counter</u>

ในการอ่านค่าจากวงจร Counter จะใช้การ์ด 8255 Port 1A ที่ Address $1B0 Port 1B $1B1 Port 1C $1B2 และ Control Word ที่ตำแหน่ง1B3 ซึ่งในการกำหนดการทำงานของ 8255 จะส่ง Control Word $93 เพื่อเป็นการกำหนดให้ 8255 ทั้ง 3 Port (A,B,C_L) ทำงานเป็น Input และ C_H เป็น Output และสำหรับโปรแกรมจะนำเอา 8 bit ที่ 1A รวมกับ 8 bit ที่ 1B และ 4 bit ที่ $1C_L$ เป็น 20 bit แล้วแปลงเป็นฐาน 10 ดังโปรแกรม

```
Function OccurStep : longint;
Var
    P1,P2,P3 : Longint;
Begin
    P1 := (Port[Pread1]);
    P2 := (Port[Pread2]);
    P3 := (Port[Pread3]);
    OccurStep := (P3*256*256)+(P2*256)+P1;
End;
```

ทุกครั้งของการเคลื่อนที่ฉากถัดไป จะต้องมีการ Reset Counter เพื่อให้เริ่มนับใหม่ โดยจะใช้ Port $1C_H$ bit ที่ 7 เป็นตัว Reset ซึ่งจะเขียนโปรแกรมดังนี้

```
Port [Pread 3] := $8.;  (1000 0000)
Delay (10)
Port [Pread 3] := $00;
Delay (10)
```

บทที่ 2 .

การเชื่อมต่อกับ INVERTER

ในส่วนของการควบคุมความเร็วของ Motor จะนำเอา Inverter เข้ามาช่วยในการควบคุม เพราะชุดที่ใช้เลื่อนฉากนั้นมีมอเตอร์ 3 เฟส ติดตั้งมาอยู่กับเพลาขับเฟือง ถ้าใช้การควบคุมธรรมดาจะเกิดค่าผิดพลาดได้มาก จึงใช้ Inverter มาควบคุมความเร็วของ Motor ซึ่งสามารถลดข้อผิดพลาดลงได้มาก

การควบคุมผ่าน Inverter มีด้วยกัน 3 วิธี คือ

1. จากการปรับ VR ที่มากับเครื่อง Inverter
2. จากการปรับแรงดันด้วยวงจร D/A 0-5V.
3. จากการปรับกระแส 4-20 mA ไม่เกิน 24V.

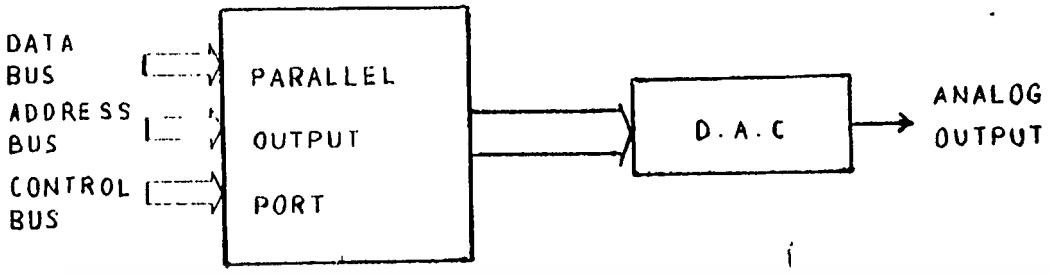
จากการควบคุมทั้ง 3 ระบบเมื่อใช้คอมพิวเตอร์เข้ามาควบคุม วิธีที่ 2 เป็นวิธีที่ง่ายที่สุด

2.1 การเปลี่ยนสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

สัญญาณในระบบไฟฟ้ามีสัญญาณพื้นฐานอยู่ 2 แบบ คือ สัญญาณดิจิตอล และสัญญาณอนาลอก สัญญาณอนาลอกหมายถึง สัญญาณที่มีค่าการเปลี่ยนแปลงต่อเนื่อง แต่สัญญาณดิจิตอลเป็นสัญญาณที่มีระดับการเปลี่ยนแปลงของสัญญาณ 2 ระดับเท่านั้นซึ่งสัญญาณสองระดับนี้เรากำหนดให้เป็น 0 และ 1 ในลักษณะของสัญญาณไบนารี่ซึ่งมีการใช้งานในระบบของไมโครโปรเซสเซอร์ ดังนั้นหากเราต้องการนำไมโครโปรเซสเซอร์ไปต่อกับอุปกรณ์ที่ใช้สัญญาณอนาลอกเราจำเป็นต้องมีการเปลี่ยนสัญญาณจากสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (DIGITAL TO ANALOG CONVERSION DAC) และเปลี่ยนสัญญาณจากอนาลอกเป็นสัญญาณดิจิตอล (ANALOG TO DIGITAL CONVERSION ADC) เพื่อให้ไมโครโปรเซสเซอร์สามารถต่อกับอุปกรณ์ที่รับ และส่งข้อมูลแบบอนาลอกได้

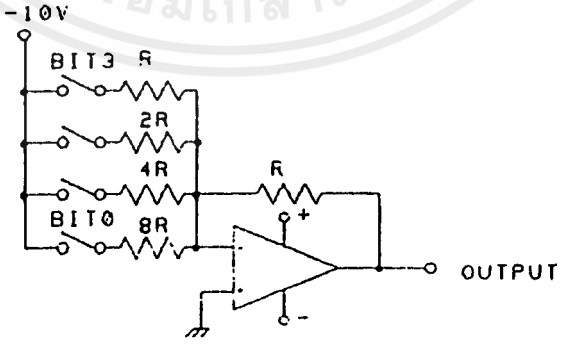
การส่งข้อมูลของสัญญาณดิจิตอลจากไมโครโปรเซสเซอร์ไปให้อุปกรณ์ที่ทำหน้าที่เปลี่ยน

สัญญาณดิจิตอลนั้น เราใช้การส่งข้อมูลออกไปที่พอร์ทเอาท์พุทแบบขนาน ดังในรูปที่ 6



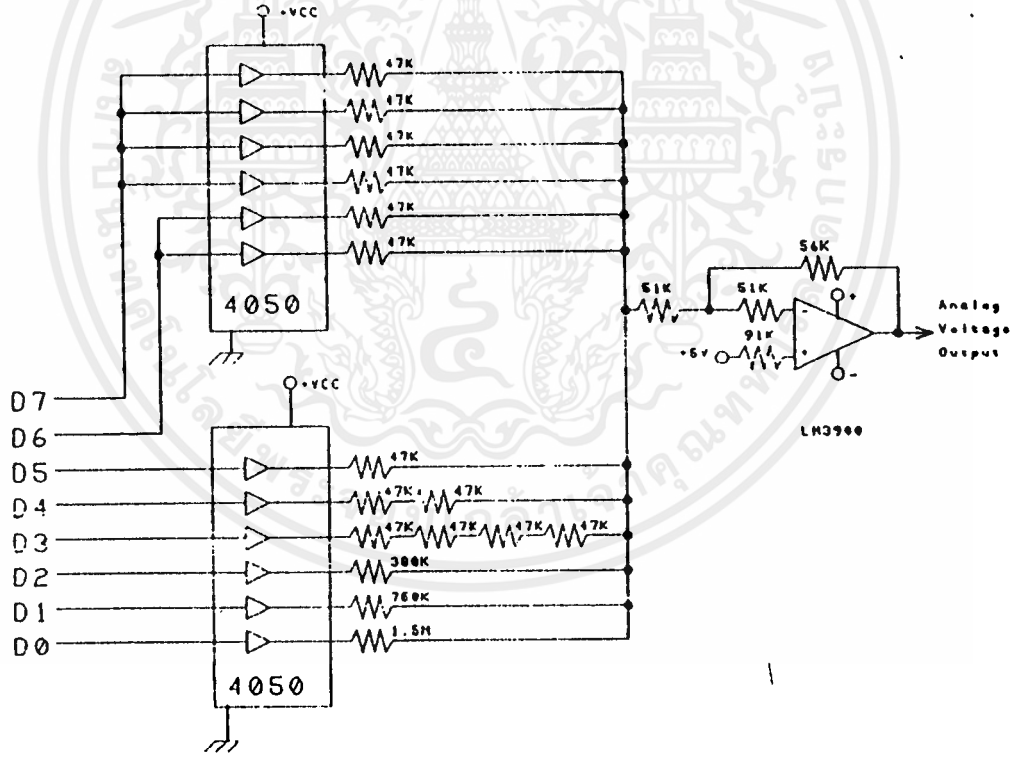
รูปที่ 6 ไดอะแกรมของการต่อซีพียูกับ DAC

หลักการเบื้องต้นของการเปลี่ยนสัญญารดิจิตอลเป็นสัญญาณอนาลอก นั้นเราอาศัยหลักการง่าย ๆ คือ ระดับของสัญญาณอนาลอกที่เกิดขึ้นจะต้องมีค่าเท่ากับตำแหน่งของเลขฐานสองที่คิดตามความสำคัญของบิท เช่น บิท 0 จะมีค่าของแรงดันเท่ากับ $2^0 = 1*V$ บิทที่ 1 จะมีค่าระดับแรงดันเท่ากับ $2^1 = 2*V$ บิทที่ 2 มีค่าระดับแรงดันเท่ากับ $2^2 = 4*V$ และบิทที่ N จะมีค่าระดับแรงดันเท่ากับ $2^n * V$ ตัวอย่างวงจรการ DAC แบบง่ายๆ จะเป็นดังรูปที่ 7



รูปที่ 7 แสดงวงจร DAC เบื้องต้น

จากรูป 7 วงจรประกอบด้วย สวิตช์ 4 ตัว และความต้านทาน 4 ตัวต่อเป็นอินพุทของวงจรรวมสัญญาณ โดยมีความต้านทาน 1 ตัวต่อเป็นวงจรป้อนกลับ ค่าของความต้านทานที่ต่ออยู่กับตำแหน่งของบิทต่างๆ จะมีค่าเป็น 1,2,4,8 ตามลำดับซึ่งจะทำให้เกณฑ์การขยายมีค่าเป็น -1/8,-1/4,-1/2 และ -1 ตามลำดับ เราสามารถทดสอบการทำงานของวงจรได้โดยการทำให้สวิทช์ทุกตัวเปิดหมด จะได้เอาท์พุทออกมามีค่าเท่ากับ 0 โวล์ท หากเราเปิดสวิทช์ 0 ก็จะได้แรงดันเอาท์พุทออกมาเท่ากับ 1.25 โวล์ท เมื่อเราเปิดสวิทช์ 1 ก็จะได้แรงดันออกมาเท่ากับ 2.5 โวล์ท หากเราเปิดสวิทช์ 1 และ 0 พร้อมกันก็จะทำให้ได้แรงดันออกมาเท่ากับ 1.25 + 2.5 = 3.75 โวล์ท



รูปที่ 8 แสดงวงจรสำหรับเปลี่ยนสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

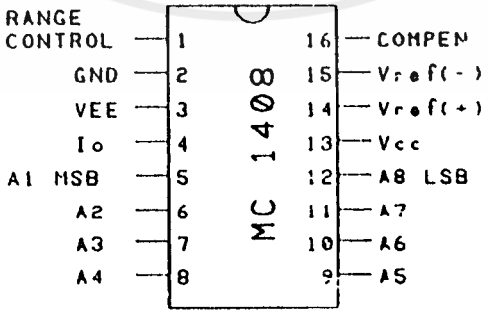
สำหรับวงจรการใช้งานสำหรับการเปลี่ยนสัญญาณดิจิตอลขนาด 8 บิท เป็นสัญ

ญาณอนาลอกแสดงได้ดังรูป 8 วงจรนี้มีความต้านทานค่าต่างๆ ต่อกับพอร์ทเอาท์พุทผ่านบัฟเฟอร์โดยค่า ความต้านทานที่บิท 7 มีค่าเท่ากับ 11.75 กิโลโอห์ม ซึ่งได้จากการต่อค่าความต้านทาน 47 กิโลโอห์ม ขนานกัน 4 ตัว ที่บิท 6 มีความต้านทาน 47 กิโลโอห์ม ขนานกัน 2 ตัว ซึ่งจะเป็นความแตกต่างของแต่ละบิทจากวงจรจะได้แรงดันเอาท์พุทมีค่าดังนี้

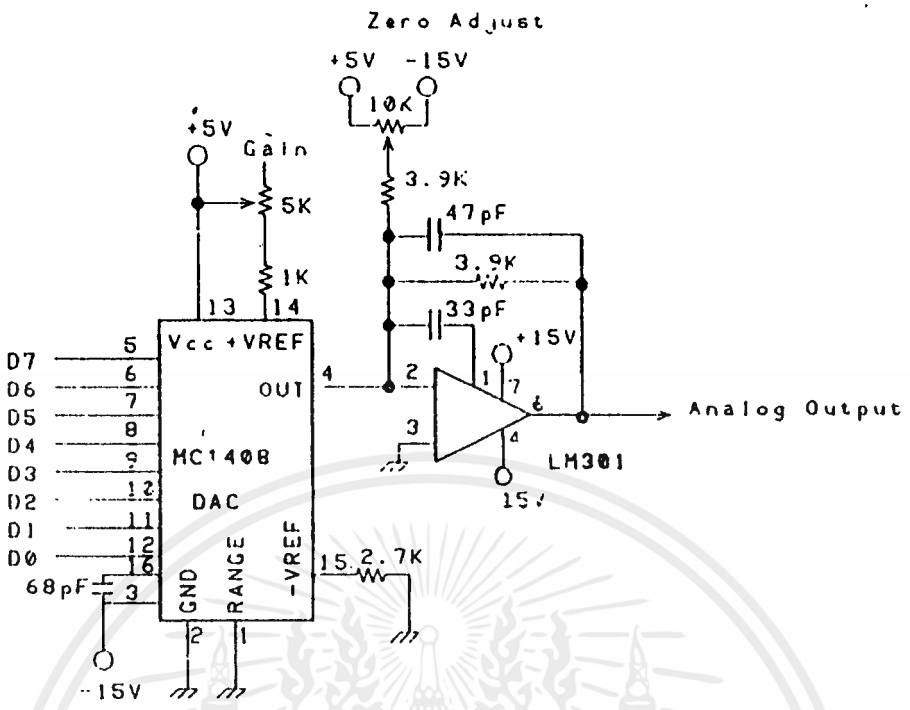
$$V_o = 5\ (n/255) \quad \text{โวลท์}$$

n : เป็นค่าของเลขฐานสิบที่ได้จากเลขฐานสองแปดบิท มีค่าตั้งแต่ 0-255 ดังนั้นค่าของข้อมูล 10001011 (139_{10}) เมื่อส่งไปให้วงจร DAC จะทำให้ได้เอาท์พุทเท่ากับ 2.7 โวลท์ ในลักษณะนี้แรงดันเอาท์พุทจะถูกแบ่งออกเป็น 255 ระดับ

การใช้งานจริงนั้นวงจรสำหรับเปลี่ยนสัญญาณดิจิตอลเป็นสัญญาณอนาลอกจะถูกรวมอยู่ในอุปกรณ์ตัวเดียว ในการทำงานนี้จะใช้IC เบอร์ MC 1408 ซึ่งเป็นผลิตภัณฑ์ของ MOTOROLAR เปลี่ยนสัญญาณดิจิตอลขนาด 8 บิท เป็นสัญญาณอนาลอก โดยขาสัญญาณต่างๆ ของ MC 1408 นี้แสดงได้ดังรูป และวงจรการใช้งานซึ่งเราสามารถปรับค่าของแรงดันอ้างอิงให้เปลี่ยนแปลงไปได้ MC 1408 จะให้เอาท์พุทอยู่ในช่วงประมาณ 0-2 mA โดยจะมีออปแอปม์ เบอร์ LM 741 ทำหน้าที่เปลี่ยนกระแสเป็นแรงดัน (CURRENT TO VOLTAGE CONVERTOR) เพื่อให้ได้แรงดันเอาท์พุทอยู่ในช่วง 0-5 โวลท์



รูปที่ 9 สัญญาณต่างๆ ของ MC 1408 DAC



รูปที่ 10 แสดงวงจรใช้งานของ MC 1408

2.2 <u>การเชื่อมต่อกับ Inverter</u>

ในการใช้โปรแกรมสั่งวงจร D/A ผ่านทางการ์ด 8255 จะใช้ Port 2A อยู่ที่ตำแหน่ง $1B4 เป็น Data Bus 8 bit ที่วงจร D/A ต้องการ และมี Control Word เป็น $80 อยู่ที่ตำแหน่ง $1B7 คือให้เป็น Output หมดทุก Port

ให้ Port 2A = $19 D/A = 0.5V.

= $35 D/A = 1 V.

= $65 D/A = 2 V.

= $FF D/A = 5 V.

ในการทดลองเมื่อใช้งานวงจร D/A ติดต่อกับ Inverter นานเข้าเกิดความผิดพลาดขึ้น ทำให้การเชื่อมต่อระหว่าง D/A กับ Inverter ใช้การไม่ได้ เนื่องจาก Inverter มี Pulse ออกมากวนตลอด แต่ตัว Inverter ยังใช้งานได้ จึงใช้ Port Output

ที่เป็น Relay มาแทน โดยต่อค่าความต้านทานปรับค่าได้ไว้ 4 ตำแหน่งโดย fixed ค่าไว้ที่ 0.5V. ,1V. ,2V. ,5V. โดยใช้ Port Output 2 ที่ตำแหน่ง $0283 4 bit บนคือ bit 4 - bit 7 โดยที่ bit 4 ตั้งค่าไว้ 0.5V. bit 5 ตั้งค่าไว้ 1V. bit 6 ตั้งค่าไว้ 2V. และ bit 7 ตั้งค่าไว้ 5V.

Port Output 2 ($0283)

bit 0 - bit 4		ไม่ได้ใช้
bit 4	Inverter	0.5V.
bit 5	Inveter	1V.
bit 6	Inverter	2V.
bit 7	Inverter	5V.

โปรแกรมในการควบคุม Inverter ที่ 5V.

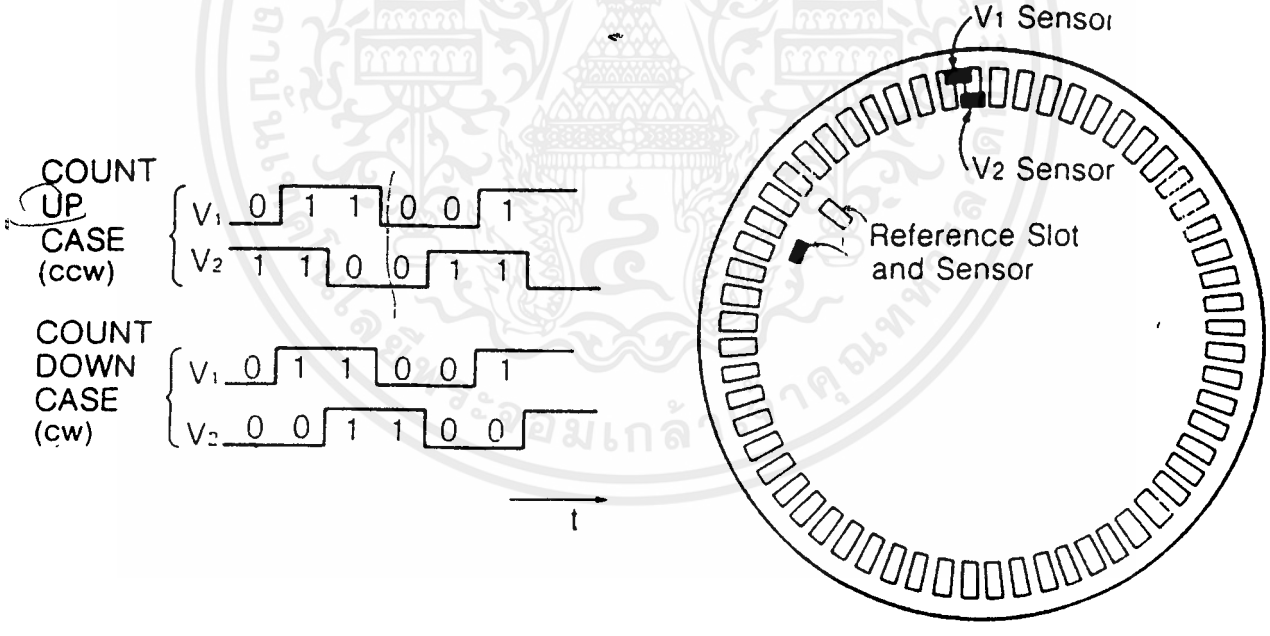
```
Port [$0283] := $80;
```

บทที่ 3

Incremental Encoder

ใช้ Incremental Encorder เป็นตัวบอกตำแหน่งของฉากกั้นหลัง Encorder จะให้ Pulse 256 pulse ต่อการหมุนไปพร้อมกับเฟืองฉากกั้นหลัง จะทำให้ 1 step มีความละเอียด = 0.00390625 cm.

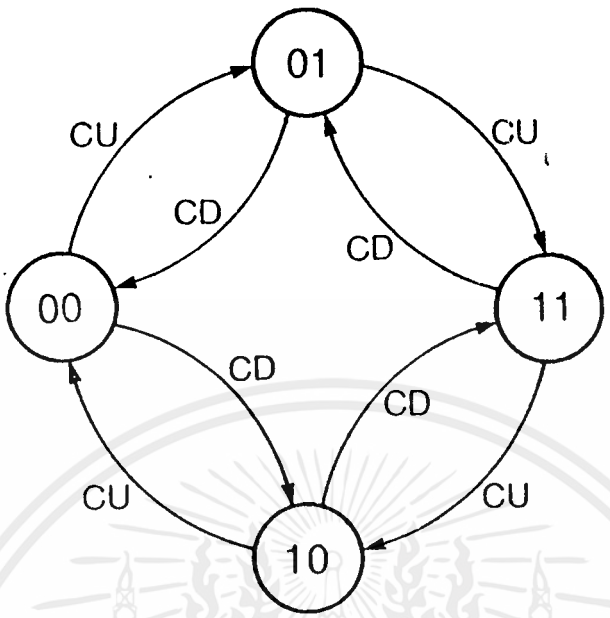
การใช้งานของ Incremental Encorder จะต่อวงจร Counter มานับ Pulse ที่เกิดขึ้นเพื่อนำไปคำนวณระยะทาง ซึ่งภายในของ Incremental Encorder แสดงดังรูป



รูปที่ 11 output of incremental encoder

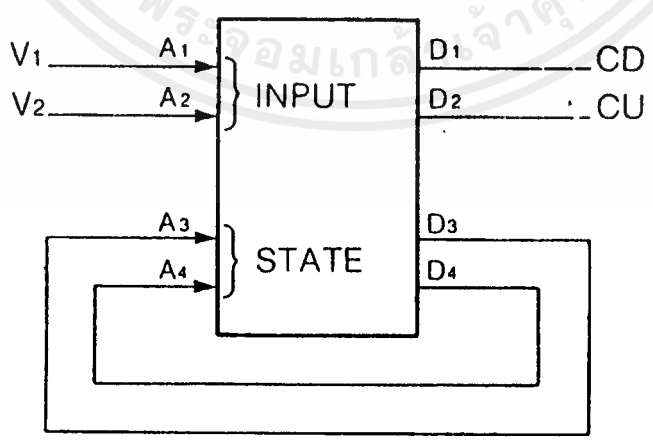
Incremental Encorder สามารถบอกทิศทางได้ด้วย ซึ่งจากการวางตำแหน่งของตัว Sensor ที่ตำแหน่ง 90 องศา มุมทางไฟฟ้าก็ต่างกัน 90 องศาด้วยเช่นกัน

ซึ่งจาก Clock Diagram รูปข้างบนนามาเขียนเป็น State Diagram ได้ดังนี้



รูปที่ 12 state diagram

และจาก State Diagram ทำให้ได้ข้อมูลใน Eprom ตาม State Digram ดังนี้

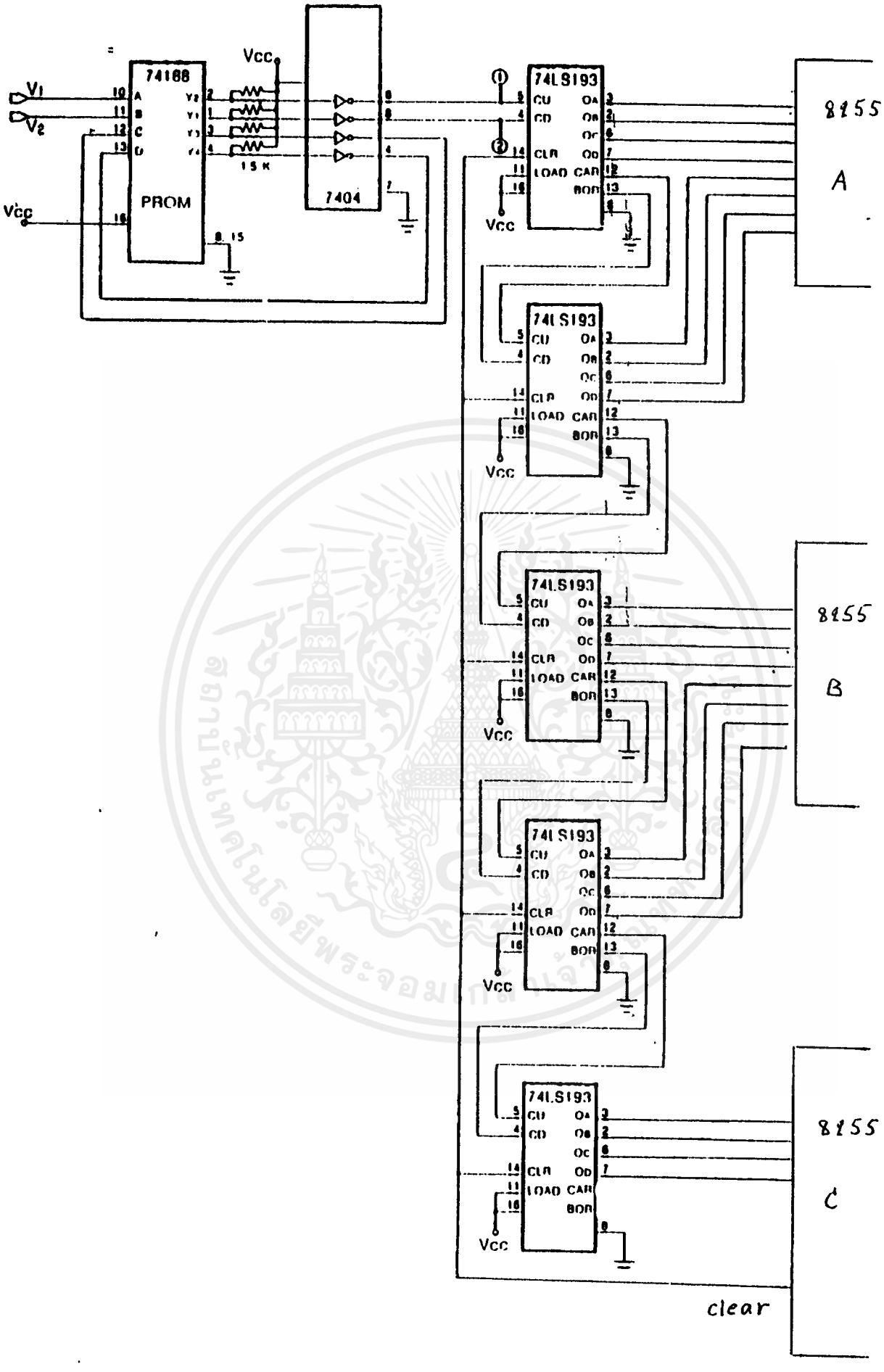


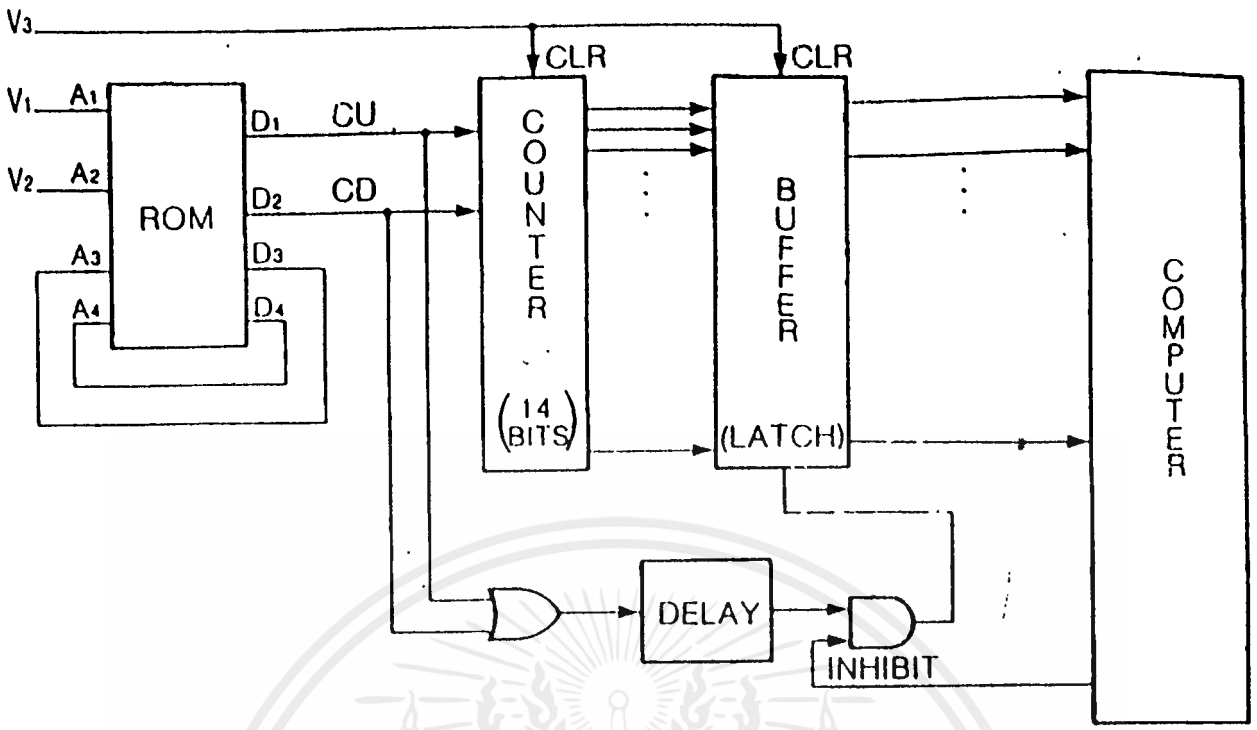
รูปที่ 13 organization of ROM

	A1	A2	A3	A4	D1	D2	D3	D4
	0	0	0	0	0	0	0	0
	0	0	0	1	1	0	0	0
	0	0	1	0	0	1	0	0
Impossible	0	0	1	1	0	0	1	1
	0	1	0	0	0	1	0	1
	0	1	0	1	0	0	0	1
Impossible	0	1	1	0	0	0	1	0
	0	1	1	1	1	0	0	1
	1	0	0	0	1	0	1	0
Impossible	1	0	0	1	0	0	1	0
	1	0	1	0	0	0	1	0
Stable 10 state	1	0	1	1	0	1	1	0
Impossible	1	1	0	0	0	0	0	0
	1	1	0	1	0	1	1	1
Not stable	1	1	1	0	1	0	1	1
Stable 11 state	1	1	1	1	0	0	1	1

ตารางข้อมูลใน EPROM

และจากที่กล่าวมานำมาเขียน Diagram ของวงจรได้ดังนี้





รูปที่ 14 block diagram of position hardware

การกำหนดตำแหน่งสำหรับ OPTICAL ENCODER

สัญญาณที่ออกมาจาก OPTICAL ENCODER แบบไม่สมบูรณ์ในตัวเอง มีทั้งหมด 3 สัญญาณคือ V1,V2 และ V3 ซึ่งจะมีข้อมูลของทิศทาง ตำแหน่ง และความเร็วแฝงอยู่ในสัญญาณเหล่านี้ ซึ่งจะต้องแยกข้อมูลเหล่านั้นออกมา โดยเริ่มจากการแยกเอาข้อมูลของทิศทางและตำแหน่งออกมาก่อน จากรูป 11 จะกำหนดให้ทิศทางของการหมุนตามเข็มนาฬิกาและ ทวนเข็มนาฬิกา เป็นการนับลงและนับขึ้นของวงจรนับ ตามลำดับ และจะให้สัญญาณ V1 และ V2 แทนบัสแรก และ บัสที่ 2 ตามลำดับ ซึ่งจะได้สภาวะทั้งหมด 4 สภาวะคือ 00,01,11 และ 10 ความสัมพันธ์ระหว่างสภาวะต่างๆ ในการนับขึ้นและลงจะเป็นไปตามไดอะแกรมแสดงสภาวะดังในรูปที่ 12 การออกแบบวงจรเพื่อให้ทำงานตามสภาวะได้ตามรูปที่ 13 มีด้วยกันหบายวิธี แต่วิธีที่นิยมก็คือ การใช้หน่วยความจำแบบ ROM ขนาดแอดแดรส และข้อมูลอย่างละ 4 บัสด้วยกันดังในรูปที่ 13 จะเห็นว่า ADDRESS BUS A1 และ A2 จะรับสัญญาณจาก V1 และ V2 ตามลำดับ และบัสข้อมูล D1 และ D2

จะเป็นสัญญาณเอาท์พุทที่จะนำไปผ่านวงจรฉับลง และฉับขึ้นตามลำดับ ส่วนบัสข้อมูล D3 และ D4 จะถูกป้อนกลับมายังบัสแอดแดรส A3 และ A4 ตามลำดับดังนั้น A3 = D3 และ D4 = A4 เสมอที่สภาวะเสถียรใดๆ

ข้อมูลภายใน ROM จะถูกป้อนเข้าไปเก็บไว้ทั้ง 16 กลุ่ม โดยมี ADDRESS ตั้งแต่ 0000-1111 ตามตาราง เพื่อให้เข้าใจการทำงานของ ROM ที่มีลักษณะการต่อดังรูปที่ 13 จะยกตัวอย่างของกรณีการหมุนทวนเข็มนาฬิกา หรือการนับลงโดยจะเปลี่ยนแปลงสภาวะที่อินพุทของ ROM จาก 00-10-11-01 โดยเริ่มจากสภาวะแรกที่ 00 ถ้าดูจากตารางจะเห็นว่าทุกบัส ขณะนี้เป็น 0 หมด ต่อมาเมื่อสภาวะจะเปลี่ยนไปเป็น 1000 ซึ่งถ้าดูจากตารางจะเห็นว่าที่ตำแหน่ง ADDRESS นี้ บัสข้อมูลจะให้ข้อมูล 1010 ออกมาแต่เนื่องจาก A3 ต้องเท่ากับ D3 และ A4 ต้องเท่ากับ D4 จึงทำให้ D4 และ A4 เปลี่ยนจาก 00 เป็น 10 ด้วย ซึ่งจะทำให้บัส ADDRESS เปลี่ยนไปเป็น 1010 และเมื่อดูจากตารางที่ ADDRESS นี้จะให้ข้อมูลแสดงค่า 0010 ออกมา และจุดนี้เองเป็นสภาวะเสถียรตำแหน่ง 10 ถ้าสภาวะต่อไปจะเปลี่ยนจาก 10 ไปเป็น 11 อีกที่ บัส ADDRESS จะเปลี่ยนไปเป็น 1110 ทำให้บัสข้อมูลเปลี่ยนไปเป็น 1011 และผลนี้เองทำให้ บัส ADDRESS ต้องเปลี่ยนไปเป็น 1111 เพื่อทำให้ A3 = D3 และ A4 = D4 เข้าสู่สภาวะเสถียรตำแหน่ง 11

การทำงานของวงจร

การเปลี่ยนสภาวะในครั้งหนึ่ง ๆ ก่อนที่ระบบจะเข้าสู่สภาวะเสถียร จะมี Pulse ออกมาทาง Data Bus D1 หรือ D2 1 Pulse เสมอ ซึ่งจะนำ Pulse D1 และ D2 ไปเป็นสัญญาณทริก Counter แบบ Up Down ทำให้รู้ตำแหน่งของ Motorไม่ว่าจะหมุนไปทางไหน

ในการทำจริงไม่สามารถจะตรวจสอบทิศทางการเคลื่อนที่ออกมาได้ จึงไม่ได้ใช้ส่วนของ Eprom แต่เอาสัญญาณที่ได้จาก Encorder มาเข้าวงจร Counter เลย ทำให้ไม่สามารถหาทิศทางได้ ดังนั้นในโปรแกรม เมื่อต้องการจะเลื่อนฉากกั้นหลังก็จะรู้อยู่แล้วว่าจะเดินหน้าหรือถอยหลัง เมื่อไม่รู้ทิศทาง ทุกครั้งที่มีการเคลื่อนฉากกั้นหลังจึงต้องมีการ clear counter ก่อนทุกครั้ง ดังนั้นเมื่อเคลื่อนที่ไปถึงที่ใดจึงต้องเก็บค่าที่

ตำแหน่งนั้นไว้ในตัวแปรเสมอ ตัวแปรนี้ชื่อ Tempdis ดังตัวอย่างของโปรแกรมข้างล่างนี้

```
if not (tempfile[select] = 0) then
begin
     if tempfile[select] > tempdis then                    {Reve}
     begin
          if ((tempfile[select] - tempdis) > 50)  then
          begin
               step := round((tempfile[select] - tempdis) * Invk);
               stepbreak := step - stepbreak5v;
               Port[Pout1] := (tempsw or ReveM2);  {Reve}
               delay(100);
               Port[Pout2] := Speed5;
               delay(100);
               Port[Pread3] := $80;              {Clear Count}
               Delay(10);
               Port[Pread3] := $00;
               Delay(10);
               Repeat
                    occurstep1 := ((tempdis) + (occurstep * k));
                    position(occurstep1);
               Until (Occurstep >= stepbreak);
               Port[pout2] := Speed_5;
               delay(100);
               Repeat
                    occurstep1 := ((tempdis) + (occurstep * k));
                    position(occurstep1);
               Until (Occurstep >= step);
```

```
        Port[Pout1] := (tempsw and (not ReveM2));
        delay(100);
        Port[Pout2] := Nor;
        delay(100);
        for i := 1 to 10 do
        begin
             occurstep1 := ((tempdis) + (occurstep * k));
             position(occurstep1);
        end;
        tempdis := occurstep1;
    end
    else
    begin
        step := round((tempfile[select] - tempdis) * Invk);
        stepbreak := step - stepbreak2v;
        Port[Pout1] := (tempsw or ReveM2);  {Reve}
        delay(100);
        Port[Pout2] := Speed2;
        delay(100);
        Port[Pread3] := $80;                {Clear Count}
        Delay(10);
        Port[Pread3] := $00;
        Delay(10);
        Repeat
             occurstep1 := ((tempdis) + (occurstep * k));
             position(occurstep1);
        Until (Occurstep >= stepbreak);
        Port[pout2] := Speed_5;
```

```
                delay(100);
                Repeat
                      occurstep1 := ((tempdis) + (occurstep * k));
                      position(occurstep1);
                Until (Occurstep >= step);
                Port[Pout1] := (tempsw and (not ReveM2));
                delay(100);
                Port[Pout2] := Nor;
                delay(100);
                for i := 1 to 10 do
                begin
                    occurstep1 := ((tempdis) + (occurstep * k));
                    position(occurstep1);
                end;
                tempdis := occurstep1;
          end;
    end
    else
    begin                                                {Forw}
         if ((tempdis - tempfile[select]) > 50) then
         begin
                step := round((tempdis - tempfile[select]) * Invk);
                stepbreak := step - stepbreak5v;
                Port[Pout1] := (tempsw or ForwM2);  {Forw}
                delay(100);
                Port[Pout2] := Speed5;
                delay(100);
                Port[Pread3] := $80;                {Clear Count}
```

```
        Delay(10);
        Port[Pread3] := $00;
        Delay(10);
        Repeat
              occurstep1 := ((tempdis) - (occurstep * k));
              position(occurstep1);
        Until (Occurstep >= stepbreak);
        Port[pout2] := Speed_5;
        delay(100);
        Repeat
              occurstep1 := ((tempdis) - (occurstep * k));
              position(occurstep1);
        Until (Occurstep >= step);
        Port[Pout1] := (tempsw and (not ForwM2));
        delay(100);
        Port[Pout2] := Nor;
        delay(100);
        for i := 1 to 10 do
        begin
             occurstep1 := ((tempdis) - (occurstep * k));
             position(occurstep1);
        end;
        tempdis := occurstep1;
    end
    else
    begin
        step := round((tempdis - tempfile[select]) * Invk);
        stepbreak := step - stepbreak2v;
```

```
Port[Pout1] := (tempsw or ForwM2);  {Forw}
delay(100);
Port[Pout2] := Speed2;
delay(100);
Port[Pread3] := $80;              {Clear Count}
Delay(10);
Port[Pread3] := $00;
Delay(10);
Repeat
     occurstep1 := ((tempdis) - (occurstep * k));
     position(occurstep1);
Until (Occurstep >= stepbreak);
Port[pout2] := Speed_5;
delay(100);
Repeat
     occurstep1 := ((tempdis) - (occurstep * k));
     position(occurstep1);
Until (Occurstep >= step);
Port[Pout1] := (tempsw and (not ForwM2));
delay(100);
Port[Pout2] := Nor;
delay(100);
for i := 1 to 10 do
begin
     occurstep1 := ((tempdis) - (occurstep * k));
     position(occurstep1);
end;
tempdis := occurstep1;
```

```
            end;
        end;
        inc(select);
    end;
    repeat
          tempin := port[pin1];
    until ((tempin and $80) = $80);
    repeat
          tempin := port[pin2];
    until ((tempin and $01) = $01);
    repeat
          tempin := port[pin1];
    until ((tempin and $40) = $40);
```

บทที่ 4.

โปรแกรมควบคุมเกี่ยวกับการสร้างหน่วยแสดงผลตัวเลขขนาดใหญ่

โปรแกรมควบคุมส่วนนี้จะแสดงตำแหน่งของแกน ณ ตำแหน่งต่าง ๆ เนื่องจากการเคลื่อนที่ของแกน ระยะของการเคลื่อนที่จะเคลื่อนที่ได้จาก 000.00 ถึง 999.99 (หน่วยเป็นมิลลิเมตร) จึงทำให้ต้องใช้ตัวเลขทั้งหมด 5 หลัก และเนื่องจากการแสดงผลในส่วนนี้ต้องอาศัยความเร็วสูง เพื่อให้ทันกับการเคลื่อนที่ของแกนที่เคลื่อนไปได้จริง จึงไม่สามารถใช้คำสั่ง write ธรรมดาได้ เพราะการใช้คำสั่ง write จะต้องกระทำผ่าน interup ของ DOS แล้วจึงไปเขียนลงใน Video Ram ซึ่งทำให้เสียเวลาเพิ่มขึ้น การแก้ไขทำได้โดยการ set ตัวแปร direct video ให้เป็น true (เป็นBoolean) จะทำให้การใช้คำสั่ง write จะไม่ผ่าน interrupt ของ DOS แต่เขียนลงในส่วนของVideo Ram โดยตรงเลย ทำให้เร็วขึ้น (ประมาณ 2 เท่า) ซึ่งโปรแกรมส่วนใหญ่จะสร้างเป็น Procedure

4.1 <u>การปิด CURSOR และ การเปิด CURSOR</u> Procedure นี้ปิด cursor และเปิดcursor เพราะในส่วนของการสแกนตัวเลขทั้ง 5 หลักด้วยความเร็ว ถ้าไม่ off cursor จะเกิด cursor ขึ้นเป็นเหมือนขยะที่เราไม่ต้องการ จึงต้องมีการ off cursor ซึ่งการควบคุม cursor จะควบคุมผ่าน register ของ IC 6845 ซึ่ง register cursor อยู่ที่ $0040 : $0060 และการติดต่อจะผ่าน ณ ตำแหน่ง $0040 : $0063 การนำค่า DF มา AND กับตำแหน่งเริ่มต้น cursor จะทำให้ cursor หายไป และนำค่า 20 มา or กับตำแหน่งเริ่มต้น cursor จะทำให้ curror กลับมา

อ้างตำแหน่งเริ่มต้น cursor --> Port(Vport) := 10;

นำค่า $DF มา AND กับเส้นเริ่มต้นของ

cursor จะทำให้ cursor หายไป--> Port(Vport+1) := Hi(cursor mode)and

อ้างตำแหน่งสุดท้ายของ cursor --> Port(Vport) :=11;

Port(Vport+1) := Lo (cursor mode)

4.2 <u>การสร้างตัวเลขขนาดใหญ่</u> Procedure จะทำการเขียน Text เป็น Block มาต่อกันเป็นตัวเลขที่ต้องเขียนอยู่ใน Text mode เพราะต้องการความเร็ว เพราะถ้าใช้ Graphic mode จะช้า แต่จะละเอียดและสวยกว่า จากความต้องการจึงเลือก mode Text Procedure นี้จะรับค่าตำแหน่งของจอภาพที่จะต้องนำตัวเลขนี้ไปวาง 2 ค่า

```
procedure  put0(x,y : byte);
begin
    textattr := $0E;
    gotoxy(x,y);
    write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219,'      ',#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219,'      ',#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219,'      ',#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219,'      ',#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219,'      ',#219,#219);
    y := y + 1;
    gotoxy(x,y);
```

```
        write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
        textattr := $0F;
    end;
```

<u>การกำหนดหลักตัวเลขทั้ง 5 หลัก</u> Procedure หาหลัก 100 จะรับค่าตำแหน่งของการเคลื่อนที่เข้ามาแล้วหาร 100 แล้วปัดเศษทิ้ง แล้วมาเข้า case เพื่อตรวจว่าเป็นเลขอะไร และกำหนดตำแหน่งของหลัก 10 ด้วย

วิธีในการหาหลักร้อย

```
a  := a/100;
a1 := trunc(a);
```

วิธีในการหาหลักสิบ

```
b  := b/100;
b  := frac(b);
b  := b * 10;
b1 := trunc(b);
```

วิธีในการหาหลักหน่วย

```
c  := c/10;
c  := frac(c);
c  := c * 10;
c1 := trunc(c);
```

วิธีในการหาทศนิยมหลักที่หนึ่ง

```
d  := frac(d);
d  := d * 10;
d1 := trunc(d);
```

วิธีในการหาทศนิยมหลักที่สอง

```
e  := frac(e);
e  := e * 10;
e  := frac(e);
e  := e * 10;
e1 := trunc(e);
```

Procedure ในการแสดงค่าตัวเลขทั้ง 5 หลัก

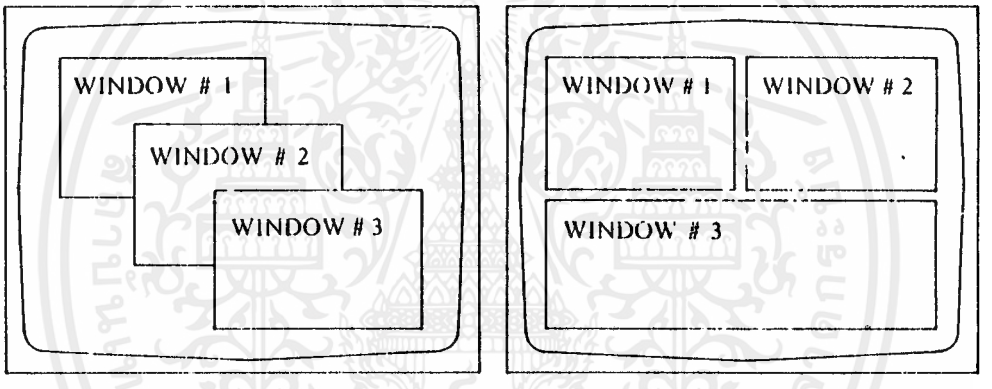
```
Procedure Position(r : real);
begin
     window(1,1,80,25);
     puthun(r);
     putten(r);
     putone(r);
     point(45,19);
     putpointone(r);
     putpointten(r);
end;
```

4.3 <u>การกำหนดการเขียนโดยไม่ผ่านอินเตอร์รัพต์ของ DOS</u> Procedure นี้จะเป็นการกำหนดตัวแปร checksnow และ Direct Video ซึ่งเป็น Boolean ให้มีค่าเป็น on ถ้าไม่ต้องการผ่าน interup DOS และ off จะผ่าน interup DOS

```
Direct Video := true;        ไม่ผ่าน  interup DOS
Checksnow    := true;
```

4.4 <u>แบบของจอหน้าต่าง</u> ลักษณะหน้าต่างอาจแบ่งออกได้เป็น 2 ประเภท คือ หน้าต่างที่มีการซ้อนกันหลาย ๆ ชั้น และหน้าต่างที่แยกจากกันหรือไม่มีส่วนที่ทับกัน

หน้าต่างแบบซ้อนกันเหมาะสำหรับงานที่แจกแจงเป็นรายละเอียดย่อยๆ ทำให้ผู้ใช้รู้สึกว่ากำลังใช้งานย่อยลึกลงไปเรื่อยๆ ตามจำนวนชั้นของการซ้อนส่วนหน้าต่างที่ไม่ซ้อนทับกันจะให้ความรู้สึกกับผู้ใช้ว่า จอภาพถูกแบ่งออกเป็นจอย่อยหลายจอ หน้าต่างแบบนี้จึงนิยมใช้แสดงเรื่องราวหรือข้อความหลายๆอย่างในเวลาเดียวกัน แต่อย่างไรก็ตามการใช้งานจะต้องอยู่ที่หน้าต่างใดหน้าต่างหนึ่งเท่านั้น ตัวอย่างของหน้าต่างทั้งสองแบบแสดงได้ดังรูปที่ 15



(ก) หน้าต่างซ้อนกันให้ความรู้สึกถึงการแจกแจงงานย่อย

(ข) หน้าต่างแยกกันให้ความรู้สึกถึงการใช้งานหลายอย่างพร้อมกัน

รูปที่ 15 แสดงระบบจอภาพหน้าต่าง

การแสดงผลแบบจอหน้าต่าง จะมีส่วนช่วยให้ผู้ใช้รู้สึกใช้งานได้สะดวกและคล่องตัวขึ้น เราสามารถนำเทคนิคของหน้าต่างไปใช้ในการสร้างโปรแกรมประเภทป๊อปอัพ (Pop-Up) อย่างเช่น ไซด์คิก ซึ่งเมื่อกดปุ่ม Ctrl - Alt แล้วจะมีหน้าต่างปรากฏ หรือการสร้างโปรแกรมระบบพูลดาวน์เมนู (Pull-Doun Menu) อย่างเช่นการใช้ ASSIST ในดีเบส เอดิเตอร์ของเทอร์โบซี หรือ เทอร์โบปาสคาล 4.0

วินโดว์ในเทอร์โบปาสคาล

เทอร์โบปาสคาล 4.0 มีคำสั่งสร้างหน้าต่าง โดยการกำหนดตำแหน่งมุมบนซ้ายและมุมล่างขวาของกรอบหน้าต่างด้วยคำสั่ง

```
Window (x1,y1,x2,y2)
```

เมื่อ (x1,y1) เป็นตำแหน่งมุมบนซ้าย

(x2,y2) เป็นตำแหน่งมุมล่างขวา

ผลจากการใช้คำสั่ง Window จะทำให้มีการจำกัดขอบเขตของการแสดงผลให้อยู่ภายในกรอบหน้าต่างเท่านั้น ดังนั้นคำสั่งที่เกี่ยวข้องกับการแสดงผลบนจอภาพได้แก่คำสั่ง Read, Readln, Write, Writeln, ClrScr และ GotoXY หรือการเลื่อนจอภาพจะเกิดผลในบริเวณหน้าต่างที่สร้างขึ้นเท่านั้น ในการปรับช่องหน้าต่างให้กลับคืนสภาวะปกติ ก็เพียงแต่สร้างกรอบหน้าต่างให้เท่ากับจอปกติ ด้วยคำสั่ง

```
Window (1,1,80,25)
```

เมื่อ (1,1) คือตำแหน่งขอบบนซ้ายสุดของจอภาพ

(80,25) คือตำแหน่งขอบล่างขวาสุดของจอภาพ

คำสั่ง Window ของเทอร์โบปาสคาล 4.0 มีขีดจำกัดต่อการใช้งานอยู่มาก ข้อที่นับว่าเป็นจุดด้อยอย่างมากก็คือ ไม่สามารถเก็บข้อความเดิมบนจอภาพที่ช่องหน้าต่างที่เกิดขึ้น การเขียนข้อความใหม่ลงไปในช่องหน้าต่างจึงเท่ากับเป็นการทำลายข้อความเดิมบนจอ ข้อจำกัดอื่นก็คือ ไม่สามารถปิดช่องหน้าต่างที่เพิ่งสร้างขึ้น เราทำได้เพียงแต่สร้างช่องหน้าต่าง

ใหม่โดยที่ข้อความเดิมของช่องหน้าต่างก็ยังคงที่ไม่เปลี่ยนแปลง อีกทั้งยังไม่สามารถสร้างหน้าต่างซ้อนกันหลายๆชั้นได้ จะเห็นได้ว่า คำสั่ง Window ของเทอร์โบปาสคาล 4.0 ก็เป็นเพียงคำสั่งพื้นฐานที่เราต้องนำไปใช้สร้างระบบหน้าต่างที่แท้จริงด้วยตนเอง

โครงงานสร้างระบบหน้าต่าง

ก่อนที่เราจะสร้างระบบหน้าต่าง เราจะต้องกำหนดความสามารถของระบบหน้าต่างเพื่อชดเชยจุดด้อย และเพิ่มขีดความสามารถที่ไม่มีอยู่ในคำสั่ง window ของเทอร์โบปาสคาล 4.0 ดังนี้

- ช่องหน้าต่างที่เกิดขึ้นจะมีกรอบปรากฏบนจอภาพเพื่อบ่งบอกขอบเขตช่องหน้าต่างให้ชัดเจน
- สามารถเลือกแบบของกรอบหน้าต่างได้หลายแบบ
- สามารถเก็บข้อความเดิมและเรียกข้อความเดิมที่เกิดช่องหน้าต่างกลับมาได้ รวมทั้งตำแหน่งเคอร์เซอร์จะไม่มีการเปลี่ยนแปลงด้วย
- หน้าต่างที่สร้างขึ้นสามารถซ้อนกันเป็นชั้นๆได้
- สามารถสร้างข้อความส่วนหัวหรือเฮดเดอร์ของช่องหน้าต่างซ้อนบนกรอบได้
- ทุกส่วนของหน้าต่างไม่ว่าจะเป็นส่วนของกรอบ ส่วนเฮดเดอร์ของกรอบ พื้นของช่องหน้าต่าง และตัวอักษรที่ปรากฏบนช่องหน้าต่างสามารถมีแอตตริบิวต์แบบใดก็ได้ตามแบบของแอตตตริบิวต์ที่ได้กล่าวไว้แล้ว โดยขึ้นอยู่กับการกำหนดของผู้ใช้

โครงสร้างช่องหน้าต่าง

จากข้อกำหนดที่กล่าวในหัวข้อที่แล้ว การออกแบบโครงสร้างของช่องหน้าต่างให้มีคุณสมบัติครบถ้วนตามที่กำหนด จำเป็นต้องเก็บข้อมูลของช่องหน้าต่างทุกๆช่องไว้เพื่อให้ตัวหน้าต่างสามารถซ้อนกันได้หลายๆชั้น ข้อมูลที่เราจำเป็นต้องเก็บมีดังนี้

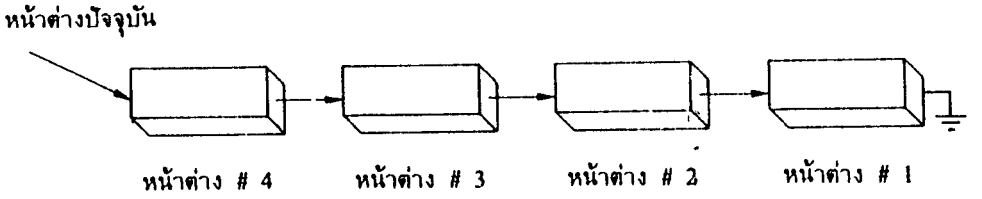
- ค่า X1, Y1, X2, Y2 ซึ่งกำหนดขอบเขตของกรอบหน้าต่าง
- ค่า X,Y เก็บตำแหน่งเคอร์เซอร์ปัจจุบัน
- ค่า ID บอกหมายเลขของช่องหน้าต่าง

- BackLink เป็นพอยน์เตอร์เชื่อมโยงหน้าต่าง
- ScreenContents เก็บข้อความเดิมทั้งหมดของช่องหน้าต่าง

การสร้างหน้าต่างให้มีคุณสมบัติซ้อนกันได้หลายชั้น ต้องใช้หลักการโครงสร้างข้อมูลเข้าช่วยเพื่อให้หน้าต่างเชื่อมโยงกันได้ ทั้งนี้เมื่อยกเลิกหน้าต่างปัจจุบัน โปรแกรมต้องย้อนไปเปิดหน้าต่างถัดไปได้ โครงสร้างข้อมูลที่เหมาะสมที่เราสามารถนำมาใช้ได้ก็คือ *ลิงก์ลิสต์ (link list)* เนื่องจากเราไม่ทราบว่าจะมีการเปิดใช้ช่องหน้าต่างเป็นจำนวนเท่าใด เราจะใช้คุณสมบัติตัวแปรพอยน์เตอร์ของเทอร์โบปาสคาล4.0 สำหรับใช้เชื่อมโยงจากหน้าต่างหนึ่งไปยังอีกหน้าต่างหนึ่ง เพื่อทำให้เกิดช่องหน้าต่างที่ซ้อนทับกันได้ โครงสร้างหน้าต่างที่ออกแบบจะอยู่ในรูปของเรคอร์ดดังนี้

```
type ScreenLine   =   array[1..80]  of integer;
     ScreenArray  =   array[1..25]  of ScreenLine;
     ScreenBlock  =   array[1..2000] of integer;
     WindowLink   =   ^WindowControlBlock;
     WindowControBlock  =  record
                             X1,Y1,X2,Y2     :byte;
                             X,Y             :byte;
                             Id              :byte;
                             BackLink        :WindowLink;
                             ScreenContents :ScreenBlock;
                           end;
```

สำหรับลักษณะลิงก์ลิสต์ของหน้าต่างจะเป็นดังรูปที่ 4.2 ซึ่งจะแสดงตัวอย่างเมื่อเปิดถึงหน้าต่างที่ 4 ดังนี้



รูปที่ 16 โครงสร้างลิงก์ลิสต์ของช่องหน้าต่าง

การเก็บช่องหน้าต่าง

เมื่อเราสร้างหน้าต่างแล้ว ต้องมีการเก็บข้อความเดิมไว้ เนื่องจากช่องหน้าต่างแต่ละช่องอาจมีขนาดไม่เท่ากัน จำนวนตัวอักษรที่เก็บจึงไม่แน่นอนขึ้นอยู่กับขนาดหน้าต่างโดยตรง เราอาจใช้วิธีการง่ายๆคือ เก็บทั้งจอภาพ ซึ่งวิธีนี้จะเขียนโปรแกรมได้ง่าย แต่เปลืองพื้นที่หน่วยความจำค่อนข้างมาก ดังนั้นในที่นี้เราจึงออกแบบโปรแกรมที่จะเก็บตัวอักษรเฉพาะช่องหน้าต่างหนึ่งเท่านั้น

ปัญหาสำคัญที่เราต้องหาคำตอบก็คือ ช่องหน้าต่างแต่ละช่องมีขนาดได้ต่างๆ กัน เราจึงควรใช้วิธีในการเก็บข้อมูลในช่องหน้าต่างอย่างประหยัดที่สุดโดยเก็บข้อความเฉพาะหน้าต่างนั้นโดยไม่เก็บทั้งจอภาพ วิธีนี้จะทำให้เราใช้พื้นที่ของหน่วยความจำได้เท่ากับขนาดของช่องหน้าต่างในการเปิดหน้าต่างแต่ละครั้ง

ถ้าเราย้อนกลับไปพิจารณาเรคอร์ด WindowControlBlock จะเห็นว่า ScreenContents ซึ่งใช้เป็นที่เก็บข้อความของช่องหน้าต่างจะมีแบบข้อมูลเป็นอะเรย์ของหน้าจอภาพพอดี การขอใช้หน่วยความจำไดนามิกมาสร้างลิงก์ลิสต์ สามารถทำได้โดยใช้คำสั่ง

```
New (P)
```

จากการใช้คำสั่ง New(P) จะได้ขนาดของหน่วยความจำเท่ากับขนาดของแบบข้อมูลที่กำหนดไว้ ซึ่งไม่ตรงกับความต้องการของเรา อย่างไรก็ตามเทอร์โบปาสคาล 4.0 ก็มีคำสั่งสำหรับการขอใช้หน่วยความจำตามขนาดที่เราต้องการโดยใช้คำสั่ง

```
GetMem (P,I)
```

ซึ่งเป็นการขอใช้หน่วยความจำขนาด I ไบต์ แล้วให้พอยน์เตอร์ P ชี้ไปที่จุดเริ่มต้นของหน่วยความจำนั้น และเมื่อหมดความจำเป็นที่ต้องการใช้งานก็สามารถคืนหน่วยความจำให้กับเทอร์โบปาสคาล 4.0 ได้ด้วยคำสั่ง

```
FreeMem(P,I)
```

การเก็บข้อมูลในช่องหน้าต่าง จึงต้องคำนวณขนาดของหน้าต่างที่ต้องใช้เก็บ ซึ่งสามารถคำนวณได้จากสูตร

$$I = (X2-X1+1)*(Y2-Y1+1)*2$$

เมื่อ (x1,y1) และ (x2,y2) เป็นตำแหน่งมุมซ้ายบนและมุมขวาล่างของกรอบหน้าต่าง ผลคูณ (X2-X1+1) กับ (Y2-Y1+1) คือ จำนวนตัวอักษรทั้งหมด และเรายังต้องเก็บแอตตริบิวต์ของตัวอักษรแต่ละตัว จึงต้องนำ 2 มาคูณเข้าไปหลังจากที่เราคำนวณขนาดหน้าต่าง และเพียงแต่ขอให้เทอร์โบปาสคาล 4.0 จัดสรรพื้นที่หน่วยความจำให้เราโดยใช้คำสั่ง GetMem แล้วจึงเก็บข้อมูลตัวอักษรกับแอตตริบิวต์ของช่องหน้าต่างไว้ในหน่วยความจำที่ได้มาด้วยคำสั่ง Move ก่อนที่จะนำไปสร้างลิงก์ลิสต์ของช่องหน้าต่าง

การตรวจสอบขนาดหน่วยความจำไดนามิก

หน่วยความจำไดนามิกที่เทอร์โบปาสคาล 4.0 จัดสรรให้ จะมีขนาดจำกัด คือไม่เกิน 1 เซกเมนต์หรือ 64 กิโลไบต์ ก่อนการใช้จึงจำเป็นต้องตรวจสอบดูว่าหน่วยความจำที่เหลืออยู่มีขนาดพอเพียงกับที่โปรแกรมต้องการใช้หรือไม่ เทอร์โบปาสคาล 4.0 มีฟังก์ชัน MemAvail สำหรับตรวจสอบขนาดหน่วยความจำที่เหลืออยู่

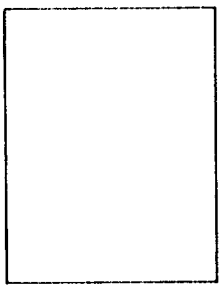
ฟังก์ชัน MemAvail ให้ค่าเป็น LongInt ในการตรวจสอบก่อนการเปิดช่องหน้าต่าง เราจึงใช้ขนาดของช่องหน้าต่างที่คำนวณได้ ซึ่งเก็บอยู่ในตัวแปร WindowSize มาเปรียบเทียบกับฟังก์ชัน MemAvail

การเรียกคืนข้อความเมื่อยกเลิกช่องหน้าต่าง

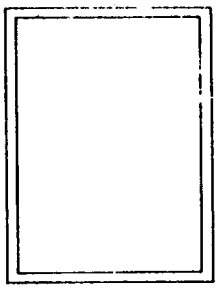
ในกรณีของการปิดหรือยกเลิกช่องหน้าต่าง จำเป็นต้องเรียกค่าที่เก็บไว้กลับคืนมา ขั้นตอนก็จะกลับกันคือ คำนวณขนาดหน้าต่างแล้วนำข้อมูลที่เก็บไว้กลับมาแสดงผลบนจอภาพโดยใช้คำสั่ง Move ย้ายข้อมูลไปสู่วิดีโอแรม แล้วจึงคืนค่าหน่วยความจำกลับคืนให้เทอร์โบปาสคาล 4.0 โดยใช้คำสั่ง FreeMem

การสร้างกรอบ

กรอบหน้าต่างมีประโยชน์สำหรับบ่งบอกขอบเขตหน้าต่าง แบบของกรอบหน้าต่างในที่นี้มีทั้งหมด 4 แบบ ดังแสดงในรูปที่ 17 ซึ่งผู้ใช้สามารถออกแบบเพิ่มเติมได้อีก



(ก) แบบเดี่ยวหรือ Single



(ข) แบบคู่หรือ Double

(ค) แบบผสมแบบที่ 1 หรือ Mix 1

(ง) แบบผสมแบบที่ 2 หรือ Mix 2

รูปที่ 17 แบบของกรอบหน้าต่าง 4 แบบที่สร้างไว้ให้ก่อน

การสร้างยูนิต WIN.PAS

โปรแกรม WIN.PAS ดังแสดงในโปรแกรม unit Win เป็นโปรแกรมยูนิตที่รวมกระบวนความสำหรับการสร้างหน้าต่าง ได้แก่ กระบวนความ WindowOpen , WindowClose และ InitWin รวมทั้งกระบวนความในการตั้งค่าพารามิเตอร์ของหน้าต่าง ได้แก่ กระบวนความ SetWinHeader , SetWinAttr , SetBoxAttr, SetHeadAttr , SetCharAttr และ SetBoxStyle ซึ่งมีการใช้งานดังนี้

```
unit Win;

interface
uses Crt,screen;
```

```
const  single    = 1;
double    = 2;
Mix1      = 3;
Mix2      = 4;
Boxstyle          : byte          = Single;
AttrOfBox         : byte          = HighDisplay;
AttrOfWindow      : byte          = HighDisplay;
AttrOfHeader      : byte          = HighDisplay;
AttrOfChar        : byte          = HighDisplay;
HeaderOfWindow    : string        = '';
TypeOfBox         : array [1..4,1..8] of char =
                    ((#196,#179,#179,#196,$218,#191,#192,#217),
                     (#205,#186,#186,#205,#201,#187,#200,#188),
                     (#205,#179,#179,#205,#213,#184,#212,#190),
                     (#196,#186,#186,#196,#214,#183,#211,#189)
                    );

   var   ErrorWindow : byte;

   procedure WindowBox  (x1,y1,x2,y2 : byte);
   procedure WindowOpen (x1,y1,x2,y2 : byte);
   procedure WindowClose;
   procedure FillWin(x1,y1,x2,y2,Acii,Attr:Byte);
   procedure SetBoxStyle (Attrib : byte);
   procedure SetBoxAttr  (Attrib : byte);
   procedure SetWinAttr  (Attrib : byte);
   procedure SetHeadAttr (Attrib : byte);
   procedure SetCharAttr (Attrib : byte);
```

```
procedure SetWinHeader (st : string);

        { end interface }


implementation


  Type   ScreenLine  = array [1..80] of integer;
         ScreenArray = array [1..25] of screenline;
         ScreenBlock = array [1..2000] of integer;
         WindowLink  = ^WindowControlBlock;
         WindowControlBlock = record
                                x1,y1,x2,y2 : integer;
                                x,y         : integer;
                                ID          : byte;
                                Backlink       :  WindowLink;
                                ScreenContents :  ScreenBlock;
                              end;
  var  ActiveWindow     : WindowLink;
       ScreenPtr        : ^ScreenArray;
       FixedSize        : integer;
       WindowCount      : byte;


  procedure WindowBox (x1,y1,x2,y2 : byte);
  const Top    = 1;  Left     = 2;
        Right  = 3;  Bottom   = 4;
        UpLeft = 5;  UpRight  = 6;
        Loleft = 7;  Loright  = 8;
  var   x,y  : byte;
        colorScr : array[1..25,1..80,1..2] of char absolute
```

```
                $B800:$0000;
    monoScr  : array[1..25,1..80,1..2] of char absolute
                $B000:$0000;
    crtType  : Byte absolute $0040:$0049;
begin
  Window(x1,y1,x2,y2);
  SetAttr(AttrOfWindow);
  clrscr;
  window(1,1,80,25);
  SetAttr(AttrOfBox);

    { top }
  Gotoxy (x1,y1);
  write (TypeOfBox[Boxstyle,Upleft]);
  for x := x1+1 to x2-1 do
    write(Typeofbox[boxstyle,top]);
  write (typeofbox[boxstyle,upright]);

    { side }
  for y := y1+1 to y2-1 do
    begin
      Gotoxy(x1,y); write(typeofbox[boxstyle,left]);
      Gotoxy(x2,y); write(typeofbox[boxstyle,right]);
    end;

    { bottom }
  Gotoxy(x1,y2);
  write(typeofbox[boxstyle,Loleft]);
```

```
  for x := x1+1 to x2-1 do
  write(typeofbox[boxstyle,bottom]);
  write(typeofbox[boxstyle,Loright]);

    { make it the current window and locate cursor to top }
  SetAttr(AttrOfHeader);
  Gotoxy ( (x1+x2-length(HeaderOfWindow))div 2,y1);
  write(HeaderOfwindow);
  for x:=x1+1 to x2+1 do
      if crtType<>7 then colorScr[y2+1,x,2]:=#7
                    else monoScr[y2+1,x,2]:=#7;

  for y:=y1+1 to y2+1 do
      if crtType<>7 then colorScr[y,x2+1,2]:=#7
                    else monoScr[y,x2+1,2]:=#7;

  window(x1+1,y1+1,x2-1,y2-1);
  SetAttr(AttrOfChar);
end;    { proc. boxwin }

procedure WindowOpen(x1,y1,x2,y2 : byte);
var  block       :  windowlink;
     Linelength,
     windowsize,
     I           :  integer;
     y           :  byte;
begin
  LineLength    := x2-x1+2;
```

```
windowsize   := linelength*(y2-y1+2)*2 + FixedSize;


  { check window valid }
if (x2 > 80) or (y2 > 25) or (x2-x1<2) or (y2-y1<2) then
   ErrorWindow  := 1
else
  if (abs(MemAvail) < WindowSize) then
     ErrorWindow := 2
  else
     ErrorWindow := 0;
if ErrorWindow = 0  then
   begin
     Getmem (Block,WindowSize);
     Block^.x1  :=  x1;
     Block^.x2  :=  x2;
     Block^.y1  :=  y1;
     Block^.y2  :=  y2;
     Block^.x   :=  whereX;
     Block^.y   :=  whereY;
     Block^.BackLink  := ActiveWindow;
     Activewindow     := Block;
     Windowcount      := windowcount + 1;
     block^.ID        := windowcount;
     I := 1;
     for y := y1 to y2+1 do
       begin
         Move(ScreenPtr^[y,x1],Block^.Screencontents[I],
         linelength*2);
```

```
            I := I + linelength;
          end;
        WindowBox(x1,y1,x2,y2);
      end;
  end;

procedure FillWin(x1,y1,x2,y2,acii,attr:byte);
  type ScreenArray = array[1..25,1..80,1..2] of byte;
  var crtType : byte absolute $0040:$0049;
    Monoscr : ScreenArray absolute $B000:$0000;
    Colrscr : ScreenArray absolute $B800:$0000;
    row,col : integer;
  begin { Procedure Fillwin}
    directvideo:=true;
    checksnow:=false;
    for row := y1 to y2 do
    for col := x1 to x2 do begin
          if crtType<>7 then begin
                Colrscr[row,col,1]:=acii;
                Colrscr[row,col,2]:=attr;
                      end
                  else  begin
                Monoscr[row,col,1]:=acii;
                Monoscr[row,col,2]:=attr;
                      end;
          end;
    directvideo:=false;
    checksnow:=true;
```

```
  end;


Procedure WindowClose;
var  block      :  windowlink;
     linelength,
     windowsize,
     I          :  integer;
     y          :  byte;
begin
  if activewindow <> nil then
  begin
    block       := ActiveWindow;
    Linelength  := block^.x2-block^.x1+2;
    windowsize  := linelength*(block^.y2-block^.y1+2)*2
                   + fixedsize;
    windowcount := windowcount - 1;
    I := 1;
    for y := block^.y1  to  block^.y2+1  do
      begin
        move(block^.screencontents[I],screenPtr^[y,block^.x1]
        ,linelength*2);
        I := I + linelength;
      end;
    ActiveWindow  := block^.Backlink;
    if activewindow = nil then
       window(1,1,80,25)
    else
       with activewindow^ do window(x1+1,y1+1,x2-1,y2-1);
```

```
    Gotoxy(block^.x,block^.y);
    FreeMem(block,Windowsize);
  end;
end;

procedure Initwin;
begin
  activewindow  := nil;
  fixedsize     := SizeOf(windowcontrolblock)-sizeof(screen
                   block);
  ScreenPtr     := Ptr(VideoSeg,0);
  Window (1,1,80,25);
  windowcount   := 0;
end;

procedure Setboxstyle (Attrib : byte);
begin
  boxstyle  := attrib;
end;

procedure SetWinHeader (st : string);
begin
  HeaderOfWindow  := st;
end;

procedure SetWinAttr (Attrib : byte);
begin
  AttrOfWindow  := Attrib;
```

```
end;

procedure SetBoxAttr(Attrib : byte);
begin
  AttrOfBox  := Attrib;
end;

procedure SetHeadAttr(Attrib : byte);
begin
  AttrOfHeader  := Attrib;
end;

procedure SetCharAttr (Attrib : byte);
begin
  AttrOfchar  := Attrib;
end;

begin
  InitWin;
end. { of unit }
```

การใช้งานของกระบวนการต่างๆในโปรแกรมยูนิต WIN.PAS มีดังนี้

WindowOpen เป็นกระบวนความเปิดช่องหน้าต่าง โดยกำหนดจุดโคออร์ดิเนต (X1,Y1) และ (X2,Y2) กระบวนความนี้จะสร้างกรอบหน้าต่าง แอตตริบิวต์ต่างๆของช่องหน้าต่าง และตรวจสอบความถูกต้องของโครงสร้างหน้าต่าง รวมทั้งขนาดหน่วยความจำไดนามิกที่เหลืออยู่ว่าพอเพียงต่อการเก็บข้อความของช่องหน้าต่างหรือไม่ โดยให้รหัส

ความผิดพลาดอยู่ในตัวแปร ErrorWindow

WindowClose	เป็นกระบวนความยกเลิกการใช้ช่องหน้าต่างแล้วย้อนกลับไปยังช่องหน้าต่างก่อนหน้านั้น
InitWin	เป็นกระบวนความกำหนดค่าเริ่มต้นสำหรับใช้ในการสร้างหน้าต่าง กระบวนความ IntWin ต้องเรียกใช้ก่อนเสมอ (เพียงครั้งเดียวก่อนที่จะมีการใช้ระบบหน้าต่าง)
SetWinHeader	เป็นกระบวนความกำหนดเฮดเดอร์ของหน้าต่าง
SetWinAttr	เป็นกระบวนความปรับแอตตริบิวต์ของพื้นหน้าต่าง
SetBoxAttr	เป็นกระบวนความปรับแอตตริบิวต์ของกรอบหน้าต่าง
SetHeadAttr	เป็นกระบวนความปรับแอตตริบิวต์ของเฮดเดอร์
SetCharAttr	เป็นกระบวนความปรับแอตตริบิวต์ของอักษรในจอหน้าต่าง
SetBoxStyle	เป็นกระบวนความเลือกแบบกรอบหน้าต่าง

ส่วนแบบข้อมูล ค่าคงที่ ตัวแปรของยูนิต WIN.PAS แสดงได้ดังตารางที่ 4.1 ตารางที่ 4.2 และตารางที่ 4.3 ตามลำดับ สำหรับโปรแกรมที่ 4.2 เป็นโปรแกรมทดสอบระบบหน้าต่าง โปรแกรมนี้จะสร้างหน้าต่างบนจอภาพ 5 หน้าต่าง โดยใช้กรอบหน้าต่างแบบต่างๆกัน

ตารางที่ 4.1 แบบข้อมูลในยูนิต WIN.PAS

ชื่อแบบข้อมูล	การใช้งาน
ScreenLine	อะเรย์ของจำนวนเต็มขนาด 80 หน่วยเป็นแบบข้อมูลของจอ
ScreenArray	อะเรย์ขนาด 25 หน่วยของ ScreenLine ใช้เป็นแบบข้อมูลของจอภาพหนึ่งหน้าจอ
ScreenBlock	อะเรย์ของตัวเลขจำนวนเต็ม 2000 ตัว ใช้เป็นแบบข้อมูลของจอภาพเช่นเดียวกับ ScreenArray แต่มองจอภาพเป็นบล็อคขนาดใหญ่ต่อเนื่องกันไป

WindowLink	พอยน์เตอร์ของ WindowControlBlock
WindowControlBlock	เรคอร์ดของหน้าต่างประกอบด้วย X1,Y1,X2,Y2เป็นขอบมุมบนซ้ายและมุมล่างขวาของหน้าต่าง X,Y เป็นตำแหน่งเคอร์เซอร์ปัจจุบัน

HeadString	ID เป็นหมายเลขหน้าต่าง BackLink เป็นพอยน์เตอร์ชี้ไปยังหน้าต่างก่อนหน้านั้น ScreenContent เป็นที่เก็บข้อมูลของจอภาพ สตริงขนาด 80 ตัวอักษรสำหรับเป็นเฮดเดอร์ของจอหน้าต่าง

ตารางที่ 4.2 ค่าคงตัวในยูนิต WIN.PAS

ชื่อค่าคงตัว	การใช้งาน
Single	เส้นกรอบหน้าต่างแบบเส้นเดี่ยว
Double	เส้นกรอบหน้าต่างแบบเส้นคู่
Mix1	เส้นกรอบหน้าต่างแบบเส้นเดี่ยวผสมเส้นคู่แบบที่ 1
Mix2	เส้นกรอบหน้าต่างแบบเส้นเดี่ยวผสมเส้นคู่แบบที่ 2
AttrOfWindow	แอตตริบิวต์ของพื้นหน้าต่าง ค่าปกติเป็น HighDisplay
AttrOfBox	แอตตริบิวต์ของเส้นกรอบหน้าต่าง ค่าปกติเป็นHighDisplaY
HeaderOfWindow	สตริงของเฮดเดอร์ ค่าปกติเป็นสตริงมีความยาวศูนย์
TypeOfBox	อะเรย์ของเส้นกรอบหน้าต่าง

ตารางที่ 4.3 ค่าตัวแปรในยูนิต WIN.PAS

ชื่อตัวแปร	การใช้งาน
ActiveWindow	เป็นพอยน์เตอร์ชี้ที่หน้าต่างที่ใช้งานปัจจุบัน
ScreenPtr	พอยน์เตอร์ชี้ที่วิดีโอแรม
FixedSize	ตัวแปรจำนวนเต็ม สำหรับเก็บขนาดข้อมูลที่ไม่เปลี่ยนแปลงใน

ErrorWindow	วินโดว์ ซึ่งได้แก่ X1, X2, Y1, Y2, X, Y ID และ BackLink เป็นตัวแปรที่บอกรหัสความผิดพลาดของการปิดช่องหน้าต่าง ถ้ามีค่า 0 หมายถึง ไม่มีความผิดพลาด ถ้ามีค่า 1 หมายถึง กำหนดกรอบไม่ถูกต้อง ถ้ามีค่า 2 หมายถึง หน่วยความจำไม่พอต่อการเก็บช่องหน้าต่าง

บทที่ 5

โปรแกรมควบคุมเกี่ยวกับแป้นพิมพ์

แป้นพิมพ์ของเครื่องพีซีมีไมโครโปรเซสเซอร์ 8048 ทำหน้าที่ในการควบคุมและตรวจรับการกดปุ่ม เมื่อผู้ใช้กดปุ่มใดไม่ว่าจะเป็นปุ่มตัวอักษรหรือกดปุ่มรหัสควบคุม เช่น Ctrl-A,Ctrl-B ฯลฯ 8048 จะส่งรหัสประจำปุ่มไปยังซีพียูทันที เราเรียกรหัสประจำปุ่มบนแป้นพิมพ์นี้ว่า สแกนโค๊ด (scan code) รหัสสแกนโค๊ดแต่ละปุ่มบนแป้นพิมพ์แสดงไว้ดังตารางที่ 2

Key	Normal	Shift	Ctrl	Alt
F1	0 59	0 84	0 94	0 104
F2	0 60	0 85	0 95	0 105
F3	0 61	0 86	0 96	0 106
F4	0 62	0 87	0 97	0 107
F5	0 63	0 88	0 98	0 108
F6	0 64	0 89	0 99	0 109
F7	0 65	0 90	0 100	0 110
F8	0 66	0 91	0 101	0 111
F9	0 67	0 92	0 102	0 112
F10	0 68	0 93	0 103	0 113
<--	0 75	52	0 115	none
-->	0 77	54	0 116	none
	0 72	56	none	none
	0 80	50	none	none
HOME	0 71	55	0 119	none

END	0 79	49	0 117	none

Key	Normal	Shift	Ctrl	Alt
PGUP	0 73	57	0 132	none
PGDN	0 81	51	0 118	none
INS	0 82	48	none	none
DEL	0 83	46	0 255	none
ESC	27	27	27	none
BACKSPACE	8	8	127	none
TAB	9	0 15	none	none
ENTER	13	13	10	none
A	97	65	1	0 30
B	98	66	2	0 48
C	99	67	3	0 46
D	100	68	4	0 32
E	101	69	5	0 18
F	102	70	6	0 33
G	103	71	7	0 34
H	104	72	8	0 35
I	105	73	9	0 23
J	106	74	10	0 36
K	107	75	11	0 37
L	108	76	12	0 38
M	109	77	13	0 50
N	110	78	14	0 49

O	111	79	15	0	24
P	112	80	16	0	25
Q	113	81	17	0	16

Key	Normal	Shift	Ctrl	Alt
R	114	82	18	0 19
S	115	83	19	0 31
T	116	84	20	0 20
U	117	85	21	0 22
V	118	86	22	0 47
W	119	87	23	0 17
X	120	88	24	0 45
Y	121	89	25	0 21
Z	122	90	26	0 44
[	91	123	27	none
\	92	124	28	none
]	93	125	29	none
'	96	126	none	none
0	48	41	none	0 129
1	49	33	none	0 120
2	50	64	0 3	0 121
3	51	35	none	0 122
4	52	36	none	0 123
5	53	37	none	0 124
6	54	94	30	0 125

7	55	38	none	0 126
8	56	42	none	0 127
9	57	40	none	0 128
*	42	none	0 144	none
Keypad +	43	43	none	none

Key	Normal	Shift	Ctrl	Alt
Keypad -	45	45	none	none
=	61	43	none	0 131
/	47	63	none	none
:	59	58	none	none
-	45	95	31	0 130

ตารางที่ 2

เราจะสังเกตเห็นได้ว่าปุ่มฟังก์ชั่นต่าง ๆ เช่น F1 ถึง F12,Ins,Del,Home End,PgUp,PgDn หรือปุ่มลูกศรทั้งสี่ทิศทางก็มีสแกนโค๊ดประจำปุ่มอยู่เช่นกัน การตรวจสอบปุ่มฟังก์ชั่นเหล่านี้จึงต้องตรวจจากสแกนโค๊ดที่อ่านได้ สำหรับปุ่มฟังก์ชั่นพิเศษอื่น ๆ เช่น Alt-F1 จะมีสแกนโค๊ดพิเศษประจำปุ่มที่เรียกว่า สแกนโค๊ดส่วนขยาย (extended scan code)ดังแสดงในตารางที่ 3

5.1 การตรวจสอบสแกนโค๊ด

ในเทอร์โบปาสคาล มีคำสั่งอ่านค่าตัวอักษรด้วยฟังก์ชั่น Readkey จะอ่านการกดแป้น

7	55	38	none	0 126
8	56	42	none	0 127
9	57	40	none	0 128
*	42	none	0 144	none
Keypad +	43	43	none	none

Key	Normal	Shift	Ctrl	Alt
Keypad -	45	45	none	none
=	61	43	none	0 131
/	47	63	none	none
:	59	58	none	none
-	45	95	31	0 130

ตารางที่ 2

เราจะสังเกตเห็นได้ว่าปุ่มฟังก์ชั่นต่าง ๆ เช่น F1 ถึง F12,Ins,Del,Home End,PgUp,PgDn หรือปุ่มลูกศรทั้งสี่ทิศทางก็มีสแกนโค๊ดประจำปุ่มอยู่เช่นกัน การตรวจสอบปุ่มฟังก์ชั่นเหล่านี้จึงต้องตรวจจากสแกนโค๊ดที่อ่านได้ สำหรับปุ่มฟังก์ชั่นพิเศษอื่น ๆ เช่น Alt-F1 จะมีสแกนโค๊ดพิเศษประจำปุ่มที่เรียกว่า สแกนโค๊ดส่วนขยาย (extended scan code)ดังแสดงในตารางที่ 3

5.1 การตรวจสอบสแกนโค๊ด

ในเทอร์โบปาสคาล มีคำสั่งอ่านค่าตัวอักษรด้วยฟังก์ชั่น Readkey จะอ่านการกดแป้น

พิมพ์หนึ่งครั้ง เมื่อเรากดปุ่มอักษร ค่าที่ได้ คือ รหัส 1 ไบท์ของตัวอักษรนั้น แต่ถ้าเรากดปุ่มฟังก์ชั่น เราจะได้รหัส 2 ไบท์ติดต่อกัน รหัสไบท์แรกจะเป็น รหัส NULL หรือ #00 ส่วนรหัสไบท์ที่สองเป็นรหัสสแกนโค๊ด ดังนั้น เราจึงใช้รหัส #00 เป็นรหัสตรวจสอบการกดปุ่มฟังก์ชั่นได้ กระบวนความในการตรวจสอบการกดปุ่มจึงเป็นดังนี้

```
Var Key     : char;
    Funckey : boolcan;
Procedure Readfunckey (Var Key : char);
begin
   Key     : = Readkey;
   if key <># 0 then
     Funckey  : = false;
   else
  begin
     Funckey  : = true;
     Key      : = Readkey;
   end;
end;
```

กระบวนความนี้จะให้รหัส 1 ไบท์อยู่ในตัวแปร key ซึ่งผู้ใช้จะตรวจสอบได้ว่ามีการกดปุ่มใด โดยดูจากแฟล็ก Funckey ถ้า Funckey มีค่าเป็น false แสดงว่ามีการกดปุ่มอักษรธรรมดา และตัวแปร key จะเก็บรหัสแอสกี้ของตัวอักษรนั้น แต่ถ้า Funckey มีค่าเป็น true แสดงว่ามีการกดปุ่มฟังก์ชั่นและตัวแปร key จะเก็บรหัสสแกนโค๊ดของตัวอักษรนั้น ดังแสดงในโปรแกรมในการทำงานจริง

5.2 โปรแกรมอ่านข้อมูลเฉพาะตัวเลข

คำสั่ง Read และ ReadIn ในเทอร์โบปาสคาลใช้สำหรับการอ่านข้อมูล

ตัวเลขหรือตัวอักษรก็ได้ ในกรณีที่เราใช้ในการอ่านข้อมูลตัวเลข ถ้าการป้อนข้อมูลเข้าไปไม่ถูกต้อง เช่น ป้อนข้อมูลเป็นตัวอักษรหรือใส่รหัสพิเศษเข้าไปจะเกิด Run-time Error ทำให้โปรแกรมหยุดทำงาน เหตุการณ์เช่นนี้อาจเกิดขึ้นได้ทั้งจากความตั้งใจหรือความผิดพลาดจากการป้อน อย่างไรก็ตาม โปรแกรมที่ดีควรมีความสามารถในการตรวจสอบความถูกต้องของข้อมูลหรือป้องกันความผิดพลาดที่อาจส่งผลให้โปรแกรมหยุดทำงานได้

เทคนิคการตรวจสอบและป้องกันความผิดพลาดจากการป้อนข้อมูล ทำได้โดยการเขียนกระบวนความเพื่ออ่านข้อมูลขึ้นโดยเฉพาะ กระบวนความจะรับข้อมูลเข้ามาอยู่ในรูปสตริงแล้วจึงใช้ฟังก์ชั่นในการแปลงสตริงไปเป็นตัวเลข พร้อมกับส่งรหัสเพื่อบ่งบอกถึงความถูกต้องของการแปลงตามมาด้วย คุณสมบัติทั้งหมดนี้มีอยู่ในกระบวนความดังโปรแกรมการทำงานจริง

จากโปรแกรมการทำงานจริง ประกอบด้วยกระบวนความที่สามารถอ่านข้อมูลตัวเลขและตรวจสอบได้ทันทีว่ากดปุ่มตัวเลขหรือไม่ นอกจากนี้เรายังสามารถกำหนดจำนวนหลักของตัวเลขที่จะป้อนเข้าไปได้อีกด้วย เช่น ยอมให้มีการป้อนตัวเลขเพียง 5 หลัก เป็นต้น

จากโปรแกรมการทำงานจริงมีกระบวนความหลักในการตรวจสอบ Key อยู่ 3 กระบวนความคือ

ReadInt เป็นกระบวนความอ่านตัวเลขจำนวนเต็ม การป้อนข้อมูลจึงต้องเป็นเฉพาะตัวเลขเท่านั้น นอกจากนั้นจะไม่รับข้อมูลอักขระอื่นใด กระบวนความนี้สามารถกำหนดความยาวของการป้อนได้โดยปกติแล้วตัวเลขจำนวนเต็มจะมีค่าบวกสูงสุดเท่ากับ 32767 คือความยาว 5 ตัวอักษร และมีค่าลบได้ถึง-32767รวม 6 ตัวอักษร (นับเครื่องหมายลบด้วย) ดังนั้น การใช้กระบวนความจึงควรจะกำหนดความยาวของการป้อนไว้ไม่เกิน 6 ตัวอักษร

การเรียกใช้กระบวนความต้องมีพารามิเตอร์ 3 ตัว ดังในโปรแกรมการเรียกใช้ด้วยReadInt (N,6,Code) พารามิเตอร์ตัวแรก คือ N เป็นตัวแปรที่ใช้เก็บค่าตัวเลขเมื่อการป้อนได้เสร็จสิ้นลง พารามิเตอร์ตัวที่สองในที่นี้คือ 6 หมายถึงความยาวของตัวอักษรที่ป้อนได้ พารามิเตอร์ตัวที่สามคือ Code เป็นตัวเก็บรหัสที่บ่งบอกผลการป้อนตัวเลข

ถ้าการป้อนเป็นไปอย่างถูกต้อง ตัวแปร N จะเก็บเลขจำนวนเต็มที่ป้อนและตัวแปร Codeจะมีค่าเป็น 0 แต่ถ้าป้อนไม่ถูกต้องจะถือว่าค่าในตัวแปรไม่นิยาม คือ เป็นค่าที่นำไปใช้ไม่ได้ และตัวแปร Code จะมีค่าเป็น 255 ซึ่งบ่งบอกว่าเกิดความผิดพลาดขึ้นและในกรณีที่ผู้ใช้กดปุ่ม Enterโดยไม่ป้อนอะไรเลยหรือกดปุ่ม Esc จะถือว่าเป็นการยกเลิกการป้อนข้อมูล ซึ่งในกรณีนี้ตัวแปรCodeจะมีค่าเป็น 1 ผลของค่าตัวเลขและรหัสนี้ได้สรุปไว้ดังตารางที่ 3

การป้อน	พารามิเตอร์ Num	พารามิเตอร์ Code
ป้อนตัวเลขถูกต้อง	เก็บค่าตัวเลขนั้น	0
ป้อนตัวเลขไม่ถูกต้อง	ไม่นิยม	255
กดปุ่ม enter โดยไม่ป้อนอะไร	ไม่นิยม	1
กดปุ่ม esc	ไม่นิยม	1

ตารางที่ 3 พารามิเตอร์แสดงผลการป้อนของกระบวนความ ReadInt และ ReadReal

ReadReal มีหลักการทำงานคล้ายกับ ReadInt แต่ใช้สำหรับการอ่าน ตัวเลขจำนวนจริง ช่วงของตัวเลขจำนวนจริงที่อ่านได้ จะเป็นไปตามข้อกำหนดของเทอร์โบปาสคาลคือ 10^{-38} ถึง 10^{+38} การป้อนตัวเลขจำนวนจริงอาจมีทศนิยมประกอบอยู่ด้วยและป้อนได้หลายแบบ เช่น เมื่อต้องการป้อนตัวเลข 0.45 อาจป้อนด้วยการพิมพ์ 0.45 หรือ.45ก็ได้ กระบวนความReadReal มีข้อจำกัดคือ ไม่สามารถป้อนตัวเลขในรูปเลขวิทยาศาสตร์ได้ กล่าวคือ ไม่อาจป้อนในรูปสัญลักษณ์ E เช่น 3E-08 นั้น คือ ต้องป้อนในรูปทศนิยมเสมอ

การเรียงใช้กระบวนความ ReadReal ต้องมีพารามิเตอร์ 3 ตัว เช่นเดียวกันกับกระบวนความ ReadInt ส่วนที่แตกต่างก็คือ พารามิเตอร์ ตัวเราต้องเป็นเลขจำนวนจริงสำหรับรหัสบ่งบอกความผิดพลาดจะมีรูปแบบเช่นเดียวกับ ReadInt

5.3 ส่วนของการตรวจสอบโปรแกรม

```
Function Mode (VAR fi : fitype) : Boolean;
Begin
     { $I - } Reset(fi);  { $I + }
     Mode : = ( Ioresult = 0 );
End;
```

เป็นการเปิด file เพื่อตรวจสอบว่าโปรแกรมที่ป้อนเข้าไปในโปรแกรมถูกต้องหรือไม่ โดยแสดงค่า error เป็น 1 ที่ Mode ถ้ามีโปรแกรมอยู่จะมีค่า Iore = 0 ทำให้ Mode เป็น 0 ด้วย การตรวจสอบดูว่ามีไฟล์ในไดเรคทอรีหรือไม่จะใช้กรรมวิธีการดำเนินการตามโปรแกรม เพราะถ้าไม่มีไฟล์นี้อยู่จะทำให้เกิด run time error เป็นผลให้โปรแกรมไปทำงานในส่วนของการเขียนโปรแกรมแทน

เมื่อมีข้อมูลในไฟล์แล้วขั้นตอนต่อไปของการดำเนินกรรมวิธีข้อมูล คือ การอ่านข้อมูลจากไฟล์มาดำเนินกรรมวิธีตามความต้องการ ซึ่งในที่นี้จะนำออกแสดงทางจอภาพ การอ่านเริ่มด้วยการตรวจดูไฟล์แล้วพบว่ามีไฟล์ที่กำหนดในไดเรคทอรี

5.4 ส่วนของการอ่านโปรแกรม

```
write ('file name'i); read In(name);
assign (nfi,name);
     if made (nfi) then
          begin
             Cntr : = Cntr + 1;
             read (nfi,k);
             write ('step',CNTR,':');
          end;
```

```
        close(nfi);
```

5.5 ส่วนของการเขียนโปรแกรม

```
begin
   assign(nfi,name);
   rewrite(nfi);
  repeat
     cntr : cntr + 1;
     write('step',CNTR,':');
     read In(k);
     if k <> ' ' then write (nfi,k);
     until k = ' ';
     close (nfi);
  end;
end;
```

ใช้เป็นการเขียนเมื่อตรวจสอบแล้วพบว่าไม่มีไฟล์ชื่อนั้น ๆ อยู่ภายในโปรแกรมโดยใช้คำสั่ง REWRITE [nfi] จากนั้นจึงเขียนค่าของโปรแกรมลงไปเก็บไว้ในไฟล์ เพื่อเป็นการสร้างไฟล์

5.6 ข้อมูลที่ใช้ในการเคลื่อนที่แกน X

จะมีการ run งานตาม step ที่ปรากฏบนจอภาพโดยมี step ตามที่เราต้องการป้อน

การเปิดไฟล์ จะดำเนินการใด ๆ กับไฟล์นั้นจะต้องทำการเปิดไฟล์นั้นก่อน เพื่อให้ไฟล์นั้นมีชื่ออยู่ในดิสค์ แบ่งออกเป็น 2 แบบ คือ

REWRITE เป็นการเปิดไฟล์เพื่อเขียน คือการบรรจุข้อมูลให้กับไฟล์ ซึ่งถ้าเป็นไฟล์ใหม่ก็

จะไม่มีชื่อไฟล์ในดิสค์ จึงเป็นไฟล์ว่าง ถ้าไฟล์เก่าจะยกเลิกข้อมูลเดิมทำให้เป็นไฟล์ว่างเช่น REWRITE [nfi]

RESET เป็นการเปิดไฟล์เพื่ออ่านหรือเขียนกับไฟล์ ซึ่งต้องเป็นไฟล์ที่มีชื่อไฟล์ในดิสค์อยู่แล้ว เช่น RESET [nfi]

การเปิดไฟล์ไม่ว่าจะด้วย REWRITE หรือ RESET จะทำให้ไฟล์พอยท์เตอร์ชี้ที่ต้นไฟล์ คือที่คอมโพเนนท์ที่ 0

ส่วนในการอ่านหรือเขียนกับไฟล์ การอ่านจะเป็นการอ่านข้อมูล 1 ตัวจากไฟล์ตรงคอมโพเนนท์ที่ไฟล์พอยท์เตอร์ชี้อยู่ให้แก่ตัวแปร การเขียนจะนำค่าในตัวแปรหรือค่าคงตัวซึ่งเป็นค่าข้อมูล 1 ตัวไปเขียนลงไฟล์ตรงคอมโพเนนท์ไฟล์พอยท์เตอร์ชี้อยู่ ทุกครั้งที่อ่านหรือเขียนกับไฟล์ ไฟล์พอยท์เตอร์จะเพิ่มค่าขึ้นอัตโนมัติ

การปิดไฟล์ เมื่อใช้งานแล้วต้องปิดด้วยทุกครั้งด้วยคำสั่ง Close [nfi] ทั้งนี้เพื่อยกเลิกบัฟเฟอร์ และถ้าเป็นการเขียนก็จะนำไฟล์ลงไปเก็บในไฟล์ให้ด้วย ถ้าตกค้างอยู่ในบัฟเฟอร์

5.7 รับค่าจากไฟล์ระยะทาง

รับค่าจากไฟล์ระยะทางมาเก็บไว้ในตัวแปร X นำค่าที่อยู่ในตัวแปร X มาหา step โดยหารด้วยค่า K (K เป็นระยะทาง) จะได้เป็นจำนวน step เก็บไว้ในอีกตัวแปรหนึ่ง Y นำค่าที่อยู่ในตัวแปร Y มาคำนวณการ CONTROL MOTOR จะควบคุมโดย D/A 0-5 V ไป CONTROL INVERTER โดย CURVE ของการเคลื่อนที่ของ MOTOR เป็นดังรูปที่ 15 มอเตอร์ที่ใช้เป็น THREE PHASE MOTOR

บทที่ 6.

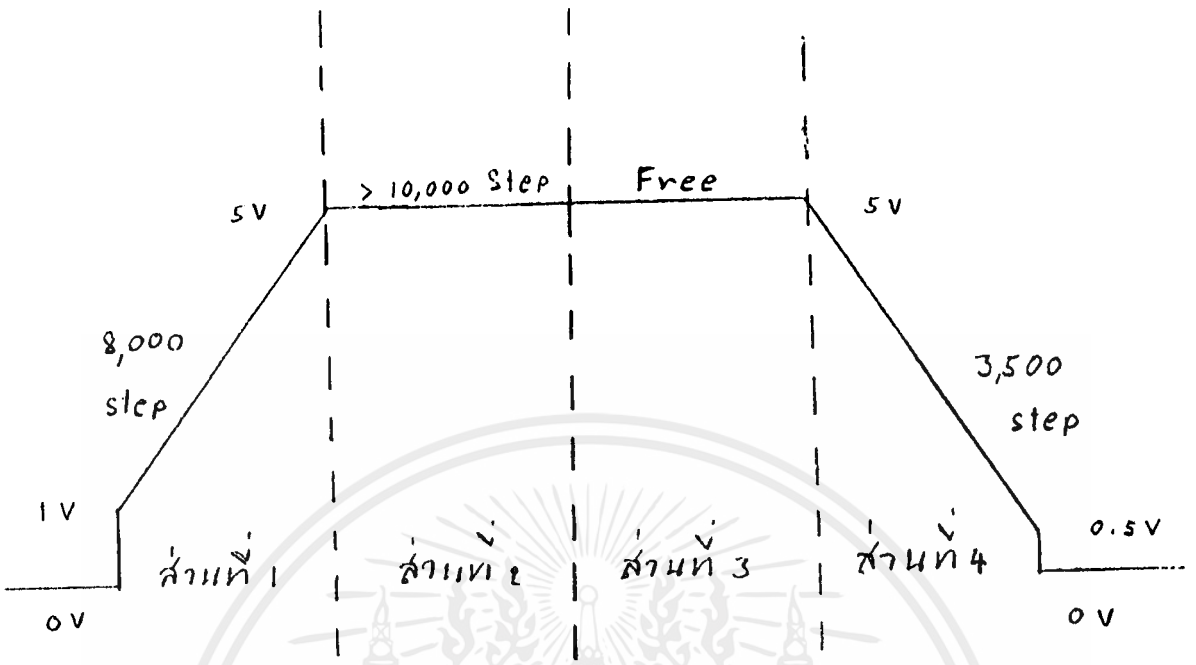
การคำนวณและการควบคุมระยะทางของ MOTOR

ในการควบคุม MOTOR จะใช้ INVERTOR เข้ามาช่วยในการควบคุม MOTOR THREE PHASE เพื่อให้การเคลื่อนที่ของ MOTOR มีความผิดพลาดน้อยที่สุด เพราะตัว INVERTOR เองมีความสามารถในการติดต่อกับ CONPUTER ได้(ในช่วง 0-5 หรือ 0-10V ดังนั้น ถ้า COMPUTER มีชุด D/A (digital to Analog Convertor) อยู่ในช่วง 0-5 หรือ 0-10 V แต่ในงานนี้ใช้ 0-5 V เพท่อเป็นแรงดันที่จะไปควบคุมความเร็วของ Motor ให้เคลื่อนที่ช้าเร็วตามการควบคุม D/A ของ computer จึงทำให้การควบคุม Motor มีความเที่ยงตรงมากขึ้น

อีกส่วนที่สำคัญในการตรวจสอบการเคลื่อนที่ของ MOTOR ก็คือ ตัว ENCODER และวงจร COUNTER จะเป็นตัวตรวจสอบระยะทางการเคลื่อนที่ของ MOTOR โดยส่งเป็น step ต่อระยะทาง มาให้ชุด COUNTER นั้นเก็บไว้เพื่อให้ COMPUTER มาเอาค่าไปคำนวณระยะทางต่อ step ว่าเคลื่อนที่ไปได้ระยะทางเท่าใดแล้ว เพื่อจะสามารถได้ระยะทางที่ถูกต้องยิ่งขึ้น

ดังนั้น การควบคุมระยะทางทำได้โดย COMPUTER ส่ง VOLTAGE ไปควบคุม INVERTOR (ในช่วง 0-5 V) และ INVERTOR จะไปควบคุม MOTOR อีกทีหนึ่งและจะมี ENCODER เป็นตัววัดระยะทางการเคลื่อนที่ของ MOTOR ส่งให้ชุด COUNTER เก็บค่าไว้แล้ว COMPUTER จะมาเอาค่าจี้ไปคำนวณหาระยะทางว่าได้ระยะทางแล้วหรือยัง ถ้าได้แล้วก็จะสั่งให้ MOTOR หยุดหมุน

การทำงานและโปรแกรมควบคุม MOTOR จะเป็นดังนี้



รูป 15 แสดงความเร็วของ MOTOR

รูปที่ 15 เป็นรูปแสดงความสัมพันธ์ระหว่างความเร็วต่อจำนวน step (จำนวน step ได้จาก Encoder)

<u>ส่วนที่ 1</u> ส่วนนี้จะเป็นส่วนที่ Computer ส่ง Voltage 1-5v ไปควบคุม Invertor และ Inverter ไปควบคุม Motor อีกที เพื่อเพิ่มความเร็วของ Motor เป็นความเร็วสูงสุดที่ Invertor จะขับ Motor ได้ และ Encoder อ่านค่าระยะทางในช่วงนี้ได้ประมาณ 8000 step และจะมีการหน่วงเวลาทุกๆช่วงของการเพิ่ม voltage จาก 1-5v (เพิ่มขึ้นทีละ $01) ด้วยเวลา 30 ms เพื่อให้ Invertor สามารถควบคุมความเร็วได้ทัน เพราะ Computer ทำงานเร็วมาก

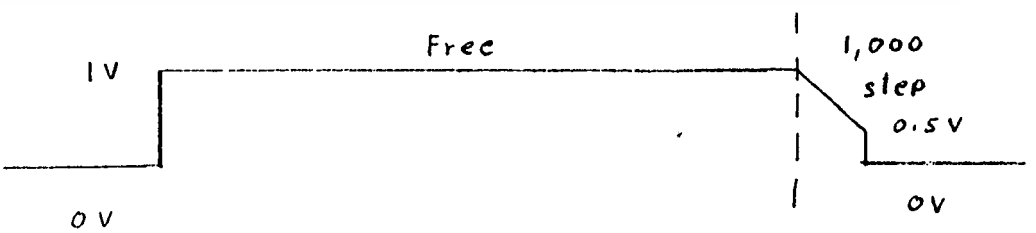
<u>ส่วนที่ 2</u> ส่วนนี้จะเป็นช่วงที่รอให้ความเร็วของ Motor ถึงจุดที่วิ่งด้วยความเร็วสูงสุดเมื่อ Invertor ได้รับ 5V จาก Computer แล้วมีจำนวน step ~ 12000

<u>ส่วนที่ 3</u> ส่วนนี้จะเป็นส่วนของระยะทางที่เราจะกำหนดให้ motor เคลื่อนที่ไปจนถึงตำแหน่งที่เรากำหนด มีจำนวน step > 12000 ขึ้นไป

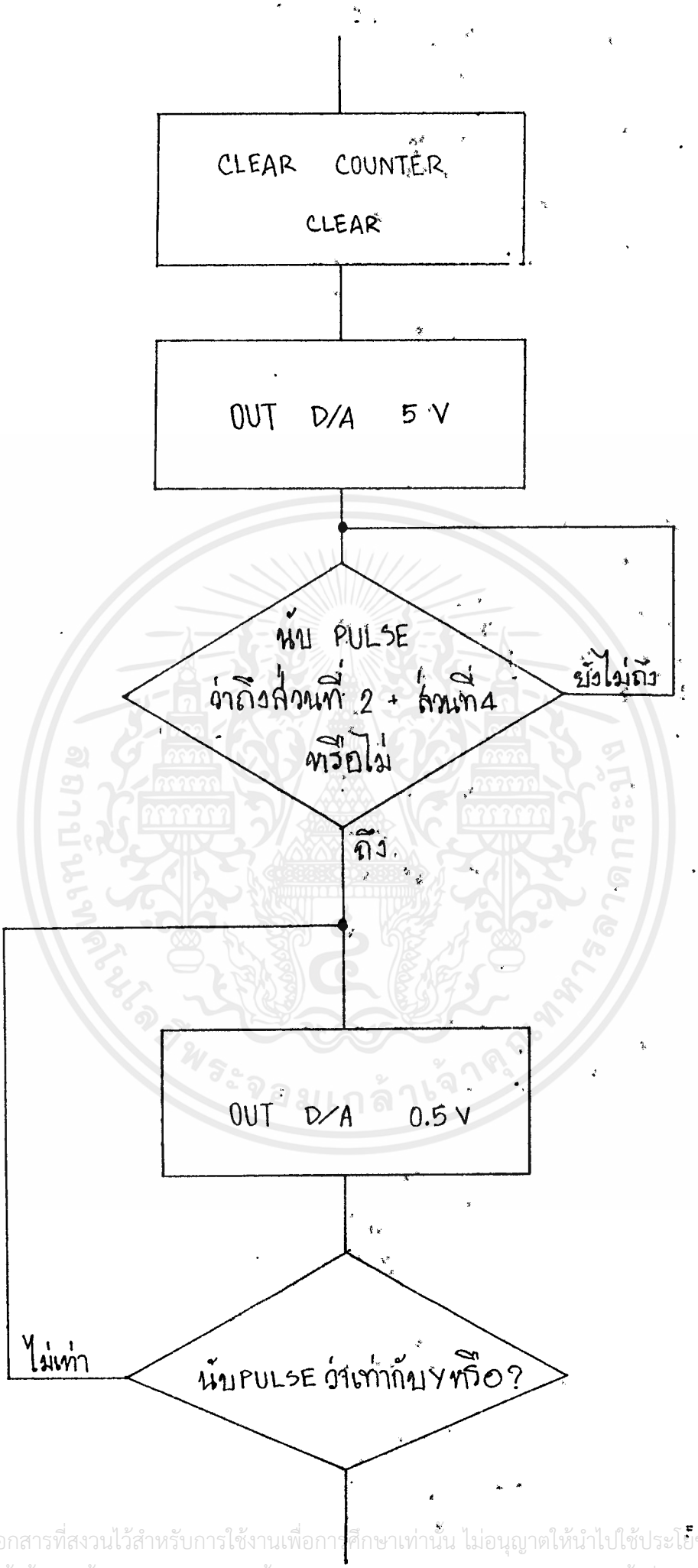
<u>ส่วนที่ 4</u> การทำงานในช่วงนี้เป็นการทำงานในช่วงหยุดของ motor เมื่อก่อนถึงระยะทาง 6000 pluse การลดความเร็วของมอเตอร์ ลงโดยการลดค่า D/A ที่ป้อนให้กับ Inverter ลงจาก 5V ($FF) ลงเหลือ 0.5V $19 (ลดลงทีละ $01) เพื่อ control การหยุดของ motor ให้มี error น้อยที่สุด แบะจาก 0.5V $19 มาเป็น 0V ($00) เพื่อหยุดการหมุนของ Motor

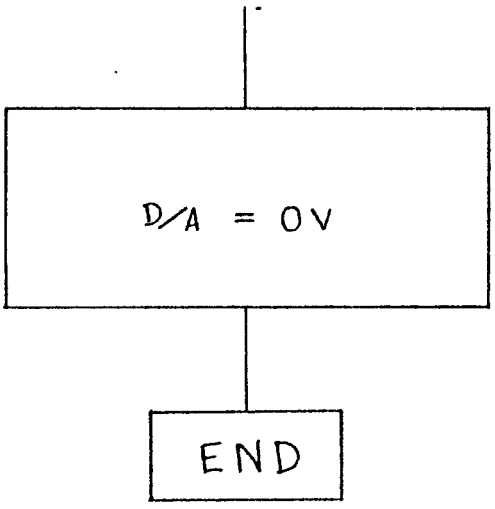
ในการควบคุม Motor ที่มีระยะทางน้อยๆ จะใช้การควบคุมส่วนนี้

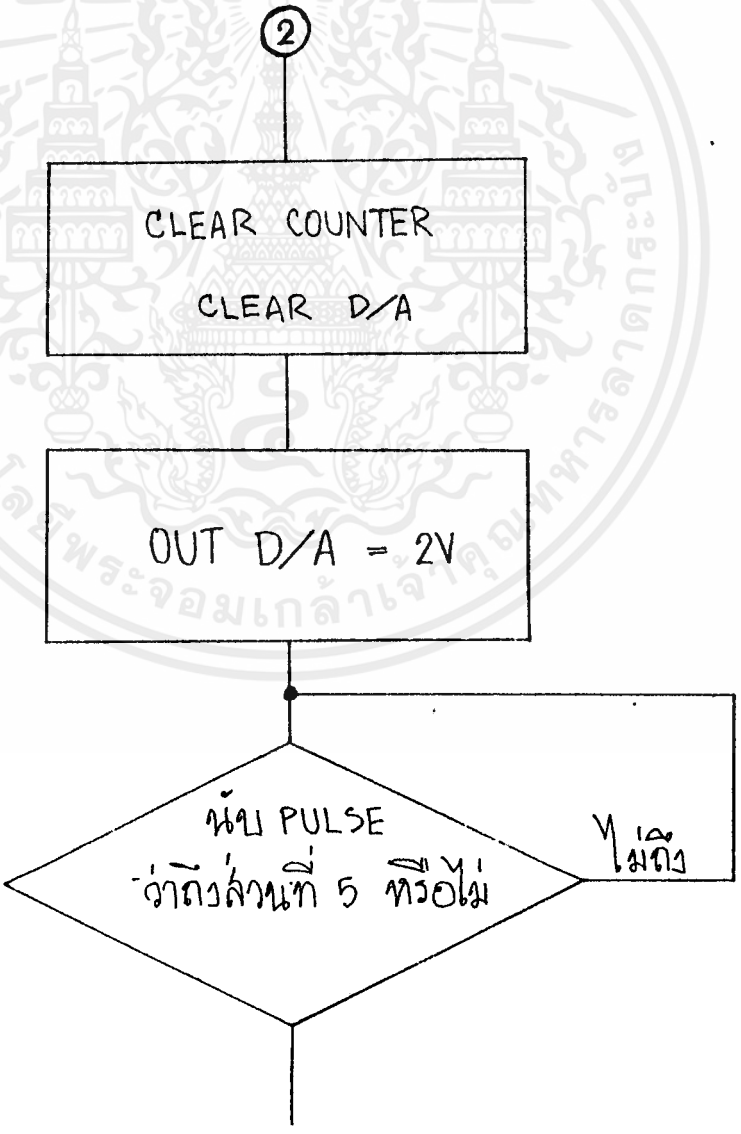
รูป 16 แสดงความเร็วของ MOTOR ที่ 2V.

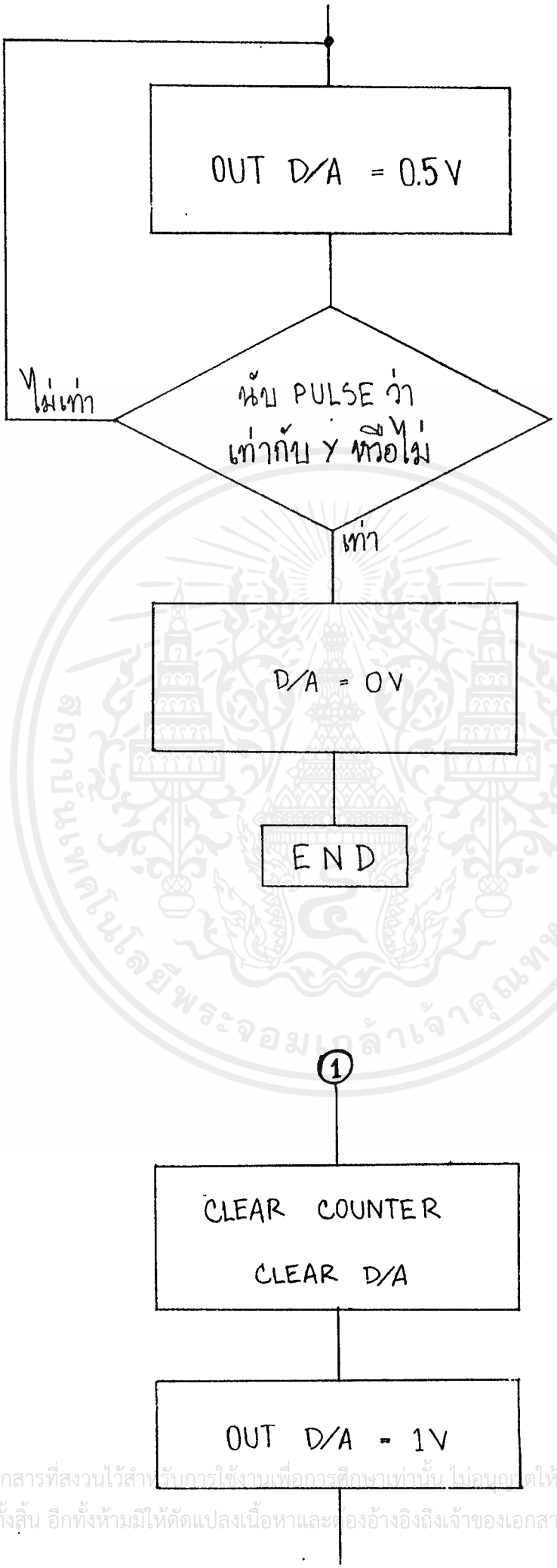


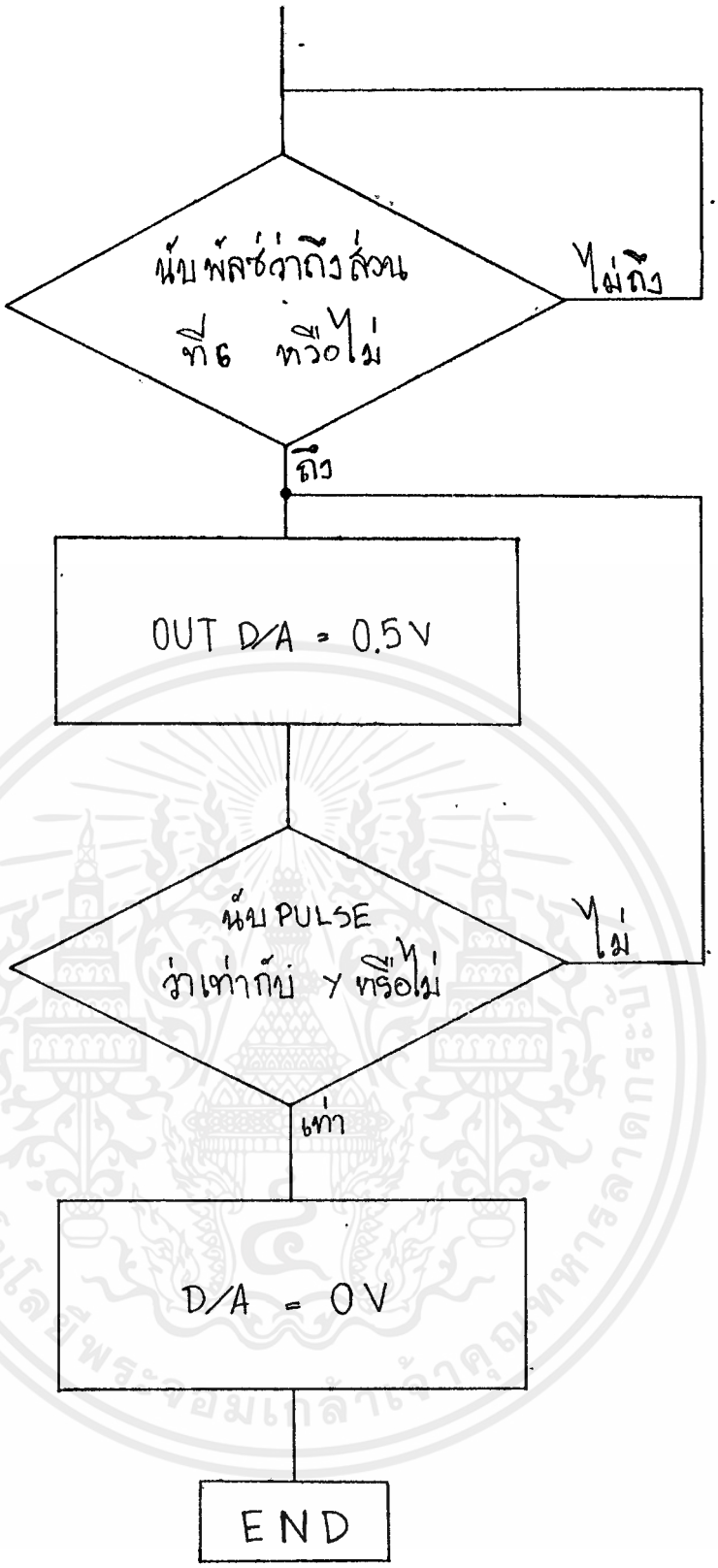
รูป 17 แสดงความเร็วของ MOTOR ที่ 1 V.











การพิจารณาว่าจะ control การหมุนของ Motor อย่างไรนั้นจะดูได้จากเมื่อรับค่าจากไฟล์ระยะทางมาคำนวณหาได้ค่า step แล้ว จะพิจารณาที่จำนวน step ว่าจะเข้าสู่การควบคุม Motor อย่างไร

ถ้าจำนวน step มากกว่า 18000 (คือ 12000 + 6000) จะควบคุม motor ตามรูปที่ 15

ถ้าจำนวน step อยู่ในช่วง 8000-18000 จะควบคุม Motor ตามรูปที่ 16

ถ้าจำนวน step น้อยกว่า 8000 จะควบคุม Motor ตามรูปที่ 17 และมี Flow Chart ในการเขียนโปรแกรมดังนี้

<u>การควบคุมส่วนต่าง ๆ ของเครื่อง</u>

เนื่องจาก Turbo Pascal มีข้อจำกัดทางภาษา คือไม่มีคำสั่งที่ใช้ set bit ได้ ดังนั้นการควบคุม Output Port 1 และ 2 จึงทำได้ยาก ซึ่ง Output แต่ละ bit จะควบคุมหน้าที่ต่างๆ ของเครื่องพับเหล็กเช่น Emergency Switch Start PB เป็นต้น เพื่อเป็นการแก้ปัญหานี้จึงกำหนดตัวแปร Tempsw ขึ้นมาเพื่อเก็บสถานะต่าง ๆ ของแต่ละ bit เพื่อใช้ในการควบคุมต่อไป

Port Output Relay 1 อยู่ที่ตำแหน่ง $0280

Port Output Relay 2 อยู่ที่ตำแหน่ง $0283

รายละเอียดของแต่ละบิทเป็นดังนี้

Port Output 1 $0280

- bit 0 Emergency Switch
- bit 1 Start Switch
- bit 2 Select Auto or Manual
- bit 3 Reverse Motor
- bit 4 Forward Motor
- bit 5 ON/OFF Foot Switch
- bit 6 Computer Control
- bit 7 Select Motor 2 or Motor 3

Port Output 2 $0283

bit 0 - bit 3 ไม่ได้ใช้

bit 4 Inverter 0.5V.

bit 5 Inverter 1V.

bit 6 Inverter 2V.

bit 7 Inverter 5V.

ควบคุมการปิดเครื่อง

เมื่อจะปิดเครื่องพับเหล็กจากโปรแกรม จะ set Output 1 bit ที่ 6 เป็น "1" บิทเดียว และจะมีการเก็บค่าสภาวะ switch ไว้เสมอที่ Tempsw

```
Procedure Stopm;
Begin
    {stop = $40}
    Port[Pout1] := stop;
    tempsw := stop;
    Delay(10);
End;
```

ควบคุมการเปิดเครื่อง

เมื่อจะเปิดเครื่องพับเหล็กจากโปรแกรม จะ set Output Port 1 bit ที่ 0,1 2 บิท

```
Procedure Startm;
Begin
    {Start = $43}
    {stop  = $40}
```

```
        if ((tempsw and start) = start) then
        begin
            Port[Pout1] := stop;
            tempsw := stop;
            Delay(10);
        end
        else
        begin
            Port[Pout1] := (tempsw or Start);
            delay(300);
            tempsw := (tempsw or start);
        end;
    End;
```

ควบคุมการ ON/OFF Foot Switch

เมื่อจะ ON หรือ OFF มาเหยียบ จะ set Outpot Port 1 bit ที่ 5 นิทเดียว

```
    Procedure Footswm;
    Begin
        {Footswitch = $20}
        if ((tempsw and footswitch) = footswitch) then
        begin
            Port[pout1] := (tempsw and (not footswitch));
            delay(10);
            tempsw := (tempsw and (not footswitch));
        end
```

```
            else
            begin
                Port[pout1] := (tempsw or footswitch);
                delay(10);
                tempsw := (tempsw or footswitch);
            end;
        end;
```

ควบคุมการเลือก Auto หรือ Manual

เมื่อจะเลือก Auto หรือ Manual ของการกดลงของแท่นพับ จะ set Output Port 1 bit ที่ 2 บิทเดียว

```
        Procedure Autom;
        Begin
            {Autoswitch = $04}
            {if ((port[pin2] and $02) = $02) then}
            if ((tempsw and autoswitch) = autoswitch) then
            begin
                Port[pout1] := (tempsw and (not autoswitch));
                delay(10);
                tempsw := (tempsw and (not autoswitch));
            end
            else
            begin
                Port[pout1] := (tempsw or autoswitch);
                delay(10);
```

```
                tempsw := (tempsw or autoswitch);
            end;
        end;
```

ควบคุมการเคลื่อนที่ของ Motor 3

จะเป็นการปรับมุมพับของแท่นพับ จะ set Output Port 1 bit ที่ 3 หรือ 4 (Forward หรือ Reverse) และ Output Port 2 bit ที่ 4 หรือ 5 หรือ 6 หรือ 7 (เลือกความเร็วของ Motor) เพียง 2 บิท

ควบคุมการเคลื่อนที่ของ Motor 2

จะเป็นการปรับระยะแากกั้นหลังของเครื่องพับเหล็ก จะ set Output Port 1 bit ที่ 3 หรือ 4 bit ที่ 7 และ Port Output 2 bit ที่ 4 หรือ 5 หรือ 6 หรือ 7 เพียง 2 บิท

Port Input 1 อยู่ที่ตำแหน่ง $2280

Port Input 2 อยู่ที่ตำแหน่ง $2283

รายละเอียดของแต่ละบิทเป็นดังนี้

Port Input 1

bit 0 ไฟเข้าเครื่อง

bit 1 Start Switch

bit 2 Control Reverse

bit 3 Limit Switch ฉากกั้นหลัง

bit 4 Control Forward

bit 5 Limit Switch ฉากกั้นหน้า

bit 6 Limit Switch แท่นพับยกขึ้น

bit 7 แท่นพับกดลง

Port Input 2

bit 0 แท่นพับยกขึ้น

bit 1 Select Switch Auto or Manual

bit 2 Limit Switch แท่นพับกดลง

bit 3 - bit 7 ไม่ได้ใช้

การรับค่าของการกดแท่นพับ

การรับค่าในส่วนนี้จะรอจนกว่าจะมีการเหยียบ Foot Switch ลง-ขึ้น และแท่นพับยกขึ้นสุดแล้ว

```
repeat
    tempin := port[pin1];
until ((tempin and $80) = $80);
repeat
    tempin := port[pin2];
until ((tempin and $01) = $01);
repeat
    tempin := port[pin1];
until ((tempin and $40) = $40);
```

การรับค่าของฉากกั้นหน้า

เมื่อสั่งให้ Motor 2 หมุนเดินหน้าจนมาชน Limit Switch กั้นหน้าแล้ว แสดงว่าฉากกั้นเลื่อนมาจนสุดแล้ว

```
Repeat
        ------
        ------
```

```
Until ((Port[Pin1] And $20) = $00);
```

การรับค่าของฉากกั้นหลัง

เมื่อสั่งให้ Motor 2 หมุนถอยหลังจนไปชน Limit Switch กั้นหลังแล้ว แสดงว่าฉากกั้นเลื่อนไปจนสุดแล้ว

```
Repeat
        ------
        ------
Until ((Port[Pin1] And $04) = $00);
```

บทที่ 7.

การใช้งานโปรแกรม

การเปิดเครื่อง

เริ่มแรกต้องเปิด Breaker ของเครื่องพับเหล็ก แล้วจึงเปิดไปเข้าเครื่อง เมื่อไฟเข้าเครื่องพับ คอมพิวเตอร์จะเริ่มทำงาน โดยทำการ Boot Dos ซึ่งเป็นโปรแกรมจัดระบบ จากนั้นจะเข้าสู่โปรแกรมที่จะใช้สำหรับควบคุมเครื่องพับเหล็ก

การใช้งานโปรแกรม

เมื่อคอมพิวเตอร์เปิด เครื่องจะ Boot DOS และ load โปรแกรมที่จะใช้ควบคุมเครื่องพับเหล็ก จอภาพจะแสดงส่วนของ Demo Program ให้กดคีย์ใด ๆ ผ่านไป หลังจากนั้นจะเข้าสู่ Menu หลัก ซึ่งจะมีทั้งหมด 7 Menu ดังนี้

1. Help Menu F1
2. Edit Menu F2
3. Run Menu F3
4. Step Menu F4
5. Manual Menu F5
6. Option Menu F6
7. Position Menu F7

--> การใช้งาน Help Menu F1

เมื่อกด key F1 ที่โปรแกรมหลักจะปรากฏ Pull Down Menu ซึ่งมีคำสั่งให้เลือก 2 คำสั่ง คือ Help Menu F1 และ Esc F10

ถ้ากด key F1 จะเข้าสู่ Help Menu ซึ่งจะแสดงรายละเอียดการใช้งานคร่าว ๆ ของแต่ละ Menu ที่ Menu หลัก และใน Help Menu จะมี Menu อีก 1 Menu คือ Exit F9 เมื่อกด F9 แล้วจะมี Pull Down Menu ย่อย เปิดขึ้นมาอีก ใน Menu

ย่อยจะมี 2 คำสั่งคือ Exit F9 และ Esc F10 ถ้ากด key F9 จะเป็นการออกจาก Help Menu กลับเข้าสู่ Menu หลัก

* สำหรับ key Esc F10 ของทุก Menu จะหมายถึงการปิด Pull Down Menu เท่านั้น จะไม่มีผลต่อคำสั่งใด ๆ ใน Menu นั้น ๆ

--> การใช้งาน Edit Menu F2

เมื่อกด key F2 ที่โปรแกรมหลัก จะมี Pull Down Menu ย่อยเปิดขึ้นมา ใน Menu ย่อยจะมี 3 คำสั่ง คือ Load F2 , New F3 และ Esc F10
ถ้ากด key F1 จะเข้าสู่ Load Menu ซึ่งจะให้ป้อนชื่อ file ที่ต้องการจะ load ก่อน การป้อนค่าในส่วนนี้ถ้าป้อนผิดรูปแบบเครื่องจะไม่รับ หรือให้ป้อนข้อมูลใหม่ หลังจากป้อนชื่อ file แล้ว ถ้าไม่มี file นี้อยู่ในแฟ้มข้อมูลก็จะกลับไปที่ Menu หลัก แต่ถ้ามีเครื่องก็จะแสดงข้อมูลใน file นั้นออกมา ซึ่งข้อมูลใน file จะเป็นจำนวน step และค่าระยะทาง (ชื่อ file จะป้อนเป็นตัวเลข) และใน Load Menu จะมี Menu อีก 4 Menu คือ Edit F1 , Insert F2 , Delete F3 และ Exit F9 ถ้ากด key F1 (Edit) จะเป็นการแก้ไขข้อมูลเดิมตามจำนวน step ที่ป้อน ถ้ากด key F2 (Insert) จะเป็นการเพิ่มหรือแทรกข้อมูลตามจำนวน step ที่ป้อน ถ้ากด key F3 (Delete) จะเป็นการลบข้อมูลตามจำนวน step ที่ป้อน ถ้ากด key F9 (Exit) จะเป็นการออกจาก Load Menu กลับไปสู่ Menu หลัก โดยจะทำการ save file ที่แก้ไขให้ด้วย นอกจากนี้เครื่องจะทำการตรวจสอบการรับจำนวน step และข้อมูลที่แก้ไขด้วย ถ้าป้อนค่าผิดรูปแบบ เครื่องจะไม่รับค่าหรือสั่งให้ป้อนค่าใหม่

ถ้ากด key F2 จะเข้าสู่ Newfile Menu ซึ่งจะให้ป้อนชื่อ file ที่ต้องการจะสร้างขึ้นใหม่ หลังจากป้อนชื่อ file แล้วจะให้ทำการป้อนค่าข้อมูลใหม่ โดยเริ่มจาก step ที่ 1 จนถึง step ที่ 17 หรือขณะที่ป้อนข้อมูล ถ้าพอแล้วก็ให้ใส่ค่า 0 หรือกด Enter ใน step ถัดไปก็จะเป็นการสิ้นสุดการป้อน ทำให้ไม่ต้องป้อนค่าจนครบ 17 step เมื่อป้อนข้อมูลเสร็จแล้วจะแสดง Newfile Menu และใน Newfile Menu จะมี Menu อีก 4 menu เหมือนใน Load Menu และมีการทำงานต่อจากนี้เหมือนกัน

--> การใช้งาน Run Menu F3

เมื่อกด key F3 ที่โปรแกรมหลัก จะมี Pull Down Menu ย่อยเปิดขึ้น ประกอบด้วย 2 คำสั่งคือ Load F3 และ Esc F10

ถ้ากด key F3 จะเข้าสู่ Run Menu ซึ่งจะให้ป้อนชื่อ file ที่ต้องการจะทำการ load ก่อน หลังจากป้อนชื่อ file แล้วถ้าไม่มี file นั้นปรากฏอยู่ในแฟ้มข้อมูล ก็จะกลับไปที่ Menu หลัก แต่ถ้ามี file นั้นปรากฏอยู่ เครื่องก็จะแสดงข้อมูลใน file นั้นออกมา และใน Run Menu จะมี Menu อีก 3 menu คือ Run F1 , Option F2 และ Esc F10 ถ้ากด key F1 (Run) จะเป็นการเลื่อนฉากกั้นหลังของเครื่องพับไปยังตำแหน่งที่ตั้งไว้ และถ้าเครื่องพับเหล็กยังไม่ได้เปิดเครื่อง โปรแกรมจะทำการเปิดเครื่องให้ และถ้ายังไม่ได้ set ฉากกั้นหลังกลับมาที่จุดเริ่มต้น โปรแกรมก็จะ set มาที่จุดเริ่มต้นให้ โดยที่การ Run จะเริ่มที่ step แรกของ file ไปจนถึง step สุดท้าย และเมื่อ step แรกถูกเลื่อนไปจนถึงค่าที่ตั้งไว้แล้ว โปรแกรมจะรอจนกว่าจะมีการกดแท่นพับเหล็กลงมาพับเหล็ก 1 ครั้ง แล้วจึงเลื่อนฉากกั้นหลังไปยังตำแหน่งต่อไป ถ้ากด key F2 (Option) จะมี Menu ย่อยของคำสั่งออกมาอีก 4 คำสั่งคือ Start Machine F1 , ON/OFF Foot Switch F2 , Select Auto F3 , Origin F4 และ Esc F5 คำสั่งเหล่านี้จะเหมือนกับ Option Menu ที่ Menu หลัก

--> การใช้งาน Step Menu F4

เมื่อกด key F3 ที่โปรแกรมหลัก จะมี Pull Down Menu ย่อยปรากฏขึ้น ใน Menu ย่อยจะมี 2 คำสั่งคือ Load F4 และ Esc F10

ถ้ากด key F3 จะมีการทำงานเหมือนกับ Run Menu แต่จะต่างกันตรงที่ Step Menu มี Menu ย่อยมากกว่า 1 menu คือ Select Menu ถ้าเลือกคำสั่งนี้จะเป็นการเลือก step ในการเลื่อนฉากกั้นหลัง คำสั่งนอกนั้นจะเหมือนกับใน Run Menu

--> การใช้งาน Manual Menu F5

ในส่วนของ Menu นี้ยังไม่มีการใช้งานใด ๆ

--> การใช้งาน Option Menu F4

เมื่อกด key F6 ที่โปรแกรมหลัก จะมี Pull Down Menu ย่อยปรากฏขึ้น ใน

Menu ย่อยจะมี 5 คำสั่ง คือ Start Machine F1 , ON/OFF Foot Switch F2 , Select Auto F3 , Origin F4 และ Esc F5 ทุกคำสั่งยกเว้น Esc F5 จะแสดงสถานะของเครื่องที่เป็นอยู่ในขณะนั้น (ON หรือ OFF)

ถ้ากด key F1 จะสังเกตเห็นเครื่องหมายกากบาทขึ้นอยู่หน้าสถานะ และถ้ากดอีกครั้งเครื่องหมายกากบาทจะหายไป ถ้ามีเครื่องหมายกากบาทปรากฏหน้าสถานะใด ถือว่าให้ทำตรงข้ามกับสถานะนั้นๆ ดังนั้นถ้ามีเครื่องหมายกากบาทปรากฏอยู่หน้าสถานะใดสถานะหนึ่ง แสดงว่าโปรแกรมจะทำการ start หรือ stop เครื่องพับเหล็ก (หรือทำการ ON หรือ OFF เครื่องพับเหล็กนั่นเอง)

ถ้ากด key F2 และมีเครื่องหมายกากบาทปรากฏอยู่หน้าสถานะนั้น โปรแกรมจะทำการ ON หรือ OFF Foot Switch

ถ้ากด key F3 และมีเครื่องหมายกากบาทปรากฏอยู่หน้าสถานะนั้น โปรแกรมจะทำการ ON หรือ OFF Auto Switch

ถ้ากด key F4 และมีเครื่องหมายกากบาทปรากฏอยู่หน้าสถานะนั้น โปรแกรมจะทำการ set ฉากกั้นหลังไปที่จุดเริ่มต้น

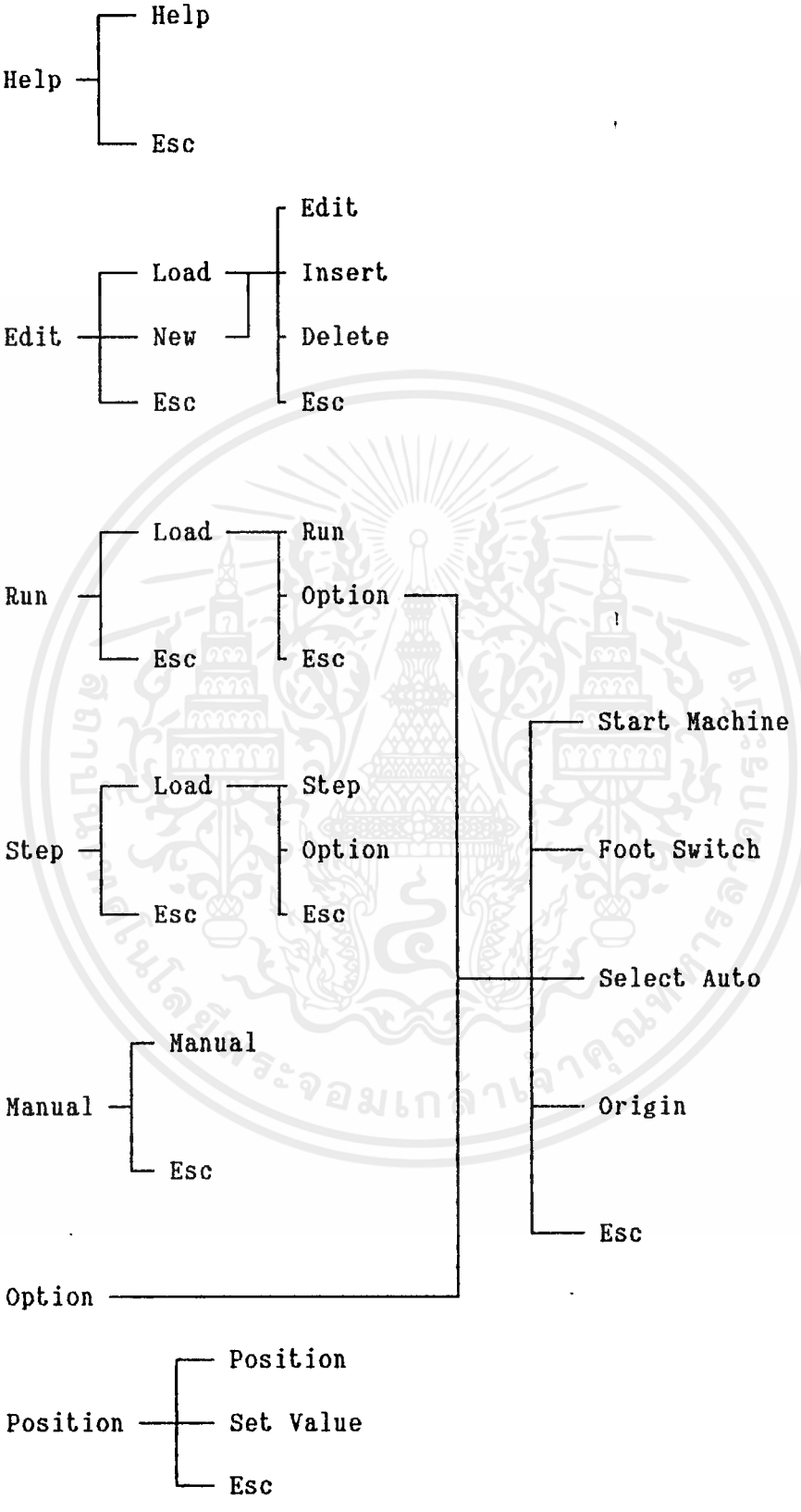
--> การใช้งาน Position Menu F7

เมื่อกด key F7 ที่โปรแกรมหลัก จะมี Pull Down Menu ย่อยปรากฏขึ้น ใน Menu ย่อยจะมี 3 คำสั่ง คือ Position F7 , Set Value F8 และ Esc F9

ถ้ากด key F7 จะเข้าสู่ Position Menu ซึ่งจะแสดงตำแหน่งของฉากกั้นหลัง และใน Position Menu จะมี Menu อีก 1 menu คือ Exit F9 เมื่อกด F9 แล้วจะมี Pull Down Menu ย่อยเปิดขึ้นมาอีก ใน Menu ย่อยนี้จะมี 2 คำสั่งคือ Exit F9 และ Esc F10 ถ้ากด F9 จะกลับเข้าสู่ Menu หลัก

ถ้ากด key F8 จะเข้าสู่ Set Value Menu ซึ่งจะเป็นการ set ค่าฉากกั้นหน้ากับฉากกั้นหลังหากมีการเปลี่ยนแปลง

ซึ่ง Key ทั้งหมดนี้สามารถเขียนเป็นโครงสร้างได้ดังนี้



โปรแกรม PULL DOWN MENU

เพื่อความสะดวกในการใช้โปรแกรม จึงเลือกการทำงานของ Menu แบบ Pull Down Menu ซึ่งเป็น Menu ซึ่งใช้งานง่ายและรวดเร็ว

การทำงานของ Pull Down Menu มีสองสถานะคือเปิด หรือปิด เมื่อ Menu ปิด คือไม่มีการเรียกใช้ Menu ก็จะปรากฏเฉพาะส่วนหัวของ Menu หลัก ที่บรรทัดบนสุดของจอภาพเท่านั้น เมื่อ Menu เปิด คือมีการเรียกใช้ Menu ก็จะปรากฏหน้าต่างขึ้นอีกหน้าต่างหนึ่งซึ่งมีคำสั่งอยู่ภายในนั้น เมื่อเปิด Menu แล้ว เครื่องจะรอรับสัญญาณการกดคีย์จากผู้ใช้โปรแกรมเพื่อเลือกคำสั่งการทำงาน และทำตามคำสั่งนั้น หลังจากนั้น Menu จะปิดลง จอภาพจะปรากฏให้เห็นเพียงส่วนหัวของ Menu หลัก เพื่อรอการเปิด Menu จากผู้ใช้โปรแกรมอีกตามความต้องการ

ส่วนของโปรแกรม Pull Down Menu จะแบ่งออกเป็น 4 ส่วน ได้แก่

1.ส่วนกำหนดตัวแปร

2.ส่วนกำหนดค่าเริ่มต้น

3.ส่วนของ Menu ย่อย

4.ส่วนของการเขียนหัว Menu

5.ส่วนของการเปิด-ปิด Menu ย่อย

1. ส่วนของการกำหนดตัวแปร

กำหนดให้ Menu Item เป็น Record และ Submenu เป็น Object เพื่อใช้ในการสืบสกุลลูกได้อีกหลายสกุล

และ Proc กำหนดเป็น Function ของ Booiean เพื่อใช้เป็นตัวแสดงว่าคำสั่งใดถูก Process

```
Type
     Proc = Function : Boolean;
MenuItem = Record
          Text : String[80];
          Attr : Byte;
```

```
                DoThis : Proc;
                Key : Word;
                End;
        SubMenu = Object
                Title : String[80];
                List : Array[1..25] Of MenuItem;
                Count : Integer;
                Left : Integer;
                Width : Integer;
                Attr : Byte;
                CursorAttr : Byte;
                Key : Word;
Procedure Init(Ttle : String; Ky : word; Lft,Wid : Integer
                ;At1,At2 : Byte);
Procedure AddItem(Text : string; Ky : Word; At : Byte
                ;DoIt : Proc);
Procedure DrawTitle;
Procedure Open(Ky : Word);
```

2. ส่วนของการกำหนดค่าเริ่มต้น

จะกำหนดค่าเริ่มต้นสำหรับ Pull Down Menu

```
Procedure SubMenu.Init(Ttle : String; Ky : word; Lft,
          Wid : Integer;   At1,At2 : Byte);
Begin
     Title      := Ttle;
     Count      := 0;
     Left       := Lft;
     Width      := Wid;
     Attr       := At1;
     CursorAttr := At2;
     Key        := Ky;
End;
```

3. ส่วนของ Menu ย่อย

ส่วนนี้จะเป็นการกำหนด Menu ย่อยที่อยู่ใน Menu หลัก โดยมี

- ชื่อคำสั่ง
- คีย์ที่จะ Process
- ตัวแปร Boolean ที่จะชี้ว่าคำสั่งใดถูกเลือก

```
Procedure SubMenu.AddItem(Text : string; Ky : Word; At : Byte
                          ;DoIt : Proc);
Begin
     Inc(Count);
     List[Count].Text   := Text;
```

```
        List[Count].Attr   := At;
        List[Count].DoThis := DoIt;
        List[Count].Key    := Ky;
    End;
```

4. ส่วนของการเขียนหัว Menu

ในส่วนนี้จะเขียนส่วนหัวของ Pull Down Menu ที่บรรทัดบนสุดของจอภาพ และความกว้างของแต่ละ Menu ขึ้นอยู่กับการกำหนดในตอนแรก

```
    Procedure Submenu.DrawTitle;
    Begin
        window(1,1,80,25);
        TextAttr := Attr;
        GotoXY(Left + (Width - Length(Title)) Div 2,1);
        Write(Title);
    End;
```

5. ส่วนของการเปิด-ปิด Menu ย่อย

ส่วนนี้จะคอยรับค่าคีย์เพื่อมาเปิด Menu ย่อย และเมื่อมีการกดคีย์ที่ตรงกับ Menu ย่อย เครื่องจะทำการเปิด Menu ย่อย ใน Menu ย่อยจะมีชื่อคำสั่งปรากฏ (มีแถบสว่างที่ชื่อคำสั่งแรก) และรับคีย์เพื่อจะไปทำคำสั่งนั้น โดยมีทางเข้าถึงคำสั่งย่อยได้ 2 วิธี คือ กดคีย์ที่เป็นคีย์เฉพาะของชื่อคำสั่งนั้น หรือเลื่อนแถบสว่างไปที่คำสั่งที่ต้องการแล้วกด Enter ก็ได้

```
    Function NewCur(Cur,Change,Max : Integer) : Integer;
    Var
        i : Integer;
    Begin
```

```
        i := Cur + Change;
        If i > Max Then
            i := Max
        Else If i < 1 Then
            i := 1;
        NewCur := i;
    End;

    procedure Submenu.open(ky : word);
    var
        i,j : integer;
        ch : char;
        Userkey : word;
        Done : boolean;
        curitem : integer;
    begin
        if ky = key then
        begin
            setboxstyle(single);
            windowopen(left,2,left+width+4,count+4);
            textattr := attr;
            for i := 1 to count do
            begin
                gotoxy(1,i);
                write(list[i].text);
            end;
            curitem := 1;
            done := false;
```

```
            while Not done do
            begin
                textattr := cursorattr;
                gotoxy(1,curitem);
                write(list[curitem].text);
                ch := readkey;
                if ord(ch) = 0 then
                userkey := ord(readkey) * 256
                else
                userkey := ord(ch);
                TextAttr := Attr;
                GotoXY(1,curItem);
                Write(List[CurItem].Text);
                Case UserKey of
                27    : Done := True;
                20480 : CurItem := NewCur(CurItem,1,Count);
                18432 : CurItem := NewCur(CurItem,-1,Count);
                13    : Done := List[CurItem].DoThis;
                Else
                for i := 1 to Count do
                if UserKey = List[i].Key Then
                Done := List[i].Dothis;
            end;
        end;
        windowclose;
    end;
end;
```

Function Newcur ใช้เพื่อหาตำแหน่งของแถบสว่างว่าอยู่ที่ชื่อคำสั่งใด

บทที่ 8

สรุปโครงงาน

โครงงานนี้เป็นการออกแบบ และ พัฒนาระบบ ทั้งทางด้าน ฮาร์ดแวร์ และ ซอฟแวร์ เพื่อช่วยในการควบคุมเชิงตัวเลขด้วยคอมพิวเตอร์ (Computerize Numerical Control) และนำไปประยุกต์ใช้งานอุตสาหกรรม

ในส่วนของงานเทอมนี้จะนำไปใช้กับเครื่องพับเหล็ก โดยใช้หลักการของ CNC ที่มีแกนเดียว (Z Axiz)

ปัญหาที่เกิดขึ้น

- ในการหาระยะทางการเคลื่อนที่ของมอเตอร์(กรณี ForWard และ Revese) ให้มอเตอร์หยุด ณ. ตำแหน่งระยะทางที่โปรแกรมไว้ จะเกิด Error ของระยะทาง เนื่องมาจากค่าทอร์กของมอเตอร์

- ไม่ทราบรายละเอียดของขนาดเฟืองต่างๆ ซึ่งไม่สามารถนำไปคำณวณหาทอร์กได้ จึงใช้วิถีการสั่งให้ทำงานหลาย ๆ ครั้งแล้วหาค่าเฉลี่ยแต่ละช่างการเคลื่อนที่ของมอเตอร์

- การเชื่อมต่อระหว่าง Digital กับ Analog เกิดการกวนกันได้ง่าย

- การเคลื่อนที่ของฉากกั้นหลัง ถ้าเคลื่อนที่ในระยะทางใกล้ๆ จะเกดความผิดผลาดมากกว่า การเคลื่อนที่ในระยะทางไกลๆ เช่น ให้กั้นหลังเคลื่อนที่ไป 1 cm ช่วงระยะทางที่น้อยที่สุดที่เคลื่อนไปแล้วเกิดค่าผิดพลาดยอมรับได้คือ ประมาณ 3 cm

```
Program CNC;
Uses Crt,Dos,Win,keyboard,screen,variable,pullmenu,
     helpmenu,loadmenu,newfmenu,posimenu,runmenu,stepmenu;

Var
        menu1,menu2,menu3               : submenu;
        menu4,menu7,menu5               : submenu;
        menu6                           : submenuo;
        Helpb,Escb,Loadb,newffb         : Boolean;
        Runmb,Stepb,Manualb,quitb       : Boolean;
        Posib,Done,escinb               : Boolean;
        key                             : Word;
        ch                              : char;
        i                               : integer;

{$f+}

Function Helpf : Boolean;
begin
        helpb := true;
        helpf := true;
end;

Function Escf : Boolean;
begin
        Escb := true;
        Escf := true;
end;

Function Escin : Boolean;
begin
        Escinb := true;
        Escin  := true;
end;

Function Loadf : Boolean;
begin
        Loadb := true;
        Loadf := true;
end;

Function Newff : Boolean;
begin
        Newffb := true;
        Newff  := true;
end;

Function Runmf : Boolean;
begin
        Runmb := true;
        Runmf := true;
end;

Function Stepf : Boolean;
begin
        Stepb := true;
        Stepf := true;
end;
```

```
Function Manualf : Boolean;
begin
        Manualb := true;
        Manualf := true;
end;

Function Posif : Boolean;
begin
        Posib := true;
        Posif := true;
end;

Function Setvaluef : Boolean;
begin
        setvalueb := true;
        setvaluef := true;
end;

Function Quitf : Boolean;
begin
        quitb := true;
        quitf := true;
end;

procedure Title;
begin
        clrscr;
        Fillwin(1,1,80,25,177,15);
        windowopen(17,4,63,8);
        directvideoon;
        writeln('     P R O G R A M  C O M P U T E R');
        writeln('                F O R');
        write ('  C O N T R O L          M A C H I N E');
        directvideooff;
        delay(500);
        setboxstyle(mix1);
        windowopen(13,11,66,23);
        directvideoon;
        writeln('             D E S I G N   B Y  ');
        writeln;
        writeln('          1. JUKKRIT LIMTHEERAKUL');
        writeln('          2. CHAIMAIPRON VONGMUNG');
        writeln('          3. VISUT CHONGPRASERTSAK');
        writeln('          4. SAKSAN VAJANAKUTNAKRON');
        writeln;
        writeln('  FACULTY OF ENGINEERING : INSTRUMENT COMPUTER ');
        writeln('             KINGMONKUT LADKBANG ');
        writeln;
        delay(1000);
        textcolor(white+blinkhigh);
        write('            HIT ');
        textcolor(white);
        write(' ANT KEY TO CONTINUES');
        directvideooff;
        readfunctionkey(key);
        windowclose;
        windowclose;
        setboxstyle(double);
```

```
        windowOpen(8,4,72,21);
        directvideoon;
        writeln('                THIS IS A MACHINE COLTROL ');
        writeln('                          PROGRAM        ');
        writeln;
        writeln('       The program is useD to control distance of Z axis');
        writeln('   machine moving and any option(s) on the manul mode or ');
        writeln('   auto mode.  ');
        writeln('       The program will easily help you to control the');
        writeln('   machine. You can ask for some information about parameters.');
        writeln;
        writeln;
        writeln('   This is senior project of the Department of INDUSTRY ');
        writeln('           COMPUTER KINGMUNKUT LADKBANG ,1992-1993');
        writeln;
        writeln('           Adviser     :       PAKRON HUTASANGKA');
        writeln;
        delay(1000);
        textcolor(white + blinkhigh);
        write('                    HIT ');
        textcolor(white);
        write(' ANY KEY TO CONTINUES');
        directvideooff;
        readfunctionkey(key);
        windowclose;
end;

Procedure printmainmenu;
begin
     menu1.drawtitle;
     menu2.drawtitle;
     menu3.drawtitle;
     menu4.drawtitle;
     menu5.drawtitle;
     menu6.drawtitle;
     menu7.drawtitle;
end;

{$f-}

Begin
     Cursoroff;
     Set8255;
     Port[pout1] := Nor;
     delay(10);
     Port[pout2] := Nor;
     delay(10);
     Tempsw  := 0;
     Tempdis := 145.30;
     select  := 1;
     for i := 1 to maxline do
         tempfile[i] := 0;
     assign(nfi,'set.ovl');
     reset(nfi);
     read(nfi,originpoint);
     read(nfi,lastpoint);
     close(nfi);
     Title;
```

```
menu1.init(' HELP  F1 ',15104,1,10,$6F,$24);      {64}
menu1.additem(' HELP     F1 ',15104,$6F,helpf);
menu1.additem(' ESC     F10 ',17408,$6F,escf);
menu2.init(' EDIT  F2 ',15360,11,10,$67,$24);     {67}
menu2.additem(' LOAD     F2 ',15360,$67,loadf);
menu2.additem(' NEW      F3 ',15616,$67,newff);
menu2.additem(' ESC     F10 ',17408,$67,escf);
menu3.init(' RUN   F3 ',15616,21,10,$6F,$24);
menu3.additem(' LOAD     F3 ',15616,$6F,runaf);
menu3.additem(' ESC     F10 ',17408,$6F,escf);
menu4.init(' SINGLE F4 ',15872,31,11,$67,$24);
menu4.additem(' LOAD      F4 ',15872,$67,stepf);
menu4.additem(' ESC      F10 ',17408,$67,escf);
menu5.init(' MANUAL F5 ',16128,42,11,$6F,$24);
menu5.additem(' MANUAL    F5 ',16128,$6F,manualf);
menu5.additem(' ESC      F10 ',17408,$6F,escf);
menu6.init(' OPTION F6 ',16384,53,11,$67,$24);
menu6.additem(' START M    ',3328,$67,startf,startin);
menu6.additem(' FOOT SW    ',3328,$67,footswf,footswin);
menu6.additem(' AUTO M     ',3328,$67,autof,autoin);
menu6.additem(' ORIGIN     ',3328,$67,originf,originin);
menu6.additem(' ESC          F10 ',17408,$67,escf,escin);
menu7.init(' POSI  F7 ',16640,64,10,$6F,$24);
menu7.additem(' POSI     F7 ',16640,$6F,posif);
menu7.additem(' SET V    F8 ',16896,$6F,setvaluef);
menu7.additem(' QUIT     F9 ',17152,$6F,quitf);
menu7.additem(' ESC     F10 ',17408,$6F,escf);
done    := false;
originb := false;
while not done do
begin
     escb      := false;
     helpb     := false;
     loadb     := false;
     newffb    := false;
     runab     := false;
     stepb     := false;
     manualb   := false;
     autob     := false;
     startb    := false;
     footswb   := false;
     originab  := false;
     posib     := false;
     setvalueb := false;
     quitb     := false;
     printmainmenu;
     readfunckey(ch);
     key := ord(ch) $ 256;
     if funckey then
        if key <> 17408 then
        begin
             menu1.open(key);
             if helpb then help(helpb);
             menu2.open(key);
             if loadb then load(loadb);
             if newffb then newf(newffb);
             menu3.open(key);
             if runab then runa(runab);
```

```
                menu4.open(key);
                if stepb then step(stepb);
                menu5.open(key);
                {if manualb then manual(manualb);}
                menu6.open(key);
                if startb then startm;
                if footswb then footswm;
                if autob   then autom;
                if originb then origin;
                menu7.open(key);
                if (posib And originb) then posi(posib);
                if setvalueb then setvalue;
                if quitb then exit;
            end;
    end;
    cursoron;
End.
```

```
Unit HelpMenu;
Interface
uses  win,pullmenu,keyboard;

    procedure help(helpb : boolean);

Implementation

var
   helpb1,done : boolean;
   escb1       : boolean;
   menuhelp1   : submenu;
   key         : word;
   ch          : char;

{$f+}

function exithf1 : boolean;
begin
     helpb1 := true;
     exithf1 := true;
end;

function eschf1 : boolean;
begin
     escb1 := true;
     eschf1 := true;
end;

procedure printhelp;
begin
     menuhelp1.drawtitle;
end;

procedure help(helpb : boolean);
begin
     menuhelp1.init(' EXIT  F9 ',17152,1,10,$6F,$24);
     menuhelp1.additem(' EXIT     F9 ',17152,$6F,exithf1);
     menuhelp1.additem(' ESC     F10 ',17408,$6F,eschf1);
     done := false;
     fillwin(1,1,80,1,177,15);
     setboxstyle(double);
     windowopen(2,2,16,24);
     writeln;
     writeln(' F1  HELP');
     writeln;
     writeln(' F2  EDIT');
     writeln;
     writeln;
     writeln(' F3  RUN ');
     writeln;
     writeln;
     writeln(' F4 SINGLE');
     writeln;
     writeln;
     writeln(' F5 MANUAL');
     writeln;
     writeln(' F6 OPTION');
```

```
        writeln;
        writeln(' F7 POSITION');
        writeln;
        writeln(' F9 EXIT');
        writeln;
        windowopen(17,2,79,24);
        writeln;
        writeln(' tell function of other mode');
        writeln;
        writeln(' F1 LOAD : read datas in file for modifier');
        writeln(' F2 NEW  : open new files for put new datas');
        writeln;
        writeln(' F1 LOAD : read datas in file and display');
        writeln(' F2 RUN  : sent data all step to control distance of motor');
        writeln;
        writeln(' F1 LOAD : read datas in file and display');
        writeln(' F2 STEP : sent data one step to control distance of motor');
        writeln;
        writeln(' control by user');
        writeln;
        writeln(' setup option');
        writeln;
        writeln(' show position ');
        writeln;
        writeln(' goto main menu ');
        writeln;
        printhelp;
        while not done do
        begin
            helpb1 := false;
            escb1 := false;
            readfunckey(ch);
            key := ord(ch) * 256;
            if funckey then
               if key <> 17408 then
                  begin
                      menuhelp1.open(key);
                  end;
            if helpb1 then done := true;
        end;
        windowclose;
        windowclose;
        windowclose;
end;

{$f-}

end.
```

```
unit loadmenu;

interface

uses crt,win,screen,variable,pullmenu,keyboard;

     procedure load(loadb : boolean);

implementation

var
   insertb,deleteb     : boolean;
   done,exitb          : boolean;
   escb,editb          : boolean;
   menuload1           : submenu;
   menuload2           : submenu;
   menuload3           : submenu;
   menuload4           : submenu;
   key                 : word;
   filename            : string;
   ch                  : char;
   outfile             : real;
   countstep           : integer;
   count               : integer;
   col,row             : byte;

{$f+}

function editf : boolean;
begin
     editb := true;
     editf := true;
end;

function insertf : boolean;
begin
     insertb := true;
     insertf := true;
end;

function escf : boolean;
begin
     escb := true;
     escf := true;
end;

function deletef : boolean;
begin
     deleteb := true;
     deletef := true;
end;

function exitlf : boolean;
begin
     exitb := true;
     exitlf := true;
end;
```

```
procedure printload;
begin
     menuload1.drawtitle;
     menuload2.drawtitle;
     menuload3.drawtitle;
     menuload4.drawtitle;
end;

procedure editp;
var
   stepn,i    : integer;
   code       : integer;
   dataedit   : real;
   temp1      : array[1..maxline-1] of real;
   done       : boolean;
begin
     if not (count = 0) then
     begin
          windowopen(5,12,32,15);
          repeat
               done := true;
               gotoxy(1,1);
               write('STEP NUMBER EDIT : ');    clreol;
               readint(stepn,2,code);
               writeln;
          until (stepn < count+1) and (stepn > 0 );
          write('NEW DATA : ');
          readreal(dataedit,6,code);
          if dataedit = 0 then done := false;
          if done then
          begin
              tempfile[stepn] := dataedit;
              window(46,5,69,21);
              for i := 1 to count do
              writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
          end
          else
          begin
               window(46,5,69,21);
               for i := 1 to count do
               writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
          end;
          window(1,1,80,25);
          windowclose;
     end;
     editb   := false;
end;

procedure insertp;
var
   stepn,i    : integer;
   code       : integer;
   datainsert : real;
   temp1      : array[1..maxline-1] of real;
   done       : boolean;
begin
     windowopen(5,12,32,15);
     repeat
```

```
     done := true;
     gotoxy(1,1);
     write('STEP NUMBER INSERT : ');    clreol;
     readint(stepn,2,code);
     writeln;
     until (stepn < count+2) and (stepn > 0 );
     write('NEW DATA : ');
     readreal(datainsert,6,code);
     if datainsert = 0 then done := false;
     if done then
     begin
          if count = maxline-1 then
          begin
               for i := 1 to count do
               tempi[i] := tempfile[i];
               tempfile[stepn] := datainsert;
               for i := stepn to count do
               tempfile[i+1] := tempi[i];
               window(46,5,69,21);
               for i := 1 to count do
               writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
          end
          else
          begin
               inc(count);
               for i := 1 to count do
               tempi[i] := tempfile[i];
               tempfile[stepn] := datainsert;
               for i := stepn to count do
               tempfile[i+1] := tempi[i];
               window(46,5,69,21);
               for i := 1 to count do
               writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
          end;
            end;
     window(1,1,80,25);
     windowclose;
     insertb := false;
end;

procedure deletep;
var
   stepn,i    : integer;
   code       : integer;
   datadelete : real;
   temp2      : array[1..maxline-1] of real;
   col,row    : byte;
begin
     if not (count = 0) then
     begin
          windowopen(5,12,32,14);
          repeat
               gotoxy(1,1);
               write('STEP NUMBER DELETE : ');   clreol;
               readint(stepn,2,code);
               writeln;
               until (stepn < count+1) and (stepn > 0 );
          for i := 1 to count do
```

```
            temp2[i] := tempfile[i];
            for i := stepn to count do
            tempfile[i] := temp2[i+1];
            tempfile[count] := 0;
            window(46,5,69,21);
            for i := 1 to count do
            writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
            col := wherex;      row := wherey;
            gotoxy(col,row-1);
            delline;
            dec(count);
            window(1,1,80,25);
            windowclose;
        end;
        deleteb := false;
    end;

    procedure load(loadb : boolean);
    var
       code    : integer;
       enterb  : boolean;
    begin
        menuload1.init(' EDIT    F1 ',15104,1,12,$6F,$24);
        menuload1.additem(' EDIT      F1 ',15104,$6F,editf);
        menuload1.additem(' ESC       F10 ',17408,$6F,escf);
        menuload2.init(' INSERT  F2 ',15360,13,12,$67,$24);
        menuload2.additem(' INSERT    F2 ',15360,$67,insertf);
        menuload2.additem(' ESC       F10 ',17408,$67,escf);
        menuload3.init(' DELETE  F3 ',15616,25,12,$6F,$24);
        menuload3.additem(' DELETE    F3 ',15616,$6F,deletef);
        menuload3.additem(' ESC       F10 ',17408,$6F,escf);
        menuload4.init(' EXIT    F9 ',17152,37,12,$67,$24);
        menuload4.additem(' EXIT      F9 ',17152,$67,exitlf);
        menuload4.additem(' ESC       F10 ',17408,$67,escf);
        done := false;
        fillwin(1,1,80,1,177,15);
        setboxstyle(single);
        windowopen(20,5,60,7);
        filename := '';
        repeat
             gotoxy(1,1);
             write('  ENTER FILE NAME :  ');
             cursoron;
             readstringnum(filename,3);
             cursoroff;
        until (filename <> #13);
        windowclose;
        assign(nfi,filename);
        if made(nfi) then
        begin
             setboxstyle(double);
             windowopen(2,2,79,24);
             setboxstyle(single);
             windowopen(45,4,70,22);
             countstep := 1;
             while not eof(nfi) do
             begin
                  read(nfi,outfile);
```

```
                tempfile[countstep] := outfile;
                writeln('  step  ',countstep:2,' : ',tempfile[countstep]:6:2);
                inc(countstep);
            end;
            count := countstep-1;
            close(nfi);
        end
        else
        begin
            windowopen(20,5,60,8);
            write('            FILE NOT FOUND   ');
            gotoxy(1,2);
            textcolor(white + blinkhigh);
            write('       HIT ');
            textcolor(white);
            write(' ANY KEY TO CONTINUES');
            ch := readkey;
            windowclose;
            windowclose;
            exit;
        end;
        window(1,1,80,25);
        printload;
        while not done do
        begin
            editb   := false;
            insertb := false;
            deleteb := false;
            exitb   := false;
            ch := readkey;
            if ord(ch) = 0 then
               key := ord(readkey)*256
            else
               key := ord(ch);
            if key <> 27 then
            begin
                menuload1.open(key);
                if editb then editp;
                menuload2.open(key);
                if insertb then insertp;
                menuload3.open(key);
                if deleteb then deletep;
                menuload4.open(key);
            end;
            if exitb then done := true;
        end;
        reset(nfi);
        for countstep := 1 to count do
           write(nfi,tempfile[countstep]);
        close(nfi);
        windowclose;
        windowclose;
    end;

    {$f-}

    end.
```

```
unit newfmenu;

interface

uses crt,win,screen,variable,pullmenu,keyboard;

procedure newf(newffb : boolean);

implementation

var
    insertb,deleteb     : boolean;
    done,exitb          : boolean;
    exitib,exitdb       : boolean;
    escb,editb          : boolean;
    menunewf1           : submenu;
    menunewf2           : submenu;
    menunewf3           : submenu;
    menunewf4           : submenu;
    key                 : word;
    ch                  : char;
    filename            : string;
    infile              : real;
    countstep           : integer;
    count               : integer;
    col,row             : byte;

{$f+}

function editf : boolean;
begin
     editb := true;
     editf := true;
end;

function insertf : boolean;
begin
     insertb := true;
     insertf := true;
end;

function escf : boolean;
begin
     escb := true;
     escf := true;
end;

function deletef : boolean;
begin
     deleteb := true;
     deletef := true;
end;

function exitlf : boolean;
begin
     exitb := true;
     exitlf := true;
end;
```

```
procedure printnew;
begin
     menunewf1.drawtitle;
     menunewf2.drawtitle;
     menunewf3.drawtitle;
     menunewf4.drawtitle;
end;

procedure editp;
var
   stepn,i    : integer;
   code       : integer;
   dataedit   : real;
   temp1      : array[1..maxline-1] of real;
   done       : boolean;
begin
     if not (count = 0) then
     begin
          windowopen(5,12,32,15);
          repeat
               done := true;
               gotoxy(1,1);
               write('STEP NUMBER EDIT : ');
               clreol;
               readint(stepn,2,code);
               writeln;
          until (stepn < count+1) and (stepn > 0 );
          write('NEW DATA : ');
          readreal(dataedit,6,code);
          if dataedit = 0 then done := false;
          if done then
          begin
              tempfile[stepn] := dataedit;
              window(46,5,69,21);
              for i := 1 to count do
              writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
          end
          else
          begin
               window(46,5,69,21);
               for i := 1 to count do
               writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
          end;
          window(1,1,80,25);
          windowclose;
     end;
     editb   := false;
end;

procedure insertp;
var
   stepn,i    : integer;
   code       : integer;
   datainsert : real;
   temp1      : array[1..maxline-1] of real;
   done       : boolean;
begin
windowopen(5,12,32,15);
```

```
    repeat
    done := true;
    gotoxy(1,1);
    write('STEP NUMBER INSERT : ');
    clreol;
    readint(stepn,2,code);
    writeln;
    until (stepn < count+2) and (stepn > 0 );
    write('NEW DATA : ');
    readreal(datainsert,6,code);
    if datainsert = 0 then done := false;
    if done then
    begin
         if count = maxline-1 then
         begin
              for i := 1 to count do
              temp1[i] := tempfile[i];
              tempfile[stepn] := datainsert;
              for i := stepn to count do
              tempfile[i+1] := temp1[i];
              window(46,5,69,21);
              for i := 1 to count do
              writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
         end
         else
         begin
              inc(count);
              for i := 1 to count do
              temp1[i] := tempfile[i];
              tempfile[stepn] := datainsert;
              for i := stepn to count do
              tempfile[i+1] := temp1[i];
              window(46,5,69,21);
              for i := 1 to count do
              writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
         end;
    end;
    window(1,1,80,25);
    windowclose;
    insertb := false;
end;

procedure deletep;
var
   stepn,i    : integer;
   code       : integer;
   datadelete : real;
   temp2      : array[1..maxline-1] of real;
   col,row    : byte;
begin
     if not (count = 0) then
     begin
          windowopen(5,12,32,14);
          repeat
               gotoxy(1,1);
               write('STEP NUMBER DELETE : ');
               clreol;
               readint(stepn,2,code);
```

```
                writeln;
            until (stepn < count+1) and (stepn > 0 );
            for i := 1 to count do
            temp2[i] := tempfile[i];
            for i := stepn to count do
            tempfile[i] := temp2[i+1];
            tempfile[count] := 0;
            window(46,5,69,21);
            for i := 1 to count do
            writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
            col := wherex;
            row := wherey;
            gotoxy(col,row-1);
            delline;
            dec(count);
            window(1,1,80,25);
            windowclose;
        end;
        deleteb := false;
    end;

    procedure newf(newffb : boolean);
    var
       code    : integer;
       enterb  : boolean;
    begin
        menunewf1.init(' EDIT    F1 ',15104,1,12,$6F,$24);
        menunewf1.additem(' EDIT      F1 ',15104,$6F,editf);
        menunewf1.additem(' ESC       F10 ',17408,$6F,escf);
        menunewf2.init(' INSERT  F2 ',15360,13,12,$67,$24);
        menunewf2.additem(' INSERT    F2 ',15360,$67,insertf);
        menunewf2.additem(' ESC       F10 ',17408,$67,escf);
        menunewf3.init(' DELETE  F3 ',15616,25,12,$6F,$24);
        menunewf3.additem(' DELETE    F3 ',15616,$6F,deletef);
        menunewf3.additem(' ESC       F10 ',17408,$6F,escf);
        menunewf4.init(' EXIT    F9 ',17152,37,12,$67,$24);
        menunewf4.additem(' EXIT      F9 ',17152,$67,exitlf);
        menunewf4.additem(' ESC       F10 ',17408,$67,escf);
        done := false;
        fillwin(1,1,80,1,177,15);
        setboxstyle(single);
        windowopen(20,5,60,7);
        filename := '';
        repeat
             gotoxy(1,1);
             write('  ENTER FILE NAME :  ');
             cursoron;
             readstringnum(filename,3);
             cursoroff;
        until (filename <> #13);
        windowclose;
        setboxstyle(double);
        windowopen(2,2,79,24);
        assign(nfi,filename);
        rewrite(nfi);
        setboxstyle(single);
        windowopen(45,4,70,22);
        countstep := 1;
```

```
repeat
      enterb := false;
      write('  step  ',countstep:2,' : ');
      col := wherex;
      row := wherey;
      cursoron;
      readreal(infile,6,code);
      cursoroff;
      if infile = 0 then
      begin
           gotoxy(1,row);
           clreol;
           enterb := true;
           dec(countstep);
      end
      else
      begin
           gotoxy(col,row);
           writeln(infile:6:2);
           write(nfi,infile);
           inc(countstep);
      end;
until (countstep = maxline) or enterb;
if countstep = maxline then countstep := maxline - 1;
count := countstep;
close(nfi);
reset(nfi);
for countstep := 1 to count do
   read(nfi,tempfile[countstep]);
close(nfi);
window(1,1,80,25);
printnew;
while not done do
begin
     editb   := false;
     insertb := false;
     deleteb := false;
     exitb   := false;
     ch := readkey;
     if ord(ch) = 0 then
        key := ord(readkey)*256
     else
        key := ord(ch);
     if key <> 27 then
     begin
          menunewf1.open(key);
          if editb then editp;
          menunewf2.open(key);
          if insertb then insertp;
          menunewf3.open(key);
          if deleteb then deletep;
          menunewf4.open(key);
     end;
     if exitb then done := true;
end;
reset(nfi);
for countstep := 1 to count do
   write(nfi,tempfile[countstep]);
```

```
        close(nfi);
        windowclose;
        windowclose;
    end;

    {$f-}

    end.
```

```
unit runmenu;

interface

uses crt,win,screen,keyboard,variable,pullmenu,movemenu;

     procedure runm(runb : boolean);

implementation

var
   runmb,escb          : boolean;
   done,exitrb         : boolean;
   escinb              : boolean;
   menurun1            : submenu;
   menurun2            : submenuo;
   menurun3            : submenu;
   key                 : word;
   ch                  : char;
   filename            : string;
   countstep           : integer;
   count               : integer;
   outfile             : real;

{$f+}

function runmf : boolean;
begin
     runmb := true;
     runmf := true;
end;

function escf : boolean;
begin
     escb := true;
     escf := true;
end;

Function Escin : Boolean;
begin
        Escinb := true;
        Escin  := true;
end;

function exitrf : boolean;
begin
     exitrb := true;
     exitrf := true;
end;

procedure printrun;
begin
     menurun1.drawtitle;
     menurun2.drawtitle;
     menurun3.drawtitle;
end;

procedure runm(runb : boolean);
```

```
var
   i : integer;
begin
     menurun1.init(' RUN     F1 ',15104,1,12,$6F,$24);
     menurun1.additem(' RUN        F3 ',15104,$6F,runmf);
     menurun1.additem(' ESC       F10 ',17408,$6F,escf);
     menurun2.init(' OPTION  F2 ',15360,13,12,$67,$24);
     menurun2.additem(' START M     ',3328,$67,startf,startin);
     menurun2.additem(' FOOT SW     ',3328,$67,footswf,footswin);
     menurun2.additem(' AUTO M      ',3328,$67,autof,autoin);
     menurun2.additem(' ORIGIN      ',3328,$67,originf,originin);
     menurun2.additem(' ESC           F10 ',17408,$67,escf,escin);
     menurun3.init(' EXIT    F9 ',17152,25,12,$6F,$24);
     menurun3.additem(' EXIT       F9 ',17152,$6F,exitrf);
     menurun3.additem(' ESC       F10 ',17408,$6F,escf);
     done      := false;
     fillwin(1,1,80,1,177,15);
     setboxstyle(single);
     windowopen(20,5,60,7);
     filename := '';
     repeat
          gotoxy(1,1);
          write('  ENTER FILE NAME :  ');
          cursoron;
          readstringnum(filename,3);
          cursoroff;
     until (filename <> #13);
     windowclose;
     assign(nfi,filename);
     if made(nfi) then
     begin
          for i := 1 to maxline do
             tempfile[i] := 0;
          setboxstyle(double);
          windowopen(2,2,79,24);
          setboxstyle(single);
          windowopen(45,4,70,22);
          countstep := 1;
          while not eof(nfi) do
          begin
               read(nfi,outfile);
               tempfile[countstep] := outfile;
               writeln('  step  ',countstep:2,' : ',tempfile[countstep]:6:2);
               inc(countstep);
          end;
          count := countstep-1;
          close(nfi);
     end
     else
     begin
          windowopen(20,5,60,8);
          write('             FILE NOT FOUND   ');
          gotoxy(1,2);
          textcolor(white + blinkhigh);
          write('       HIT ');
          textcolor(white);
          write(' ANY KEY TO CONTINUES');
          ch := readkey;
```

```
            windowclose;
            exit;
        end;
        while not done do
        begin
            runmb     := false;
            startb    := false;
            footswb   := false;
            autob     := false;
            originmb  := false;
            exitrb    := false;
            escb      := false;
            printrun;
            readfunckey(ch);
            key := ord(ch) % 256;
            if key <> 27 then
            begin
                menurun1.open(key);
                   if runmb then
                   begin
                        if ((tempsw and start) = start) then
                        begin
                            if not originb then origin;
                        end
                        else
                            begin
                                startm;
                                origin;
                            end;
                        movem(runmb,tempfile,select);
                   end;
                menurun2.open(key);
                   if startb   then startm;
                   if footswb  then footswm;
                   if autob    then autom;
                   if originmb then origin;
                menurun3.open(key);
            end;
            if exitrb then done := true;
        end;
        windowclose;
        windowclose;
end;

{$f-}

end.
```

```
unit stepmenu;

interface

uses crt,win,screen,keyboard,variable,pullmenu,movemenu;

     procedure step(stepb : boolean);

implementation

var
   stepmb,escb          : boolean;
   done,exitsb          : boolean;
   escinb,selectmb      : boolean;
   menustep1            : submenu;
   menustep2            : submenu;
   menustep3            : submenuo;
   menustep4            : submenu;
   key                  : word;
   ch                   : char;
   filename             : string;
   countstep            : integer;
   count                : integer;
   outfile              : real;

{$f+}

function selectmf : boolean;
begin
     selectmb := true;
     selectmf := true;
end;

function stepmf : boolean;
begin
     stepmb := true;
     stepmf := true;
end;

function escf : boolean;
begin
     escb := true;
     escf := true;
end;

Function Escin : Boolean;
begin
       Escinb := true;
       Escin  := true;
end;

function exitsf : boolean;
begin
     exitsb := true;
     exitsf := true;
end;

procedure selectm;
```

```
var
   i          : integer;
   code       : integer;
begin
     windowopen(5,12,32,14);
     if not (count = 0) then
     repeat
     gotoxy(1,1);
     write('STEP NUMBER SELECT : ');
     clreol;
     readint(select,2,code);
     writeln;
     until (select < count+1) and (select > 0 );
     window(46,5,69,21);
     for i := 1 to count do
     if i = select then
     begin
          textcolor($0E);
          writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
          textcolor($0F);
     end
     else
          writeln('  step  ',i:2,' : ',tempfile[i]:6:2);
     window(1,1,80,25);
     windowclose;
     selectab := false;

end;

procedure printstep;
begin
     menustep1.drawtitle;
     menustep2.drawtitle;
     menustep3.drawtitle;
     menustep4.drawtitle;
end;

procedure step(stepb : boolean);
var
   i : integer;
begin
     menustep1.init(' SELECT  F1 ',15104,1,12,$6F,$24);
     menustep1.additem(' STEP      F1 ',15104,$6F,selectmf);
     menustep1.additem(' ESC      F10 ',17408,$6F,escf);
     menustep2.init(' STEP    F2 ',15360,13,12,$67,$24);
     menustep2.additem(' STEP      F2 ',15360,$67,stepmf);
     menustep2.additem(' ESC      F10 ',17408,$67,escf);
     menustep3.init(' OPTION  F3 ',15616,25,12,$6F,$24);
     menustep3.additem(' START M     ',3328,$6F,startf,startin);
     menustep3.additem(' FOOT SW     ',3328,$6F,footswf,footswin);
     menustep3.additem(' AUTO M      ',3328,$6F,autof,autoin);
     menustep3.additem(' ORIGIN      ',3328,$6F,originf,originin);
     menustep3.additem(' ESC           F10 ',17408,$6F,escf,escin);
     menustep4.init(' EXIT    F9 ',17152,37,12,$67,$24);
     menustep4.additem(' EXIT      F9 ',17152,$67,exitsf);
     menustep4.additem(' ESC      F10 ',17408,$67,escf);
     done := false;
     fillwin(1,1,80,1,177,15);
```

```
setboxstyle(single);
windowopen(20,5,60,7);
filename := '';
repeat
      gotoxy(1,1);
      write('  ENTER FILE NAME :  ');
      cursoron;
      readstringnum(filename,3);
      cursoroff;
until (filename <> #13);
windowclose;
assign(nfi,filename);
if made(nfi) then
begin
     for i := 1 to maxline do
         tempfile[i] := 0;
     setboxstyle(double);
     windowopen(2,2,79,24);
     setboxstyle(single);
     windowopen(45,4,70,22);
     countstep := 1;
     while not eof(nfi) do
     begin
          read(nfi,outfile);
          tempfile[countstep] := outfile;
          writeln('  step  ',countstep:2,' : ',tempfile[countstep]:6:2);
          inc(countstep);
     end;
     count := countstep-1;
     close(nfi);
end
else
begin
     windowopen(20,5,60,8);
     write('            FILE NOT FOUND   ');
     gotoxy(1,2);
     textcolor(white + blinkhigh);
     write('      HIT ');
     textcolor(white);
     write(' ANY KEY TO CONTINUES');
     ch := readkey;
     windowclose;
     exit;
end;
while not done do
begin
     selectmb := false;
     stepmb   := false;
     startb   := false;
     footswb  := false;
   autob    := false;
     originmb := false;
     exitsb   := false;
     escb     := false;
     printstep;
     readfunckey(ch);
     key := ord(ch) * 256;
     if key <> 27 then
```

```
          begin
               menustep1.open(key);
                  if selectmb then selectm;
               menustep2.open(key);
                  if stepmb then
                  begin
                       if ((tempsw and start) = start) then
                       begin
                            if not originb then origin;
                       end
                       else
                           begin
                                startm;
                                origin;
                           end;
                       movem(stepmb,tempfile,select);
                  end;
               menustep3.open(key);
                  if startb   then startm;
                  if footswb  then footswm;
                  if autob    then autom;
                  if originmb then origin;
               menustep4.open(key);
          end;
          if exitsb then done := true;
     end;
     windowclose;
     windowclose;
end;

{$f-}

end.
```

```
Unit MoveMenu;

Interface

uses  crt,keyboard,screen,variable,win,pullmenu,unitposi;

    procedure Movem(Movepb : boolean;tempfile : temparray;select : integer);

Implementatidn

var
   exitmb,done,escb     : boolean;
   menumovel            : submenu;
   key                  : word;
   ch                   : char;

{$f+}

function exitmf : boolean;
begin
     exitmb := true;
     exitmf := true;
end;

function escmf : boolean;
begin
     escb := true;
     escmf := true;
end;

procedure printmove;
begin
     menumovel.drawtitle;
end;

procedure movem(movepb : boolean;tempfile : temparray;select : integer);
var
   i             : integer;
   step          : longint;
   stepbreak     : longint;
   tempin        : integer;
begin
     menumovel.init(' EXIT  F9 ',17152,1,10,$6F,$24);
     menumovel.additem(' EXIT     F9 ',17152,$6F,exitmf);
     menumovel.additem(' ESC     F10 ',17408,$6F,escmf);
     done := false;
     fillwin(1,1,80,1,177,15);
     setboxstyle(double);
     windowopen(2,2,79,24);
     printmove;
     while not done do
     begin
          exitmb := false;
          escb  := false;
          position(tempdis);
          for i := 0 to 2 do
          begin
               if (tempfile[select+i] = 0) then
```

```
        begin
            gotoxy(38,i+4);
            write('          ');
        end
        else
        begin
            gotoxy(38,i+4);
            write(tempfile[select+i]:5:2);
        end;
    end;
    if not (tempfile[select] = 0) then
    begin
        if tempfile[select] > tempdis then                {Reve}
        begin
            if ((tempfile[select] - tempdis) > 50)  then
            begin
                step := round((tempfile[select] - tempdis) * Invk);
                stepbreak := step - stepbreak5v;
                Port[Pout1] := (tempsw or ReveM2);  {Reve}
                delay(100);
                Port[Pout2] := Speed5;
                delay(100);
                Port[Pread3] := $80;              {Clear Count}
                Delay(10);
                Port[Pread3] := $00;
                Delay(10);
                Repeat
                    occurstep1 := ((tempdis) + (occurstep * k));
                    position(occurstep1);
                Until (Occurstep >= stepbreak);
                Port[pout2] := Speed_5;
                delay(100);
                Repeat
                    occurstep1 := ((tempdis) + (occurstep * k));
                    position(occurstep1);
                Until (Occurstep >= step);
                Port[Pout1] := (tempsw and (not ReveM2));
                delay(100);
                Port[Pout2] := Nor;
                delay(100);
                for i := 1 to 10 do
                begin
                    occurstep1 := ((tempdis) + (occurstep * k));
                    position(occurstep1);
                end;
                tempdis := occurstep1;
            end
            else
            begin
                step := round((tempfile[select] - tempdis) * Invk);
                stepbreak := step - stepbreak2v;
                Port[Pout1] := (tempsw or ReveM2);  {Reve}
                delay(100);
                Port[Pout2] := Speed2;
                delay(100);
                Port[Pread3] := $80;              {Clear Count}
                Delay(10);
                Port[Pread3] := $00;
```

```
                Delay(10);
                Repeat
                     occurstep1 := ((tempdis) + (occurstep * k));
                     position(occurstep1);
                Until (Occurstep >= stepbreak);
                Port[pout2] := Speed_5;
                delay(100);
                Repeat
                     occurstep1 := ((tempdis) + (occurstep * k));
                     position(occurstep1);
                Until (Occurstep >= step);
                Port[Pout1] := (tempsw and (not RevaM2));
                delay(100);
                Port[Pout2] := Nor;
                delay(100);
                for i := 1 to 10 do
                begin
                    occurstep1 := ((tempdis) + (occurstep * k));
                    position(occurstep1);
                end;
                tempdis := occurstep1;
           end;
      end
      else
      begin                                          {Forw}
           if ((tempdis - tempfile[select]) > 50) then
           begin
                step := round((tempdis - tempfile[select]) * Invk);
                stepbreak := step - stepbreak5v;
                Port[Pout1] := (tempsw or ForwM2);  {Forw}
                delay(100);
                Port[Pout2] := Speed5;
                delay(100);
                Port[Pread3] := $80;              {Clear Count}
                Delay(10);
                Port[Pread3] := $00;
                Delay(10);
                Repeat
                     occurstep1 := ((tempdis) - (occurstep * k));
                     position(occurstep1);
                Until (Occurstep >= stepbreak);
                Port[pout2] := Speed_5;
                delay(100);
                Repeat
                     occurstep1 := ((tempdis) - (occurstep * k));
                     position(occurstep1);
                Until (Occurstep >= step);
                Port[Pout1] := (tempsw and (not ForwM2));
                delay(100);
                Port[Pout2] := Nor;
                delay(100);
                for i := 1 to 10 do
                begin
                     occurstep1 := ((tempdis) - (occurstep * k));
                     position(occurstep1);
                end;
                tempdis := occurstep1;
           end
```

```
                else
                begin
                    step := round((tempdis - tempfile[select]) * Invk);
                    stepbreak := step - stepbreak2v;
                    Port[Pout1] := (tempsw or ForwM2);  {Forw}
                    delay(100);
                    Port[Pout2] := Speed2;
                    delay(100);
                    Port[Pread3] := $80;              {Clear Count}
                    Delay(10);
                    Port[Pread3] := $00;
                    Delay(10);
                    Repeat
                        occurstep1 := ((tempdis) - (occurstep * k));
                        position(occurstep1);
                    Until (Occurstep >= stepbreak);
                    Port[pout2] := Speed_5;
                    delay(100);
                    Repeat
                        occurstep1 := ((tempdis) - (occurstep * k));
                        position(occurstep1);
                    Until (Occurstep >= step);
                    Port[Pout1] := (tempsw and (not ForwM2));
                    delay(100);
                    Port[Pout2] := Nor;
                    delay(100);
                    for i := 1 to 10 do
                    begin
                        occurstep1 := ((tempdis) - (occurstep * k));
                        position(occurstep1);
                    end;
                    tempdis := occurstep1;
                end;
            end;
            inc(select);
        end;
        repeat
            tempin := port[pin1];
        until ((tempin and $80) = $80);
        repeat
            tempin := port[pin2];
        until ((tempin and $01) = $01);
        repeat
            tempin := port[pin1];
        until ((tempin and $40) = $40);

        if (tempfile[select] = 0) then
           begin
               readfunckey(ch);
               key := ord(ch) * 256;
               if funckey then
               if key <> 17408 then
               begin
                   menumove1.open(key);
               end;
               if exitmb then done := true;
           end;
    end;
```

```
    windowclose;
end;

{$f-}

end.
```

```
Unit PosiMenu;

Interface

uses  crt,keyboard,variable,win,pullmenu,unitposi;

    procedure Posi(Posipb : boolean);

Implementation

var
   posib,done,escb     : boolean;
   menuposition1       : submenu;
   key                 : word;
   ch                  : char;
   speed,i             : integer;

{$f+}

function exitpf : boolean;
begin
     posib  := true;
     exitpf := true;
end;

function escpf : boolean;
begin
     escb  := true;
     escpf := true;
end;

procedure printposi;
begin
     menuposition1.drawtitle;
end;

procedure posi(posipb : boolean);
begin
     menuposition1.init(' EXIT  F9 ',17152,1,10,$6F,$24);
     menuposition1.additem(' EXIT     F9 ',17152,$6F,exitpf);
     menuposition1.additem(' ESC     F10 ',17408,$6F,escpf);
     done := false;
     fillwin(1,1,80,1,177,15);
     setboxstyle(double);
     windowopen(2,2,79,24);
     printposi;
     while not done do
     begin
             posib := false;
             escb  := false;
             position(tempdis);
             readfunckey(ch);
             key := ord(ch) * 256;
             if funckey then
                if key <> 17408 then
                   begin
                        menuposition1.open(key);
                    end;
```

```
                if posib then done := true;
        end;
        windowclose;
end;

{$f-}

end.
```

```
Unit Variable;

Interface

Uses dos,crt,win,unitposi,keyboard;

Const
        Pctrl1      = $1b3;   Pctrl2      = $1B7;    Pread1    = $1b0;
        Pread2      = $1b1;   Pread3      = $1b2;    Pout1     = $0280;
        Pout2       = $0283;  Pin1        = $2280;   Pin2      = $2283;
          Speed_5     = $10;    Speed1      = $20;     Speed2    = $40;
          Speed5      = $80;    Start       = $43;     Stop      = $40;
          Footswitch  = $20;    Autoswitch  = $04;     ForwM3    = $10;
          ReveM3      = $08;    ForwM2      = $90;     ReveM2    = $88;
          StepBreak5v = 4000;   StepBreak2v = 2500;    k         = 0.00390625;
          InvK        = 256;    Maxline     = 17;      Nor = $00;

Type
        temparray = Array[1..maxline] of real;
        filetype = file of real;
Var
        tempfile     : temparray;
        tempdis      : real;
        occurstep1   : real;
        originpoint  : real;
        lastpoint    : real;
        tempsw       : integer;
        select       : integer;
        nfi          : filetype;
        startb       : boolean;
        footswb      : boolean;
        autob        : boolean;
        originb      : boolean;
        originmb     : boolean;
        setvalueb    : boolean;

        Function OccurStep : longint;
        Function made(var fi : filetype) : boolean;
        Function Startf   : Boolean;
        Function Startin  : Boolean;
        Function footswf  : Boolean;
        Function Footswin : Boolean;
        Function Autof    : boolean;
        Function Autoin   : Boolean;
        Function Originf  : Boolean;
        Function Originin : Boolean;
        Procedure Set8255;
        Procedure Stopm;
        Procedure Startm;
        Procedure Footswm;
        Procedure Autom;
        Procedure origin;
        Procedure setvalue;

Implementation

Function OccurStep : longint;
Var
```

```
        P1,P2,P3 : Longint;
Begin
        P1 := (Port[Pread1]);
        P2 := (Port[Pread2]);
        P3 := (Port[Pread3]);
        OccurStep := (P3*256*256)+(P2*256)+P1;
End;

Function made(var fi : filetype) : boolean;
Begin
    {$i-}  reset(fi);  {$i+}
    made := (ioresult = 0);
End;

Function Startf : Boolean;
begin
        if Startb then
        begin
            Startb := false;
            Startf := true;
        end
        else
        begin
            Startb := true;
            Startf := true;
        end;
end;

Function Startin : Boolean;
begin
        if ((tempsw and $43) = $43) then
        begin
            startin  := true;
        end
        else
        begin
            startin  := false;
        end;
end;

Function footswf : Boolean;
begin
        if footswb then
        begin
            footswb := false;
            footswf := true;
        end
        else
        begin
            footswb := true;
            footswf := true;
        end;
end;

Function Footswin : Boolean;
begin
        if ((tempsw and $20) = $20) then
        begin
```

```
            footswin  := true;
        end
        else
        begin
            footswin  := false;
        end;
end;

Function Autof : boolean;
begin
        if autob then
        begin
            autob := false;
            autof := true;
        end
        else
        begin
            autob := true;
            autof := true;
        end;
end;

Function Autoin : Boolean;
begin
        {if ((port[pin2] and $02) = $02) then}
        if ((tempsw and $04) = $04) then
        begin
            Autoin  := true;
        end
        else
        begin
            Autoin  := false;
        end;
end;

Function Originf : Boolean;
begin
    originab := true;
    originf := true;
end;

Function Originin : Boolean;
begin
    if originb then
        originin := true
    else
        originin := false;
end;

Procedure Set8255;
Begin
        Port[Pctrl1] := $93;
        Delay(10);
        Port[Pctrl2] := $80;
        Delay(10);
End;

Procedure Stopa;
```

```
Begin
        {stop = $40}
        Port[Pout1] := stop;
        tempsw := stop;
        Delay(10);
End;

Procedure Starta;
Begin
        {Start = $43}
        {stop  = $40}
        if ((tempsw and start) = start) then
        begin
             Port[Pout1] := stop;
             tempsw := stop;
             Delay(10);
        end
        else
        begin
             Port[Pout1] := (tempsw or Start);
             delay(300);
             tempsw := (tempsw or start);
        end;
End;

Procedure Footswa;
Begin
        {Footswitch = $20}
        if ((tempsw and footswitch) = footswitch) then
        begin
             Port[pout1] := (tempsw and (not footswitch));
             delay(10);
             tempsw := (tempsw and (not footswitch));
        end
        else
        begin
             Port[pout1] := (tempsw or footswitch);
             delay(10);
             tempsw := (tempsw or footswitch);
        end;
end;

Procedure Autoa;
Begin
        {Autoswitch = $04}
        {if ((port[pin2] and $02) = $02) then}
        if ((tempsw and autoswitch) = autoswitch) then
        begin
             Port[pout1] := (tempsw and (not autoswitch));
             delay(10);
             tempsw := (tempsw and (not autoswitch));
        end
        else
        begin
             Port[pout1] := (tempsw or autoswitch);
             delay(10);
             tempsw := (tempsw or autoswitch);
        end;
```

```
end;

Procedure Origin;                        {Go To Origin}
var
        i          : integer;
        step1done  : boolean;
Begin
        fillwin(1,1,80,1,177,15);
        setboxstyle(double);
        windowopen(2,2,79,24);
        step1done := false;
        Port[Pout1] := (tempsw or ForwM2);  {Forw}
        delay(100);
        if ((Port[Pin1] And $20) = $00) then
        else
        begin
             Port[Pout2] := Speed5;
             delay(100);
             Port[Pread3] := $80;              {Clear Count}
             Delay(10);
             Port[Pread3] := $00;
             Delay(10);
             Repeat
                  occurstep1 := (lastpoint - (occurstep * k));
                  position(occurstep1);
             Until  ((Port[Pin1] And $20) = $00);
             Port[Pout1]:= (tempsw  and (not ForwM2));
             delay(100);
             Port[Pout2]:= Nor;
             delay(100);
             for i := 1 to 10 do
                 begin
                     occurstep1 := (lastpoint - (occurstep * k));
                     position(occurstep1);
                 end;
             step1done := true;
        end;
        Port[Pout1] := (tempsw or ReveM2);  {Reve}
        delay(100);
        Port[Pout2] := Speed_5;
        delay(100);
        Port[Pread3] := $80;              {Clear Count}
        Delay(10);
        Port[Pread3] := $00;
        Delay(10);
        Repeat
              if step1done then
              begin
                   occurstep1 := ((originpoint - 2) + (occurstep * k));
                   position(occurstep1);
              end
              else
              begin
                   occurstep1 := (originpoint + (occurstep * k));
                   position(occurstep1);
              end;
        Until (Occurstep >= 3000);
        Port[Pout1] := (tempsw and (not ReveM2));
```

```
        delay(100);
        Port[Pout2] := Nor;
        delay(100);
        for i := 1 to 10 do
              if stepidone then
              begin
                   occurstep1 := ((originpoint - 2) + (occurstep * k));
                   position(occurstep1);
              end
              else
             begin
                   occurstep1 := (originpoint + (occurstep * k));
                   position(occurstep1);
              end;
        tempdis := occurstep1;
        Port[Pout1] := (tempsw or ForwM2 );   {Forw}
        Delay(100);
        Port[Pout2] := Speed_5;
        delay(100);
        Port[Pread3] := $80;            {Clear Count}
        Delay(10);
        Port[Pread3] := $00;
        Delay(10);
        Repeat
                   occurstep1 := (tempdis - (occurstep * k));
                   position(occurstep1);
        Until ((Port[Pin1] And $20) = $00);
        Port[Pout1]:= (tempsw and (not ForwM2));
        delay(100);
        Port[Pout2]:= Nor;
        delay(100);
        for i := 1 to 10 do
             begin
                   occurstep1 := (tempdis - (occurstep * k));
                   position(occurstep1);
             end;
        tempdis := originpoint;
        Port[Pread3] := $80;            {Clear Count}
        Delay(10);
        Port[Pread3] := $00;
        delay(10);
        windowclose;
        originb := true
End;

Procedure setvalue;
var
   code         : integer;
begin
     windowopen(5,12,48,15);
     write('SET VALUE ORIGINPOINT : (',originpoint:6:2,') ');
     clreol;
     readreal(originpoint,6,code);
     writeln;
     write('SET VALUE LASTPOINT   : (',lastpoint:6:2,') ');
     readreal(lastpoint,6,code);
     writeln;
     window(1,1,80,25);
```

```
        windowclose;
        assign(nfi,'set.ovl');
        reset(nfi);
        write(nfi,originpoint);
        write(nfi,lastpoint);
        close(nfi);
        setvalueb := false;

    end;

    End.
```

```
Unit PullMenu;

Interface

Type
    Proc = Function : Boolean;
    MenuItem = Record
             Text : String[80];
             Attr : Byte;
             DoThis : Proc;
             Key : Word;
    End;
    SubMenu = Object
            Title : String[80];
            List : Array[1..25] Of MenuItem;
            Count : Integer;
            Left : Integer;
            Width : Integer;
            Attr : Byte;
            CursorAttr : Byte;
            Key : Word;

            Procedure Init(Ttle : String; Ky : word; Lft,Wid : Integer
                  ;At1,At2 : Byte);
            Procedure AddItem(Text : string; Ky : Word; At : Byte
                  ;DoIt : Proc);
            Procedure DrawTitle;
            Procedure Open(Ky : Word);
    End;

    MenuItemo = Record
             Text : String[80];
             Attr : Byte;
             DoThis : Proc;
             Active : Proc;
             Key : Word;
    End;
    SubMenuo = Object
            Title : String[80];
            List : Array[1..25] Of MenuItemo;
            Count : Integer;
            Left : Integer;
            Width : Integer;
            Attr : Byte;
            CursorAttr : Byte;
            Key : Word;
            Procedure Init(Ttle : String; Ky : word; Lft,Wid : Integer
                  ;At1,At2 : Byte);
            Procedure AddItem(Text : string; Ky : Word; At : Byte
                  ;DoIt : Proc;Activethis : Proc);
            Procedure DrawTitle;
            Procedure Open(Ky : Word);
    End;

Implementation

Uses   Crt,Win,keyboard;
```

```
Procedure SubMenu.Init(Ttle : String; Ky : word; Lft,Wid : Integer
                    ;At1,At2 : Byte);
     Begin
          Title      := Ttle;
          Count      := 0;
          Left       := Lft;
          Width      := Wid;
          Attr       := At1;
          CursorAttr := At2;
          Key        := Ky;
     End;

Procedure SubMenu.AddItem(Text : string; Ky : Word; At : Byte
                        ;DoIt : Proc);
     Begin
          Inc(Count);
          List[Count].Text   := Text;
          List[Count].Attr   := At;
          List[Count].DoThis := DoIt;
          List[Count].Key    := Ky;
     End;

Procedure Submenu.DrawTitle;
     Begin
          window(1,1,80,25);
          TextAttr := Attr;
          GotoXY(Left + (Width - Length(Title)) Div 2,1);
          Write(Title);
     End;

Function NewCur(Cur,Change,Max : Integer) : Integer;
Var
   i : Integer;
     Begin
          i := Cur + Change;
          If i > Max Then   i := Max
          Else If i < 1 Then
             i := 1;
          NewCur := i;
     End;

procedure Submenu.open(ky : word);
var
   i,j : integer;
   ch : char;
   Userkey : word;
   Done : boolean;
   curitem : integer;
begin
    if ky = key then
    begin
        setboxstyle(single);
        windowopen(left,2,left+width+4,count+4);
        textattr := attr;
        for i := 1 to count do
        begin
            gotoxy(1,i);
```

```
            write(list[i].text);
        end;
        curitem := 1;
        done := false;
        while Not done do
        begin
            textattr := cursorattr;
            gotoxy(1,curitem);
            write(list[curitem].text);
            ch := readkey;
            if ord(ch) = 0 then
              userkey := ord(readkey) * 256
            else
               userkey := ord(ch);
            TextAttr := Attr;
            GotoXY(1,curItem);
            Write(List[CurItem].Text);
            Case UserKey of
                 27    : Done := True;
                 20480 : CurItem := NewCur(CurItem,1,Count);
                 18432 : CurItem := NewCur(CurItem,-1,Count);
                 13    : Done := List[CurItem].DoThis;
            Else
                for i := 1 to Count do
                    if UserKey = List[i].Key Then
                       Done := List[i].Dothis;
            end;
        end;
        windowclose;
    end;
end;

Procedure SubMenuo.Init(Title : String; Ky : word; Lft,Wid : Integer
                    ;At1,At2 : Byte);
      Begin
           Title     := Title;
           Count     := 0;
           Left      := Lft;
           Width     := Wid;
           Attr      := At1;
           CursorAttr := At2;
           Key       := Ky;
      End;

Procedure SubMenuo.AddItem(Text : string; Ky : Word; At : Byte
                        ;DoIt : Proc;Activethis : Proc);
      Begin
           Inc(Count);
           List[Count].Text   := Text;
           List[Count].Attr   := At;
           List[Count].DoThis := DoIt;
           List[Count].Active := Activethis;
           List[Count].Key    := Ky;
      End;

Procedure Submenuo.DrawTitle;
      Begin
           window(1,1,80,25);
```

```
            TextAttr := Attr;
            GotoXY(Left + (Width - Length(Title)) Div 2,1);
            Write(Title);
        End;

procedure Submenuo.open(ky : word);
var
   i,j        : integer;
   ch         : char;
   ch1        : array[1..25] of char;
   Userkey    : word;
   Done       : boolean;
   Done1      : Boolean;
   activenow  : boolean;
   donethis   : boolean;
   curitem    : integer;
   st         : string[4];
begin
    if ky = key then
    begin
        for j := 1 to 25 do
        ch1[j] := ' ';
        Done1 := false;
        setboxstyle(single);
        windowopen(left,2,left+width+10,count+4);
        textattr := attr;
        for i := 1 to count-1 do
        begin
            activenow := list[i].active;
            if activenow then st := ' ON '
            else st := 'OFF ';
            gotoxy(1,i);
            write(list[i].text,' ',ch1[count],'  ',st);
        end;
        write(list[count].text);
        curitem := 1;
        donethis := false;
        while Not donethis do
        begin
             done := false;
             while Not done do
             begin
                  if curitem = count then
                  begin
                       TextAttr := cursorattr;
                       GotoXY(1,curItem);
                       Write(List[CurItem].Text);
                  end
                  else
                  begin
                       activenow := list[curitem].active;
                       if activenow then st := ' ON '
                       else st := 'OFF ';
                       if done1 then
                          begin
                              if ch1[curitem] = 'x' then
                                    ch1[curitem] := ' '
                           else
```

```
                                    ch1[curitem] := 'x';
                               end;
                          textattr := cursorattr;
                          gotoxy(1,curitem);
                          write(list[curitem].text,' ',ch1[curitem],'  ',st);
                     end;
                     readarrowenterf10(userkey);
                     if curitem = count then
                     begin
                          TextAttr := Attr;
                          GotoXY(1,curItem);
                          Write(List[CurItem].Text);
                     end
                     else
                     begin
                          activenow := list[curitem].active;
                          if activenow then st := ' ON '
                          else st := 'OFF ';
                          TextAttr := Attr;
                          GotoXY(1,curItem);
                          Write(List[CurItem].Text,' ',ch1[curitem],'  ',st);
                     end;
                     Done1 := false;
                     Case UserKey of
                          20480 : CurItem := NewCur(CurItem,1,Count);
                          18432 : CurItem := NewCur(CurItem,-1,Count);
                           3328 : Done := List[CurItem].DoThis;
                     end;
                     if userkey = 17408 then
                          for i := 1 to Count do
                          if UserKey = List[i].Key Then
                             Done := List[i].Dothis;
                end;
           if ((curitem = count) and (userkey = 3328)) or (Userkey = 17408) then
           if UserKey = 3328 then
                Donethis := List[CurItem].DoThis;
           if userkey = 17408 then
                for i := 1 to Count do
                if UserKey = List[i].Key Then
                   Donethis := List[i].Dothis;
           Done1 := Done;
           end;
           windowclose;
      end;
end;

end.
```

```
unit UNITposi;

interface

uses crt;

        procedure position(r : real);

implementation

procedure  put0(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     textattr := $0F;
end;

procedure  put1(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write('    ',#219,#219,'   ');
     y := y + 1;
     gotoxy(x,y);
     write('    ',#219,#219,'   ');
     y := y + 1;
     gotoxy(x,y);
     write('    ',#219,#219,'   ');
     y := y + 1;
     gotoxy(x,y);
     write('    ',#219,#219,'   ');
     y := y + 1;
     gotoxy(x,y);
     write('    ',#219,#219,'   ');
     y := y + 1;
     gotoxy(x,y);
```

```
     write('    ',#219,#219,'    ');
     textattr := $0F;
end;

procedure  put2(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219'       ');
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219'       ');
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     textattr := $0F;
end;

procedure  put3(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     textattr := $0F;
end;

procedure  put4(x,y : byte);
begin
```

```
    textattr := $0E;
    gotoxy(x,y);
    write(#219,#219,'     ',#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219,'     ',#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219,'     ',#219,#219);
    y := Y + 1;
    gotoxy(x,y);
    write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write('       '#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write('       '#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write('       '#219,#219);
    textattr := $0F;
end;

procedure  put5(x,y : byte);
begin
    textattr := $0E;
    gotoxy(x,y);
    write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219'       ');
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219'       ');
    y := Y + 1;
    gotoxy(x,y);
    write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write('       '#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write('       '#219,#219);
    Y := y + 1;
    gotoxy(x,y);
    write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
    textattr := $0F;
end;

procedure  put6(x,y : byte);
begin
    textattr := $0E;
    gotoxy(x,y);
    write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
    y := y + 1;
    gotoxy(x,y);
    write(#219,#219'       ');
```

```
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219'       ');
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     textattr := $0F;
end;

procedure  put7(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     textattr := $0F;
end;

procedure  put8(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219);
```

```
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219,#219);
     textattr := $0F;
end;

procedure  put9(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,'     ',#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write('       '#219,#219);
     y := y + 1;
     gotoxy(x,y);
     write(#219,#219,#219,#219,#219,#219,#219,#219,#219,#219);
     textattr := $0F;
end;

procedure point(x,y : byte);
begin
     textattr := $0E;
     gotoxy(x,y);
     write(#219,#219);
     textattr := $0F;
end;

procedure puthun(a : real);
var
     a1 : integer;
begin
     a  := a/100;
     a1 := trunc(a);
     case a1 of
          0 : put0(9,13);
          1 : put1(9,13);
          2 : put2(9,13);
          3 : put3(9,13);
          4 : put4(9,13);
```

```
            5 : put5(9,13);
            6 : put6(9,13);
            7 : put7(9,13);
            8 : put8(9,13);
            9 : put9(9,13);
        end;
    end;

    procedure putten(b : real);
    var
        b1 : integer;
    begin
        b  := b/100;
        b  := frac(b);
        b  := b * 10;
        b1 := trunc(b);
        case b1 of
            0 : put0(21,13);
            1 : put1(21,13);
            2 : put2(21,13);
            3 : put3(21,13);
            4 : put4(21,13);
            5 : put5(21,13);
            6 : put6(21,13);
            7 : put7(21,13);
            8 : put8(21,13);
            9 : put9(21,13);
        end;
    end;

    procedure putone(c : real);
    var
        c1 : integer;
    begin
        c  := c/10;
        c  := frac(c);
        c  := c * 10;
        c1 := trunc(c);
        case c1 of
            0 : put0(33,13);
            1 : put1(33,13);
            2 : put2(33,13);
            3 : put3(33,13);
            4 : put4(33,13);
            5 : put5(33,13);
            6 : put6(33,13);
            7 : put7(33,13);
          8 : put8(33,13);
            9 : put9(33,13);
        end;
    end;

    procedure putpointone(d : real);
    var
       d1 : integer;
    begin
        d  := frac(d);
        d  := d * 10;
```

```
     d1 := trunc(d);
     case d1 of
          0 : put0(50,13);
          1 : put1(50,13);
          2 : put2(50,13);
          3 : put3(50,13);
          4 : put4(50,13);
          5 : put5(50,13);
          6 : put6(50,13);
          7 : put7(50,13);
          8 : put8(50,13);
          9 : put9(50,13);
     end;
end;

procedure putpointten(e : real);
var
    e1 : integer;
begin
     e  := frac(e);
     e  := e * 10;
     e  := frac(e);
     e  := e * 10;
     e1 := trunc(e);
     case e1 of
          0 : put0(62,13);
          1 : put1(62,13);
          2 : put2(62,13);
          3 : put3(62,13);
          4 : put4(62,13);
          5 : put5(62,13);
          6 : put6(62,13);
          7 : put7(62,13);
          8 : put8(62,13);
          9 : put9(62,13);
     end;
end;

Procedure Position(r : real);

begin
     window(1,1,80,25);
     puthun(r);
     putten(r);
     putone(r);
     point(45,19);
     putpointone(r);
     putpointten(r);
end;

end.
```

```
unit Win;

interface

uses Crt,screen;

 const  single    = 1;
        double    = 2;
        Mix1      = 3;
        Mix2      = 4;
        Boxstyle          : byte          = Single;
        AttrOfBox         : byte          = HighDisplay;
        AttrOfWindow      : byte          = HighDisplay;
        AttrOfHeader      : byte          = HighDisplay;
        AttrOfChar        : byte          = HighDisplay;
        HeaderOfWindow    : string        = '';
        TypeOfBox         : array [1..4,1..8] of char =
                            ((#196,#179,#179,#196,#218,#191,#192,#217),
                             (#205,#186,#186,#205,#201,#187,#200,#188),
                             (#205,#179,#179,#205,#213,#184,#212,#190),
                             (#196,#186,#186,#196,#214,#183,#211,#189)
                            );

 var   ErrorWindow : byte;

 procedure WindowBox  (x1,y1,x2,y2 : byte);
 procedure WindowOpen (x1,y1,x2,y2 : byte);
 procedure WindowClose;
 procedure FillWin(x1,y1,x2,y2,Acii,Attr:Byte);
 procedure SetBoxStyle (Attrib : byte);
 procedure SetBoxAttr  (Attrib : byte);
 procedure SetWinAttr  (Attrib : byte);
 procedure SetHeadAttr (Attrib : byte);
 procedure SetCharAttr (Attrib : byte);
 procedure SetWinHeader (st : string);
        { end interface }


implementation

 Type   ScreenLine  = array [1..80] of integer;
        ScreenArray = array [1..25] of screenline;
        ScreenBlock = array [1..2000] of integer;
        WindowLink  = ^WindowControlBlock;
        WindowControlBlock = record
                               x1,y1,x2,y2 : integer;
                               x,y         : integer;
                               ID          : byte;
                               Backlink       :  WindowLink;
                               ScreenContents :  ScreenBlock;
                             end;
 var  ActiveWindow     : WindowLink;
      ScreenPtr        : ^ScreenArray;
      FixedSize        : integer;
      WindowCount      : byte;

 procedure WindowBox (x1,y1,x2,y2 : byte);
 const Top    = 1;  Left     = 2;
```

```
      Right  = 3;  Bottom   = 4;
      UpLeft = 5;  UpRight  = 6;
      Loleft = 7;  Loright  = 8;
var   x,y  : byte;
      colorScr : array[1..25,1..80,1..2] of char absolute $B800:$0000;
      monoScr  : array[1..25,1..80,1..2] of char absolute $B000:$0000;
      crtType  : Byte absolute $0040:$0049;
begin
  Window(x1,y1,x2,y2);
  SetAttr(AttrOfWindow);
  clrscr;
  window(1,1,80,25);
  SetAttr(AttrOfBox);

    { top }
  Gotoxy (x1,y1);
  write (TypeOfBox[Boxstyle,Upleft]);
  for x := x1+1 to x2-1 do
    write(Typeofbox[boxstyle,top]);
  write (typeofbox[boxstyle,upright]);

    { side }
  for y := y1+1 to y2-1 do
    begin
      Gotoxy(x1,y); write(typeofbox[boxstyle,left]);
      Gotoxy(x2,y); write(typeofbox[boxstyle,right]);
    end;

    { bottom }
  Gotoxy(x1,y2);
  write(typeofbox[boxstyle,Loleft]);
  for x := x1+1 to x2-1 do
    write(typeofbox[boxstyle,bottom]);
  write(typeofbox[boxstyle,Loright]);

    { make it the current window and locate cursor to top }
  SetAttr(AttrOfHeader);
  Gotoxy ( (x1+x2-length(HeaderOfWindow))div 2,y1);
  write(HeaderOfwindow);
  for x:=x1+1 to x2+1 do
      if crtType<>7 then colorScr[y2+1,x,2]:=#7
                else monoScr[y2+1,x,2]:=#7;

  for y:=y1+1 to y2+1 do
      if crtType<>7 then colorScr[y,x2+1,2]:=#7
                else monoScr[y,x2+1,2]:=#7;

  window(x1+1,y1+1,x2-1,y2-1);
  SetAttr(AttrOfChar);
end;   { proc. boxwin }

procedure WindowOpen(x1,y1,x2,y2 : byte);
var  block       :  windowlink;
     Linelength,
     windowsize,
     I           :  integer;
     y           :  byte;
begin
```

```
  LineLength   := x2-x1+2;
  windowsize   := linelength*(y2-y1+2)*2 + FixedSize;

    { check window valid }
  if (x2 > 80) or (y2 > 25) or (x2-x1<2) or (y2-y1<2) then
     ErrorWindow  := 1
  else
    if (abs(MemAvail) < WindowSize) then
       ErrorWindow := 2
    else
       ErrorWindow := 0;
  if ErrorWindow = 0  then
     begin
       Getmem (Block,WindowSize);
       Block^.x1  :=  x1;
       Block^.x2  :=  x2;
       Block^.y1  :=  y1;
       Block^.y2  :=  y2;
       Block^.x   :=  whereX;
       Block^.y   :=  whereY;
       Block^.BackLink  := ActiveWindow;
       Activewindow     := Block;
       Windowcount      := windowcount + 1;
       block^.ID        := windowcount;
       I := 1;
       for y := y1 to y2+1 do
         begin
           Move(ScreenPtr^[y,x1],Block^.Screencontents[I],linelength*2);
           I := I + linelength;
         end;
       WindowBox(x1,y1,x2,y2);
     end;
end;

procedure FillWin(x1,y1,x2,y2,acii,attr:byte);
   type ScreenArray = array[1..25,1..80,1..2] of byte;
   var crtType : byte absolute $0040:$0049;
       Monoscr : ScreenArray absolute $B000:$0000;
       Colrscr : ScreenArray absolute $B800:$0000;
       row,col : integer;
   begin { Procedure Fillwin}
      directvideo:=true;
      checksnow:=false;
      for row := y1 to y2 do
         for col := x1 to x2 do begin
             if crtType<>7 then begin
                     Colrscr[row,col,1]:=acii;
                     Colrscr[row,col,2]:=attr;
                               end
                  else  begin
                     Monoscr[row,col,1]:=acii;
                     Monoscr[row,col,2]:=attr;
                               end;
             end;
      directvideo:=false;
      checksnow:=true;
   end;
```

```
Procedure WindowClose;
var  block      :  windowlink;
     linelength,
     windowsize,
     I          :  integer;
     y          :  byte;
begin
  if activewindow <> nil then
  begin
    block        := ActiveWindow;
    Linelength   := block^.x2-block^.x1+2;
    windowsize   := linelength*(block^.y2-block^.y1+2)*2 + fixedsize;
    windowcount  := windowcount - 1;
    I := 1;
    for y := block^.y1  to  block^.y2+1  do
      begin
        move(block^.screencontents[I],screenPtr^[y,block^.x1],linelength*2);
        I := I + linelength;
      end;
    ActiveWindow  := block^.Backlink;
    if activewindow = nil then
       window(1,1,80,25)
    else
       with activewindow^ do window(x1+1,y1+1,x2-1,y2-1);
    Gotoxy(block^.x,block^.y);
    FreeMem(block,Windowsize);
  end;
end;

procedure Initwin;
begin
  activewindow  := nil;
  fixedsize     := SizeOf(windowcontrolblock)-sizeof(screenblock);
  ScreenPtr     := Ptr(VideoSeg,0);
  Window (1,1,80,25);
  windowcount  := 0;
end;

procedure Setboxstyle (Attrib : byte);
begin
  boxstyle  := attrib;
end;

procedure SetWinHeader (st : string);
begin
  HeaderOfWindow  := st;
end;

procedure SetWinAttr (Attrib : byte);
begin
  AttrOfWindow  := Attrib;
end;

procedure SetBoxAttr(Attrib : byte);
begin
  AttrOfBox  := Attrib;
  end;
```

```
procedure SetHeadAttr(Attrib : byte);
begin
  AttrOfHeader  := Attrib;
end;

procedure SetCharAttr (Attrib : byte);
begin
  AttrOfchar  := Attrib;
end;

begin
  InitWin;
end. { of unit }
```

```
unit Keyboard;

interface

uses Crt,Dos,screen;
  var     Key      : char;
          FuncKey  : boolean;

  const   Return_Key = #13;    Esc_Key = #27;

          Shift_Tab  = #15;

          Alt_Q_key  = #16;  Alt_W_Key = #17;
          Alt_E_key  = #18;  Alt_R_Key = #19;
          Alt_T_key  = #20;  Alt_Y_Key = #21;
          Alt_U_key  = #22;  Alt_I_Key = #23;
          Alt_O_key  = #24;  Alt_P_Key = #25;

          Alt_A_key  = #30;  Alt_S_Key = #31;
          Alt_D_key  = #32;  Alt_F_Key = #33;
          Alt_G_key  = #34;  Alt_H_Key = #35;
          Alt_J_key  = #36;  Alt_K_Key = #37;
          Alt_L_key  = #38;

          Alt_Z_key  = #44;  Alt_X_Key = #45;
          Alt_C_key  = #46;  Alt_V_Key = #47;
          Alt_B_key  = #48;  Alt_N_Key = #49;
          Alt_M_key  = #50;

          F1_Key     = #59;   F2_Key     = #60;
          F3_Key     = #61;   F4_Key     = #62;
          F5_Key     = #63;   F6_Key     = #64;
          F7_Key     = #65;   F8_Key     = #66;
          F9_Key     = #67;   F10_Key    = #68;

          Home_Key   = #71;    Up_Key    = #72;
          PgUp_Key   = #73;    Lt_Key    = #75;
          Rt_Key     = #77;    End_Key   = #79;
          Dn_Key     = #80;    PgDn_Key  = #81;
          Ins_Key    = #82;    Del_Key   = #83;

          Shift_F1_Key   = #84;   Shift_F2_Key   = #85; { Same as F11..F20 }
          Shift_F3_Key   = #86;   Shift_F4_Key   = #87;
          Shift_F5_Key   = #88;   Shift_F6_Key   = #89;
          Shift_F7_Key   = #90;   Shift_F8_Key   = #91;
          Shift_F9_Key   = #92;   Shift_F10_Key  = #93;

          Ctrl_F1_Key    = #94;   Ctrl_F2_Key    = #95; { Same as F21..F30 }
          Ctrl_F3_Key    = #96;   Ctrl_F4_Key    = #97;
          Ctrl_F5_Key    = #98;   Ctrl_F6_Key    = #99;
          Ctrl_F7_Key    = #100;  Ctrl_F8_Key    = #101;
          Ctrl_F9_Key    = #102;  Ctrl_F10_Key   = #103;

          Alt_F1_Key     = #104;  Alt_F2_Key     = #105;{ Same as F31..F40 }
          Alt_F3_Key     = #106;  Alt_F4_Key     = #107;
          Alt_F5_Key     = #108;  Alt_F6_Key     = #109;
          Alt_F7_Key     = #110;  Alt_F8_Key     = #111;
```

```
        Alt_F9_Key     = $112;  Alt_F10_Key    = $113;

        Ctrl_PrtSc_Key  = $114;   Ctrl_Lt_Key    = $115;
        Ctrl_Rt_Key     = $116;   Ctrl_End_Key   = $117;
        Ctrl_PgDn_Key   = $118;   Ctrl_Home_Key  = $119;

        Alt_1_Key  = $120;        Alt_2_Key = $121;
        Alt_3_Key  = $122;        Alt_4_Key = $123;
        Alt_5_Key  = $124;        Alt_6_Key = $125;
        Alt_7_Key  = $126;        Alt_8_Key = $127;
        Alt_9_Key  = $128;        Alt_0_Key = $129;

        Ctrl_PguP_Key  = $132;
        F11_Key        = $133;   F12_Key        = $134;
        Shift_F11_Key  = $135;   Shift_F12_Key  = $136;
        Ctrl_F11_Key   = $137;   Ctrl_F12_Key   = $138;
        Alt_F11_Key    = $139;   Alt_F12_Key    = $140;

  procedure  ReadFunctionKey(var code : word);
  procedure  ReadFuncKey(var Key : char);
  procedure  Readarrowenterf10(var userkey : word);
  procedure  ReadString(var St:String);
  procedure  ReadIntNum(var Num,Code : integer);
  procedure  ReadRealNum(var Num : real; var code : integer);
  procedure  Readstringnum(var st : string;count : integer);
  procedure readint(var value : integer;count : integer;var code : integer);
  procedure  Readreal(var value : real;count : integer;var code : integer);
  function   Softblink(attr : byte ; count : word) : char;

{ End of interface section }

implementation

  function softblink(attr:byte;count:word):char;
  var
     invis : byte;
  begin
       invis := attr div 16 + (attr div 16) * 16;
       while not keypressed do
       begin
            setattrblink(invis,count);
            delay(500);
            setattrblink(attr,count);
            delay(500);
       end;
       softblink := readkey;
  end;

  procedure  ReadFunctionKey(var code : word);
  var REGS : registers;
  begin
    REGS.AH := $00;
    Intr ($16,REGS);
    code := REGS.AX;
  end;

  procedure  ReadFuncKey(var Key : char);
  begin
```

```
  Key := ReadKey;
  if Key = #0 then
     begin
       FuncKey := true;
       Key     := Readkey;
     end
  else
       FuncKey := false;
end;

procedure  Readarrowenterf10(var userkey : word);
var
     done : boolean;
begin
     done := false;
     while not done do
     begin
          readfunckey(key);
          userkey := ord(key) * 256;
          case userkey of
               20480 :  done := true;
               18432 :  done := true;
               17408 :  done := true;
                3328 :  done := true;
          else done := false;
          end;
     end;
end;

procedure  MapKey(var Key : char);
begin
  if FuncKey then
     case Key of
        Home_Key  : Key := ^A;  { Home }
        End_Key   : Key := ^Z;  { End  }
        Lt_Key    : Key := ^S;  { Lt Arrow }
        Rt_Key    : Key := ^D;  { Rt Arrow }
        Del_Key   : Key := ^G;  { Del  }
     else
        Key := #00; { Null }
     end;
end;

Procedure  BackSpace (var St : string; var Ptr:byte);
var Tail : string;
begin
  if Ptr <> 1 then
  begin
    Tail := Copy (St,Ptr,length(St)-Ptr+1);
    Delete (St,Ptr-1,1);
    Ptr  := Ptr-1;
    Write (^H);
    ClrEol;
    write (Tail);
    GotoXY( WhereX-Length(Tail),WhereY);
  end;
end;
```

```
procedure  LeftArrow (var Ptr:byte);
begin
  if Ptr <> 1 then
  begin
    Ptr := Ptr-1;
    write(^H);
  end;
end;

procedure  RightArrow(St : string; var Ptr:byte);
var Len : byte;
begin
  Len := Length(St);
  if (Ptr <= Len) and (Len<>0) then
  begin
    Ptr := Ptr+1;
    GotoXY(WhereX+1,WhereY);
  end;
end;

procedure  ToHome ( var Ptr:byte);
begin
  GotoXY(WhereX-Ptr+1,WhereY);
  Ptr:=1;
end;

procedure  ToEnd (St : string; var Ptr:byte);
var Len : byte;
begin
  len := Length(St);
  if Ptr <= Len then
  begin
     Tohome (Ptr);
     GotoXY(WhereX+Len,WhereY);
     Ptr := Len+1;
  end;
end;

procedure  TrunCate (var St : string; var Ptr : byte);
begin
  St  := copy (St,1,Ptr-1);
  Ptr := length(St)+1;
  if Ptr =1 then St :='';
  ClrEol;
end;

procedure  Del (var St:string; var Ptr:byte);
var Tail : string;
    Len  : byte;
begin
  Len := length(St);
  if (Ptr<=Len) and (Len<>0) then
  begin
    Tail := copy(St,Ptr+1,Length(St)-Ptr);
    Delete(St,Ptr,1);
    ClrEol;
    write(Tail);
    GotoXY(WhereX-Length(Tail),WhereY);
```

```
  end;
end;

procedure  Clear(var St : string; var Ptr : byte);
begin
  ToHome(Ptr);
  ClrEol;
  St := '';
end;

procedure  Character (var St : string; Ch : char; var Ptr:byte);
var P,X,Len : byte;
begin
  Len := Length(St);
  if Ptr> Len then
     begin
     if ( WhereX <= (Lo(WindMax)-Lo(windMin)) )  then
       begin
         St := St + Ch;
         Ptr := Ptr+1;
         write(ch);
       end;
     end
  else
    if (Len < (Lo(WindMax)-Lo(WindMin))) then
    begin
      X := WhereX;
      p := Ptr;
      Insert(Ch,St,Ptr);
      Ptr := Ptr+1;
      ToHome(P);
      write(St);
      GotoXY(X+1,WhereY);
    end;
  end;

procedure  ProcessFuncKey(   Key  : char;
                         var St   : string;
                         var Ptr  : byte;
                         var Quit : boolean);
begin
  case Key of
    ^M : begin if St = '' then St := #13; Quit := true; end;
    ^[ : begin St := #27; Quit := true; end;
    ^H : BackSpace(St,Ptr);
    ^G : Del(St,Ptr);
    ^S : LeftArrow(Ptr);
    ^D : RightArrow(St,Ptr);
    ^T : Truncate(St,Ptr);
    ^A : ToHome(Ptr);
    ^Z : ToEnd(St,Ptr);
    ^Y : Clear(St,Ptr);
    else
         if Key >=#32 then
            Character(St,Key,Ptr);
  end;
end;
```

```
procedure  ReadString(var St: string);
var Ch   : char;
    X,Y,
    Ptr  : byte;
    Quit : boolean;
begin
  Quit := false;
  Ptr  := 1;
  X    := WhereX;
  Y    := WhereY;
  write(St);
  GotoXY(X,Y);
  ReadFuncKey(Key);
  MapKey(Key);
  if Key < #32 then
       ProcessFuncKey(Key,St,Ptr,Quit)
     else
     begin
       St  := Key;
       Ptr := 2;
       GotoXY(X,Y);
             ClrEOl;
       write(St);
     end;
  while not(Quit) do
  begin
    ReadFuncKey(Key);
    MapKey(Key);
    ProcessFuncKey(Key,St,Ptr,Quit);
  end;
end;

procedure  ReadIntNum(var Num,Code : integer);
{
 Output   : Code =0  ---> Valid number, no error detected
            Code =1  ---> press only RETURN or ESC
            Code =255 --> Not a VAlid Number , Num is undefined }
var St : String;
begin
  Str(Num,St);
  ReadString(St);
  val(St,Num,Code);
  if Code <> 0 then Code := 255;
  if (St = #27)or(St = #13) then Code := 1;
end;

procedure ReadRealNum(var Num : real; var code : integer);

var St : string;
begin
  Str(Num:1:12,St);
  { Trim Rigth Zero }
  while St[Length(St)] ='0' do St[0] := Chr(ord(St[0])-1);
  ReadString(St);
  val(St,Num,Code);
  if code <> 0 then Code := 255;
  if (St=#27) or (St=#13) then Code := 1;
```

```
  end;

procedure readstringnum(var st : string;count : integer);
var
   ch     : char;
   key    : word;
   strg   : string;
   ptr    : integer;
   quit   : boolean;

procedure delete;
begin
     if ptr <> 1 then
     begin
          ptr := ptr - 1;
          write(ch);
          write(' ');
          write(ch);
          strg := copy(strg,1,length(strg) - 1);
     end
     else
         write(^g);
end;

procedure character;
begin
     if ptr <> count then
        write(^g)
     else
         begin
             write(ch);
             ptr := ptr + 1;
             strg := strg+ch;
         end;
end;

begin
     ptr  := 1;
     strg := '';
     quit := false;
     repeat
           readfunckey(ch);
           key := ord(ch) * 256;
           case key of
                256  : begin strg := #27; quit := true; end;
               3328  : begin if strg = '' then strg := #13; quit := true; end;
              21248  : begin ch := ^h; delete; end;
           end;
           if (ch in ['0'..'9']) then character;
           if not ((ch in ['0'..'9']) or (key = 256) or
              (key = 3328) or (key = 21248)) then
               write(^g);
     until quit;
     st := strg;
end;

procedure readint(var value : integer;count : integer;var code : integer);
```

```
var
   ch      , : char;
   key       : word;
   strg      : string;
   ptr       : integer;
   quit      : boolean;

procedure delete;
begin
     if ptr <> 1 then
     begin
          ptr := ptr - 1;
          write(ch);
          write(' ');
          write(ch);
             strg := copy(strg,1,length(strg) - 1);
     end
     else
         write(^g);
end;

procedure number;
begin
     if ptr > count then
        write(^g)
     else
         begin
              write(ch);
              ptr := ptr + 1;
              strg := strg+ch;
         end;
end;

begin
     ptr  := 1;
     strg := '';
     quit := false;

     repeat
           readfunckey(ch);
           key := ord(ch) * 256;
           case key of
                256  : begin strg := #27; quit := true; end;
               3328  : quit := true;
              21248  : begin ch := ^h; delete; end;
           end;
           if (ch in ['0'..'9']) then number;
           if not ((ch in ['0'..'9']) or (key = 256) or
              (key = 3328) or (key = 21248)) then
              write(^g);
     until quit;
     val(strg,value,code);
     if code <> 0 then code := 255;
     if (strg = #27) or (strg = '') then code := 1;
end;

procedure readreal(var value : real;count : integer;var code : integer);
```

```
var
   decimalflag  : boolean;
   ch           : char;
   key          : word;
   strg         : string;
   ptr          : integer;
   twopoint     : integer;
   quit         : boolean;

procedure delete;
begin
     if ptr <> 1 then
     begin
          ptr := ptr - 1;
          write(ch);
          write(' ');
          write(ch);
          if strg[ptr] = '.' then decimalflag := true;
          strg := copy(strg,1,length(strg) - 1);
     end
     else
         write(^g);
end;

procedure point;
begin
     if decimalflag then
     begin
          write(ch);
          ptr := ptr + 1;
          strg := strg+ch;
          twopoint := length(strg);
          decimalflag := false;
     end;
end;

procedure number;
begin
     if ptr > count then
          write(^g)
     else
     begin
          write(ch);
          ptr  := ptr + 1;
          strg := strg+ch;
     end;
end;
begin
     ptr   := 1;
     twopoint := 1;
     strg  := '';
     value := 0.0;
     decimalflag := true;
     quit        := false;
     repeat
          readfunckey(ch);
          key := ord(ch) * 256;
          case key of
```

```
            256  : begin strg := #27; quit := true; end;
           3328  : quit := true;
          21248  : begin ch := ^h; delete; end;
        end;
        if ch = '.' then point;
        if (ch in ['0'..'9']) then number;
        if not ((ch in ['0'..'9']) or (ch = '.') or (key = 256) or
           (key = 3328) or (key = 21248)) then
            write(^g);
        if decimalflag then
        begin
            if length(strg) > 3 then
            if not (ch = '.') then
            begin
               ch := ^h;
               delete;
               write(^g);
            end;
        end
        else
        if (length(strg) > (twopoint + 2)) then
        begin
            ch := ^h;
            delete;
            write(^g);
        end;
    until quit;
    if strg[1] = '.' then strg := '0' + strg;
    if strg[length(strg)] = '.' then strg := strg + '0';
    val(strg,value,code);
    if code <> 0 then code := 255;
    if (strg = #27) or (strg = '') then code := 1;
end;

end.  { of Unit }
```

```
unit screen;

interface

uses Crt,Dos;

    Const NoDisplay     = $00;
          LowDisplay    = $07;
          HighDisplay   = $0F;
          UnderLineLow  = $01;
          UnderLineHigh = $09;
          ReverseLow    = $70;
          ReverseHigh   = $78;
          BlinkLow      = $87;
          BlinkHigh     = $8F;
          UndBlinkLow   = $81;
          UndBlinkHigh  = $89;
          RevBlinkLow   = $F0;
          RevBlinkHigh  = $F8;
          ColorSeg      = $B800;
          MonoSeg       = $B000;
    Var   VideoSeg      : word;
          CrtType       : byte Absolute $0040:$0049;
          CursorMode    : word Absolute $0040:$0060;
          Vport         : word Absolute $0040:$0063;

    procedure Setattr (Attrib : byte) ;
    procedure SetattrBlink (data : Byte; Count : Word);
    procedure SetCursor (Top,Bottom : byte) ;
    procedure Cursoron;
    procedure CursorOff;
    procedure Directvideooff;
    procedure Directvideoon;

  { End of inteface section }

  implementation

    var Regs : registers;

    procedure SetAttr ( Attrib : byte);
    begin
      TextAttr := Attrib;
    end;

    procedure SetattrBlink (data : Byte; Count : Word);
    var
        offset : word;
        scrptr : ^byte;
    begin
         offset := ((wherey-1)*80+wherex-1)*2+1;
         scrptr := ptr(videoseg,offset);
         while (count <> 0) do
         begin
              scrptr^ := data;
              inc(scrptr,2);
              dec(count);
```

```
    end;
end;

procedure SetCursor (Top,Bottom : byte);
begin
  Regs.AH := 1;  { function set cursor mode }
  Regs.CH := Top;
  Regs.Cl := Bottom;
  intr($10,Regs);
end;

procedure CursorOn;
begin
  Port[Vport]   := 10;
  Port[Vport+1] := Hi(CursorMode) and $DF;
  Port[Vport]   := 11;
  Port[Vport+1]  := Lo(CursorMode);
end;

procedure CursorOff;
begin
  Port[Vport]   := 10;
  Port[Vport+1] := Hi(CursorMode) or $20;
  Port[Vport]   := 11;
  Port[Vport+1]  := Lo(CursorMode);
end;

procedure directvideooff;
begin
    directvideo := true;
    checksnow   := false;
end;
procedure directvideoon;
begin
    directvideo := false;
    checksnow   := true;
end;

procedure IdentifyCrt;
begin
  case CrtType of
    0..3      : VideoSeg := ColorSeg;
      7       : VideoSeg := MonoSeg;
  end;
end;

begin
  IdentifyCrt;
end. { of unit }
```

กิตติกรรมประกาศ

ขอขอบพระคุณ อ.ภากร หุตสังกาศ เป็นอย่างสูง ที่ได้ให้การประสิทธิ์ประสาทวิชา ให้คำแนะนำปรึกษา ในเรื่องต่างๆ แก่คณะผู้จัดทำ ขอขอบพระคุณอาจารย์ทุกท่านที่สอนและให้ความรู้ มาตั้งแต่เริ่มเข้ารับการศึกษา ส่งผลให้การทำปริญญานี้ สำเร็จลุล่วงได้ด้วยดี

เอกสารอ้างอิง

1. สุรศักดิ์ สงวนพงษ์, "เทคนิคการเขียนโปรแกรมขั้นสูง", บริษัทซีเอ็ดยูเคชั่น จำกัด,2521
2. ผศ.นุกูล กระจาย,"การเขียนโปรแกรมและประมวลผลข้อมูลด้วยเทอร์โบปาสคาล", บริษัทซีเอ็ดยูเคชั่น จำกัด, 2521
3. John G. Bolliner Neil A. Duffie, "Computer Control of Machines and Process", Addison-Wesley Publishing Company Inc,1988
4. IBM, "IBM Technical Reference Manual For IBM PC-XT", IBM Co.Ltd.,1983
5. Michael Tischer, "Turbo Pascal System Programming", Abacus,1991
6. Turbo Pascal User's Guide Reference Guide Version 6.0