ชุด DIGITAL TRAINER ควบคุมโดย MICRO COMPUTER

อำนวย คงเกียรติไกร

สมบูรณ์ โชคชัยพรรักษ์

ปริญญานิพนธ์ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร ปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาวิชา อิเล็คทรอนิคส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหารลาดกระบัง

2534

ชุด DIGITAL TRAINER ควบคุมโดย MICRO COMPUTER

อำนวย คงเกียรติไกร

สมบูรณ์ โชคชัยพรรักษ์

ได้รับพิจารณาอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตร ปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาวิชา อิเล็คทรอนิคส์

คณะกรรมการตรวจสอบ ปริญญานิพนธ์

_____ประธานกรรมการ
(ผศ. นิกร สุขุตมตันติ)

_____กรรมการ
( )

_____กรรมการ
( )

_____กรรมการ
( )

_____กรรมการ
( )

วันที่ 1 เดือน มีนาคม พ.ศ. 2534

ชุด DIGITAL TRAINER ควบคุมโดย MICRO COMPUTER

อำนวย คงเกียรติไกร

สมบูรณ์ ไชคชัยพรรักษ์

ผศ. นิกร สุขุตมตันติ
ปีการศึกษา 2534

บทคัดย่อ

ในการสร้างชุด DIGITAL TRAINER ขึ้นมาด้วยจุดประสงค์เพื่อที่จะใช้เป็นเครื่องมือ ในการเริ่มต้นศึกษาค้นคว้าทางด้าน DIGITAL ELECTRONIC โดยจะสามารถใช้งานชุด DIGITAL TRAINER อย่างสะดวกและลดปัญหาต่างๆ ทุกขั้นตอน อย่างไรก็ตามสำหรับชุด DIGITAL TRAINER นี้ยังเป็นเพียงเครื่องต้นแบบ ที่สามารถพัฒนาต่อไปทั้งด้าน SOFTWARD และ HARDWARE ได้อีก

ทางผู้จัดทำหวังว่ารายงานการวิจัยเล่มนี้จะทำให้ผู้ที่ต้องการศึกษาค้นคว้าเกิดความเข้าใจในรายละเอียดที่ต้องการ เพื่อสามารถนำไปใช้ให้เกิดประโยชน์ต่อไป.

ผู้จัดทำ

สารบัญ

บทนำ

ปัจจุบันการศึกษาทางด้าน DIGITAL นับว่าเป็นสิ่งจำเป็นอย่างยิ่ง เราจะเห็นว่าเทคโนโลยี หรืออุปกรณ์ทางด้านไฟฟ้าหรืออิเลคทรอนิคส์สมัยใหม่มักจะมีส่วนประกอบทางด้าน DIGITAL เสมอ ในการเริ่มต้นศึกษาทางด้าน DIGITAL เราจะแบ่งออกเป็น 2 ส่วน ที่สำคัญคือ ทางด้านทฤษฎี และทางด้านปฏิบัติ

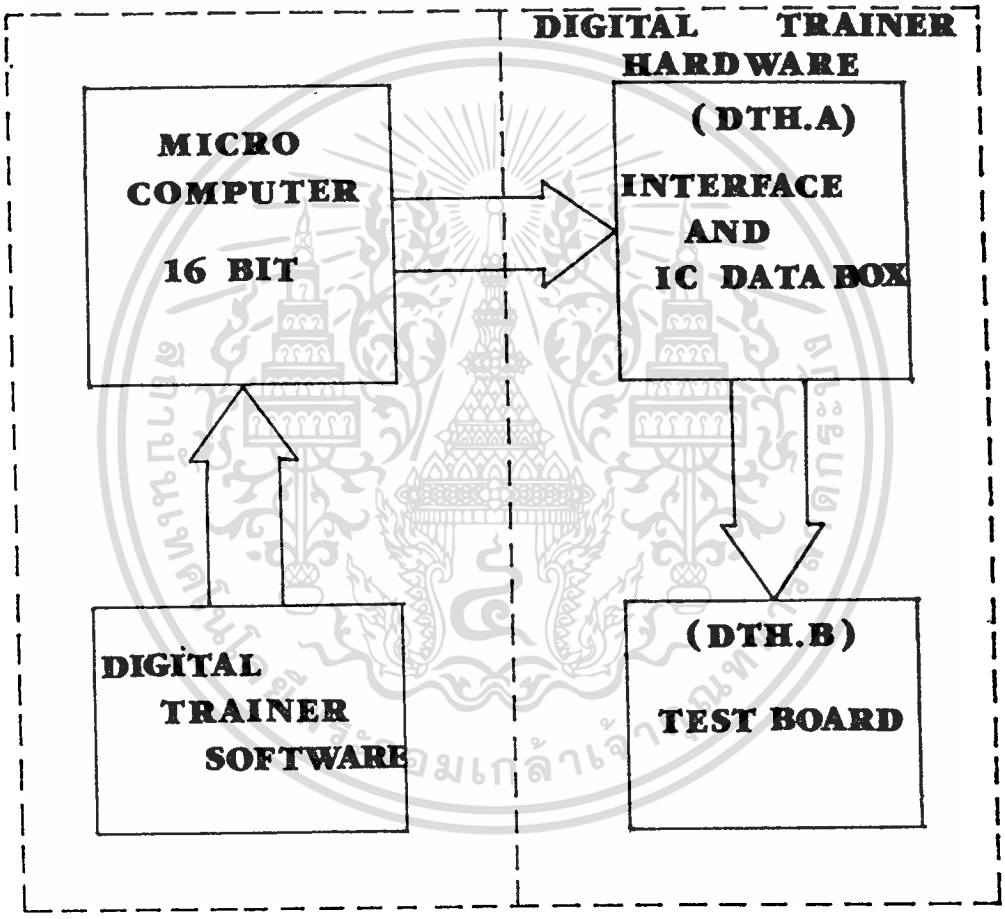
- <u>ทางทฤษฎี</u> จะเป็นการศึกษาเกี่ยวกับอุปกรณ์พื้นฐานทางด้าน DIGITAL ไม่ว่าจะเป็นคุณสมบัติต่าง ๆ ในการศึกษาทางด้านทฤษฎี อุปกรณ์ที่จำเป็นคือตำราต่าง ๆ และข้อมูลเฉพาะของอุปกรณ์แต่ละตัว
- <u>ทางด้านปฏิบัติ</u> ในการศึกษาทางด้าน DIGITAL ให้ได้ผลจริง ๆ จำเป็นที่จะต้องมีการทดลอง หรือตรวจสอบการทำงานของอุปกรณ์ต่าง ๆ หรือวงจรทางด้าน DIGITAL เพื่อทำการเปรียบเทียบกับทางทฤษฎีที่ได้ศึกษา

ในการสร้างชุด DIGITAL TRAINER มีจุดหมายที่จะนำส่วนประกอบที่สำคัญในการเริ่มต้นศึกษาทางด้าน DIGITAL เข้ามารวมไว้ด้วยกัน เพื่อความสะดวกในการศึกษาค้นคว้า หรือการนำไปใช้งานจริง ๆ

คุณสมบัติ และการนำไปใช้งาน ของชุด DIGITAL TRAINER

- ใช้ในการศึกษาคุณสมบัติเบื้องต้นของ IC DIGITAL เช่น OR, AND, NAND, NOR, NOT บน COMPUTER PC 16 BIT สามารถแสดงรายละเอียดของ IC เบอร์ต่าง ๆ ตระกูล TTL ที่สำคัญได้
- สามารถเลือก IC TTL เบอร์ใดเบอร์หนึ่งพร้อมกันหรือต่างเบอร์กันครั้งละ 2 ตัว เพื่อใช้ในการศึกษาทดลองบนส่วนของ HARDWARE
- ส่วนของชุด HARDWARE มีอุปกรณ์ช่วยเหลือในการทดลอง เช่น เครื่องกำเนิดสัญญาณสี่เหลี่ยม, แหล่งจ่ายไฟ , สวิทช์แบบต่าง ๆ
- สามารถดัดแปลงเป็นเครื่องตรวจเช็ค IC ตามคุณสมบัติที่มีอยู่
- สามารถใช้เป็นชุดทดลองในการเรียนการสอนทางด้าน DIGITAL
- สามารถนำเครื่องต้นแบบพัฒนา IC เบอร์อื่น ๆ เพิ่มขึ้นได้อีก

BLOCK DIAGRAM OF SYSTEM



รูปที่ 1 แสดง BLOCK DIAGRAM ของ DIGITAL TRAINER

<u>รายละเอียดของส่วนประกอบต่าง ๆ</u>

- DIGITAL TRAINER SOFTWARE (DTS)

DTS ประกอบด้วย PROGRAM ต่าง ๆ ที่ถูกสร้างขึ้นเป็นระบบ UNIX โดยใช้ภาษา PASCAL และภาษา ASSEMBER หน้าที่สำคัญของ DTS มีหน้าที่สำคัญ 2 อย่าง คือ

1. แสดงรายละเอียดต่าง ๆ ของข้อมูลที่สร้างขึ้นทั้งตัวอักษรและรูปภาพต่าง ๆ
2. ใช้ในการควบคุมส่วน DIGITAL TRAINER HARDWARE SOFTWARE ของ DIS นั้นจัดว่าเป็น SOFTWARE ประเภท APPLICATION SOFTWARE

- DIGITAL TRAINER HARDWARE (DTH)

DTH เป็นส่วนของ HARDWARE ซึ่งประกอบด้วยส่วนสำคัญ 2 ส่วนคือ

1. DTH_A คือส่วนที่ใช้ในการ INTERFACE กับเครื่อง COMPUTER ที่ควบคุมจาก DTS ในส่วนนี้ยังเก็บข้อมูลของ IC ต่าง ๆ เก็บไว้
2. DTH_B เป็นส่วนของแผงทดลองที่มีส่วนของ SLOT อยู่ 2 SLOT คือ SLOT#1 ; SLOT#2 โดยส่วนของ SLOT ทั้ง 2 นี้จะเชื่อมมาจากส่วน DTH_A ส่วนประกอบอื่นบนแผงทดลอง DTH_B นี้จะประกอบด้วยอุปกรณ์ที่สำคัญที่ใช้ในการทดลอง เช่น SUPPLY, CLOCK, SW, DISPLAY, BOARD ทดลองเอนกประสงค์

<u>MICROCOMPUTER PC 16BIT</u>

ส่วนนี้เป็นส่วนที่ใช้รวมกับชุด DIGITAL TRAINER โดยสามารถใช้กับเครื่อง COMPUTER 16 BIT ทั่วไปทุกเครื่องโดยบางรุ่นอาจต้องมีการดัดแปลงในส่วนของการเชื่อมโยงเล็กน้อย

สำหรับโครงงานนี้จะแบ่งการสร้างออกเป็น 2 ส่วนคือ ส่วนของ SOFTWARE และ HARDWARE ในส่วนแรกจะกล่าวถึงส่วนของ SOFTWARE ก่อน และในส่วนหลังจะกล่าวถึงส่วนของ HARDWARE

DIGITAL TRAINER SOFTWARE
(DTS)

บทที่ 1
การเขียน PROGRAM ด้วยภาษา PASCAL

การสร้าง DIGITAL TRAINER SOFTWARE นั้นถูกสร้างขึ้นจากภาษา PASCAL และบางส่วนมีการเชื่อมโยงกับภาษาอื่นตามความเหมาะสมในการทำงาน เช่น การเชื่อมโยงกับภาษา ASSEMBLY

ภาษา PASCAL เป็นภาษาระดับสูงภาษาหนึ่งที่พัฒนาขึ้นโดยอาศัยภาษา ALGOL 60 เป็นหลัก ซึ่งมีข้อดีดังนี้

- ดีในเรื่องของคำสั่งควบคุม เพราะมีหลายระดับ และมีความเหมาะสมกับงานลักษณะต่าง ๆ ที่ผู้เขียนโปรแกรมสามารถเลือกใช้ตามความเหมาะสม คำสั่งควบคุมนั้นทำให้โปรแกรมที่เขียน โดยภาษา PASCAL มีโครงสร้างที่กระทัดรัดมองเห็นขั้นตอนการทำงานได้ง่าย
- ดีในด้านโครงสร้างข้อมูล เพราะมีแบบข้อมูลให้ใช้มากทั้งแบบที่กำหนดไว้แล้วหรือจะเป็นแบบที่ผู้ใช้สามารถกำหนดขึ้นเองได้ตามความเหมาะสม
- ดีในด้านโครงสร้างของโปรแกรม เพราะลักษณะโครงสร้างโปรแกรม ภาษา PASCAL เป็นแบบบล็อค หรือโมดูล ทำให้โปรแกรมมีโครงสร้างที่ดี สั้น และง่ายต่อการศึกษาถึงการทำงานของโปรแกรม

ภาษา PASCAL
และลักษณะโปรแกรมภาษาปาสคาล

ภาษาปาสคาลเป็นภาษาระดับสูง ที่มีโครงสร้างของภาษาอังกฤษแต่มีข้อกำหนดเข้มงวดมากกว่า เพื่อหลีกเลี่ยงความยุ่งยาก และเพื่ออำนวยความสะดวกในการแปลภาษาจากภาษาระดับสูงเป็นภาษาเครื่อง (รหัสเลขฐานสอง) จึงมีการกำหนดไวยกรณ์ของภาษาไว้อย่างเคร่งครัด

ในการเขียนโปรแกรม บางครั้งก็ต้องการความคล่องแคล่ว และมีหลักการแต่ก็จะต้องมีกฎเกณฑ์ที่เข้มงวดด้วย ทุก ๆ คำสั่งในโปรแกรมภาษาปาสคาลจะต้องเป็นไปตามกฎของไวยกรณ์ภาษาปาสคาล ถ้าหากมีคำสั่งใดที่ไม่เป็นไปตามกฎที่กำหนดไว้ จะเป็นเหตุให้เกิดความผิดพลาดโดยไม่มีการยกเว้น ดังนั้นจึงเป็นความจำเป็นที่จะต้องทำความเข้าใจ และยึดหลักของไวยกรณ์อย่างเคร่งครัด แม้จะมีจุลภาคหรือจุดอยู่ผิดที่เพียงอันเดียวก็เป็นเหตุให้โปรแกรมผิดพลาดได้

1. <u>อักขระในภาษาปาสคาล</u>

การเขียนโปรแกรมภาษาปาสคาล จะต้องมีการนำเอาอักขระต่าง ๆ มาเรียงต่อกันให้มีความหมายตามต้องการ และอักขระที่จะนำมาใช้ในภาษาปาสคาลมีอยู่ด้วยกัน 3 ชนิด คือ

1. ตัวอักษร ประกอบด้วยตัวอักษรในภาษาอังกฤษทั้งหมด รวมทั้ง @ ด้วย ดังนี้

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	a	b	c	d
e	f	g	h	i	j	k	l	m	n
o	p	q	r	s	t	u	v	w	x
y	z	@							

2. ตัวเลข คือตัวเลขในระบบเลขฐานสิบ ดังนี้

0	1	2	3	4	5	6	7	8	9

และถ้าต้องการใช้เป็นระบบเลขฐานสิบหก จะต้องมีอักษรต่อไปนี้รวมอยู่ด้วย

A	B	C	D	E	F

3. สัญลักษณ์พิเศษ ได้แก่พวกเครื่องหมายต่าง ๆ มีทั้งหมดด้วยกัน 33 ตัว ดังนี้

!	"	#	$	%	&	'	(	)	=	
~	¦	-	^	\	`	{	@	[	+	
}	;	:	]	<	>	?	_	.	/	,

อย่างไรก็ตาม ในการสั่งงานคอมพิวเตอร์ จะต้องมีอักขระหลาย ๆ ตัว หรืออาจเป็นกลุ่มของอักขระที่มีความหมายพิเศษในตัวเอง ซึ่งจะใช้ในกรณีเฉพาะเท่านั้น สำหรับในภาษาบาสคาล มีอยู่ 2 แบบ คือ

1. <u>สัญลักษณ์เฉพาะ</u> สัญลักษณ์เฉพาะในภาษาบาสคาลมีดังแสดงในรูปที่ 1.1

+	-	*	/	:=	.	,	;	:
=	<>	<	<=	>=	>	(	)	[
]	{	}	(*	*)	^	'	blank	

รูปที่ 1.1 สัญลักษณ์เฉพาะ

สัญลักษณ์เฉพาะเหล่านี้ส่วนมากจะใช้เป็น operators ในนิพจน์ เช่น +, -, *, /, ฯลฯ ส่วนสัญลักษณ์อื่น ๆ นั้น ใช้สำหรับการเป็นรูปแบบเฉพาะในคำสั่งต่าง ๆ เช่น ";" ใช้ในการแยกประโยค 2 ประโยคจากกัน, ใช้แยกประโยค หรือ (* หรือ { กับ *) หรือ }ใช้ในเมื่อต้องการเขียนประโยคหมายเหตุ โดยเขียนไว้ในวงเล็บ

2. <u>คำเฉพาะ (Reserved Words)</u> หมายถึง คำที่มีความหมายเฉพาะตัวอยู่แล้วไม่จำเป็นต้องกำหนดความหมายใหม่โดยผู้เขียนโปรแกรม สำหรับคำเฉพาะในภาษาบาสคาลมีดังแสดงในรูปที่ 1.2

คำเฉพาะบางคำจะถูกใช้เป็น operator เช่น AND, OR, NOT, DIV ส่วนคำอื่น ๆ นั้นเป็น declarations หรือ definitions เช่น PROGRAM, CONST, VAR, TYPE นอกนั้น ใช้เป็นส่วนหนึ่งของคำสั่ง เช่น IF, WHILE, REPEAT เป็นต้น

AND	FILE	NIL	SET
ARRAY	FOR	NOT	THEN
BEGIN	FORWARD	OF	TO
CONST	FUNCTION	OR	TYPE
CASE	GOTO	PACKED	UNTIL
DO	IF	PROCEDURE	VAR
DOWNTO	IN	PROGRAM	WHILE

ELSE	LABEL	RECORD	WITH
END	MOD	REPEAT	

รูปที่ 1.2 คำเฉพาะในภาษาปาสคาล

คำเฉพาะนี้ผู้เขียนโปรแกรมจะต้องไม่นำไปใช้ให้มีความหมายในทางอื่นโดยเด็ดขาด เช่น ชื่อของโปรแกรมจะชื่อว่า "PROGRAM" ไม่ได้

2. IDENTIFIERS

Identifier คือชื่อที่ตั้งขึ้น ซึ่งอาจจะเป็นชื่อของตัวแปร, ชื่อโปรแกรม, ชื่อข้อมูล , ชื่อค่าคงที่ , ชื่อฟังก์ชั่น หรือชื่อโปรซีเดอร์ (Procedure) ก็ได้ Identifier มีอยู่ด้วยกัน 3 ชนิดคือ :-

- คำเฉพาะ (Reserved Word)
- Standard Identifiers
- User-defined Identifiers

สำหรับคำเฉพาะได้กล่าวไปแล้ว ฉะนั้นในที่นี้จะกล่าวถึง Standard Identifiers และ User-defined Identifier เท่านั้น

STANDARD IDENTIFIERS

Standard Identifiers คือ คำที่กำหนดความหมายมาก่อนแล้ว แต่ผู้เขียนโปรแกรม สามารถกำหนดความหมายใหม่ได้ให้แตกต่างจากเดิม ซึ่งคำเหล่านี้มีอยู่ดังรูปที่ 1.3

สำหรับผู้ที่เพิ่งเริ่มหัดเขียนโปรแกรมใหม่ ๆ ไม่ควรที่จะทำการเปลี่ยนความหมายของคำใหม่ การเปลี่ยนความหมายของคำใหม่ ควรจะทำโดยผู้ที่เชี่ยวชาญ ในการเขียนโปรแกรมดีกว่า และจะใช้ในกรณีเฉพาะเท่านั้น เช่น เมื่อคำ Standard Identifier นั้นไม่จำเป็นต้องใช้ในโปรแกรมนั้น ๆ

ในการปฏิบัตินั้น Standard Identifier จะต้องถูกกระทำ เช่นเดียวกับคำเฉพาะ เว้นเสียแต่ว่ามีเหตุผลที่ดีพอที่จะเปลี่ยนให้เป็นอย่างอื่น

```
FILES  :
     INPUT      OUTPUT
CONSTANTS  :
     FALSE      TRUE       MAXINT
TYPES   :
     BOOLEAN    CHAR       INTEGER    REAL     TEXT
FUNCTIONS  :
     ABS        EOF        ODD        SIN      TRUNC
     ARCTAN     EOLN       ORD        SQR
     CHR        EXP        PRED       SQRT
FUNCTIONS  :
     COS        LN         ROUND      SUCC
PROCEDURES  :
     GET        PAGE       READLN     UNPACK
     NEW        PUT        RESET      WRITE
     PACK       READ       REWRITE    WRITELN
```

รูปที่ 1.3 Standard Identifiers

User-defined Identifiers

สำหรับ Identifiers ชนิดนี้เป็นชนิดที่ผู้เขียนโปรแกรมกำหนดขึ้นเองเพื่อใช้งานในการเขียนโปรแกรม โดยในที่นี้จะเรียก Identifier เฉย ๆ เช่น การตั้งชื่อโปรแกรม การตั้งชื่อตัวแปร หรือชื่อฟังก์ชั่น เป็นต้น ซึ่งการกำหนดขึ้นมาใช้เองจะต้องอยู่ภายใต้กฎเกณฑ์ ต่อไป

1. ต้องขึ้นต้นด้วยตัวอักษรเท่านั้น
2. จะต้องให้มีความยาวเท่าใดก็ได้แต่จะมีเพียง 8 ตัวแรกเท่านั้น ที่เครื่องจะไปเป็นชื่อที่แท้จริง
3. ต้องไม่ใช่คำเฉพาะ
4. จะประกอบด้วยตัวอักษร และตัวเลขเท่านั้น จะสัญลักษณ์พิเศษไม่ได้ยกเว้น "
5. จะมีช่องว่างระหว่างชื่อ ๆ หนึ่งไม่ได้

3. รูปแบบโปรแกรมภาษาปาสคาล

ภาษาปาสคาลเป็นภาษาที่มีรูปแบบเป็นโปรแกรมโครงสร้าง (Structure programming) และแยกโปรแกรมออกเป็นส่วน ๆ คือ เป็นโมดูล (modular programming) ดังนั้นแต่ละขั้นตอนของวิธีการแก้ปัญหาจึงสามารถแบ่งออกเป็นโมดูลได้ และเนื่องจากไวยกรณ์ของภาษาปาสคาล จำเป็นต้องมีการกำหนดไว้ในตอนต้นโปรแกรมก่อน ดังนั้นรูปแบบของโปรแกรมจึงเขียนได้ดังรูป 1.4

```
                PROGRAM ...

                  Declarations
                  Definitions
PROGRAM
BLOCK           BEGIN ...
                    Program
                    Body
                END.
```

รูปที่ 1.4 โครงสร้างโปรแกรมภาษาปาสคาล

และถ้าจะพิจารณาแต่ละส่วนโดยละเอียด จะมีลักษณะดังรูป 1.5

```
                PROGRAM HEADING                    PROGRAM
                   EXAMPLE  :  PROGRAM SUM         HEADER


                LABEL DECLARATION                  LABLE
                  เพื่อใช้ในการ JUMP โดยใช้ GOTO   DECLARATION


                CONSTANT DEFINITION                DATA
DECLARATIONS    TYPE DEFINITION                    DESCRIPTION
                VARIABLE DECLARATION


                PROCEDURE  DECLARATION             โปรแกรมย่อย
                FUNCTION DECLARATION


                BEGIN
                  STATEMENT  (S);
                    EXAMPLE:TOTAL:=A + B;
                      BEGIN
MAIN                    STATEMENT (S);             โปรแกรมหลัก
BODY                  END;
                      BEGIN
                        STATEMENT (S);
                      END;
                  STATEMENT  (S);
                END.
```

รูปที่ 1.5 โครงสร้างโปรแกรมภาษาปาสคาลอย่างละเอียด

3.1 PROGRAM HEADING

เป็นส่วนที่ใช้สำหรับการตั้งชื่อโปรแกรมสำหรับโปรแกรมนั้น ๆ จะเป็นชื่ออะไรก็ได้ที่เป็นชื่อที่ถูกต้องตามหลักการตั้งชื่อ ภายใต้ส่วนนี้ผู้เขียนต้องบอกด้วยว่าใช้ไฟล์ข้อมูลภายนอกอะไรบ้าง ไฟล์ที่ใช้โดยปกติเป็น Input file และ Output file ซึ่งปกติจะหมายถึง แป้นพิมพ์ (Keyboard) และจอภาพ (Cathode Ray Tube) ซึ่งสามารถละไว้ได้

ตัวอย่าง

```
PROGRAM SUM (Input, Output);
```

โปรแกรมนี้กำหนดว่าให้ชื่อ SUM ใช้ไฟล์ Input และ Output ซึ่งถ้าต้องการละการกำหนดไฟล์ สามารถเขียนได้ดังนี้

```
PROGRAM SUM;
```

3.2 DECLARATIONS

ในภาษาปาสคาลมีส่วนของ Declaration อยู่หลายชนิด ซึ่งจะต้องปรากฏตามที่กำหนดไว้ในรูป 1.5 แต่ในส่วนนี้ก็สามารถละได้ถ้าไม่จำเป็นต้องมี อย่างเช่นในตัวอย่างโปรแกรมต่อไปนี้

```
PROGRAM  GREETING;
(* THIS IS A SAMPLE PASCAL PROGRAM *)
BEGIN
    WRITE ('HELLO')
END.
```

จะเห็นว่าโปรแกรมข้างต้น ไม่มีส่วน declarations มีเฉพาะส่วนหัว และ mainbody เท่านั้น และสำหรับข้อความที่อยู่ในเครื่องหมาย (* กับ *) หมายถึง ข้อความหมาย (COMMENT) ซึ่งเครื่องจะไม่สนใจ

แต่ตามปกติเวลาเขียนโปรแกรมจริง ๆ แล้ว อย่างน้อยที่สุดจะต้องมีการใช้ตัวแปร และถ้ามีการใช้ตัวแปรเมื่อไร ก็จะต้องใช้ส่วน declaration เมื่อนั้น ดังตัวอย่างต่อไปนี้

```
PROGRAM SUM
VAR A, B, TOTAL : INTEGER;
BEGIN
   WRITELN ('ENTER TWO NUMBERS TO BE ADDED...');
   REAE (A, B);
   TOTAL := A + B;
   WRITELN ('THE SUM OF' , A, 'AND', B, 'IS',TOTAL)
END.
```

ในโปรแกรมข้างต้นนี้ ประกอบด้วย ส่วนหัว, ส่วน declaration และ mainbody

ในการทำงานของส่วน declaration จะเป็นการบอก compiler ว่าต้องการกำหนดอะไรบ้าง จากรูป 1.5 นั้นจะเห็นว่ามีส่วนของ declaration อยู่ 5 ส่วนด้วยกัน ซึ่งถ้าส่วนใดที่ไม่ต้องการใช้ก็ไม่ต้องเขียน เช่น ส่วน LABEL นาน ๆ จะใช้สักครั้ง และ FUNCTION และ PROCEDURE จะใช้ในกรณีที่โปรแกรมยาวมาก ๆ เท่านั้น จะมีอยู่เพียง 3 ส่วนที่เหลือเท่านั้นที่ใช้บ่อย ๆ ซึ่งก็กำหนดรูปแบบในหัวข้อที่ผ่านมาแล้ว

3.3 PROGRAM BODY

ส่วนนี้บางครั้งจะเรียกโปรแกรมหลัก ซึ่งประกอบด้วยลำดับของคำสั่งต่าง ๆ ซึ่งจะใช้เพื่อทำการแก้ปัญหาตามที่กำหนด และคำสั่งทั่ว ๆ ไปมีอยู่มากมายหลายคำสั่งที่ใช้ได้ในภาษาปาสคาล ซึ่งจะกล่าวรายละเอียดต่อไป

สำหรับในส่วนนี้ จะขึ้นต้นด้วยคำว่า BEGIN และลงท้ายด้วย END. (หลัง END. ต้องตามด้วยจุดเสมอ) และในช่วงของคำว่า BEGIN กับ END. จะมีคำสั่งกี่คำก็ได้ แล้วแต่ผู้เขียนโปรแกรมจะมีวิธีการสำหรับการแก้ปัญหาอย่างไร

กราฟฟิกใน Turbo Pascal และคำสั่งเบื้องต้น

ใน Standard Pascal ไม่มีคำสั่งเขียนกราฟฟิก แต่ใน Turbo Pascal สามารถเขียนกราฟฟิกได้ โดยใช้คำสั่ง uses graph เรียก unit ชื่อ turbo.tpl หรือ graph.tpu การใช้คำสั่งดังกล่าว สามารถเขียนโดยคำสั่ง uses ดังที่เห็นข้างล่าง

```
program test graph;
uses graph;
```

คำว่ากราฟฟิกไม่ใช่หมายถึงการเขียนกราฟเส้นตรง หรือลากเส้นเท่านั้น แต่หมายถึงการวาดรูป ยกตัวอย่างเช่น การเขียการ์ตูน กรือการเขียนลวดลายต่าง ๆ

จอของคอมพิวเตอร์จะมีลักษณะดังที่เห็นข้างล่าง นั่นคือจุดบนจอจะมีค่าเท่ากับ (0,0) ที่ตำแหน่งซ้ายบน และตำแหน่งขวาล่างมีค่าเท่ากับ (getmaxx, getmaxy)

จอ (monitor)

getmaxx และ getmaxy คือฟังก์ชันแบบไม่มีพารามิเตอร์ ดำเนินการส่งค่ามากสุดบนจอกราฟฟิกของแกนนอน และแกนตั้งตามลำดับ สาเหตุที่ใช้ฟังก์ชัน (720*348) เป็นต้น ฟังก์ชันทั้งสองจะทำการตรวจความละเอียดของจอ และส่งค่าดังกล่าวกลับมา

ฟังก์ชัน และโพรซิเจอร์เบื้องต้นที่ใช้สำหรับเขียนกราฟฟิกมีดังต่อไปนี้ คือ

arc bar bar3d circle cleardevice clearviewport closegrap ellipse fillellipse fillpoly floodfill getarccoords getaspectratio getfillsettings getimage getlinesettings getmaxx getmaxy getpixel gettextsettings getviewsettings getx sety imagesize initgraph line linerel lineto moverel moveto outtext outtextxy pieslice putimage putpixel rectangle sector setaspectratio setactivepage setfillpattern setfillstyle setlinestlye settextjustify settextstyle setviewport setusercharsize setvisualpage texthigh textwidth

1) <u>คำสั่งพื้นฐานที่ใช้ในการแสดง และเขียนกราฟฟิก มีดังนี้ คือ</u>

< detect >

แบบ graphdriver:=detect

การดำเนินการ

ตรวจดูว่าจอที่ใช้เป็นจอชนิดใด graphdriver เป็นตัวแปรแบบ integer อาจใช้ชื่ออื่นก็ได้ เมื่อใช้คำสั่งข้างบนแล้ว graphdriver จะเก็บค่าที่กำหนดแบบการแสดงกราฟฟิกของจอที่ใช้อยู่ ควบคู่ไปกับตัวแปร graphmode ซึ่งเป็นตัวแปรเก็บแบบความละเอียดของจอ การให้ค่าตัวแปรทั้งสองนั้นดำเนินการโดยใช้คำสั่งข้างบน

< initgraph >

แบบ initgraph(graphdriver,graphmode,'a:\')

การดำเนินการ

เปลี่ยนการแสดงบนจอเป็นแบบกราฟฟิกโดยนำค่า graphdrive, graphmode ที่ได้มาจากคำสั่งแรกเพื่อใช้ในการเขียนกราฟฟิกบนจอที่ใช้อยู่ ส่วน 'a:\' หมายถึงโปรแกรมที่เกี่ยวกับการแสดงกราฟฟิก (ไฟล์ graph.tpu หรือ ไฟล์.bgi) อยู่ในแผ่นดิสก์อยู่ที่ไดรฟ์ a หรือ อาจเขียนเป็น " ซึ่งจะต้องมีไฟล์ดำเนินการกราฟฟิกอยู่ที่ไดรฟ์ปัจจุบัน (Current Drive)

< line >

แบบ line(x1,y1,x2,y2)

การดำเนินการ

ลากเส้นจากจุด (x1,y1) ไปที่ (x2,y2)

< circle >

แบบ circle(x,y,radius)

การดำเนินการ

เขียนรูปวงกลม จุดศูนย์กลางอยู่ที่ (x,y) รัศมียาว radius

< rectangle >

แบบ rectangle(x1,y1,x2,y2)

การดำเนินการ

เขียนรูปสี่เหลี่ยมโดยส่งค่าจุด 2 จุด

< bar >

แบบ bar(x1,y1,x2,y2)

การดำเนินการ

ระบายพื้นที่ภายในสี่เหลี่ยมโดยส่งค่าจุด 2 จุด

< bar3d >

แบบ bar3d(x1,y1,x2,y2,depth,top)

การดำเนินการ

เขียนกราฟแท่ง 3 มิติ ลักษณะเป็นสี่เหลี่ยมเช่นเดียวกับ คำสั่ง bar สิ่งที่แตกต่างก็คือ การแสดงความลึก (depth) ส่วน top เป็นพารามิเตอร์แบบบูลีน ในกรณีที่เป็น true จะมีเส้นส่วนบนของกราฟแท่ง ถ้าเป็น false จะไม่มีเส้นส่วนบน ซึ่งใช้สำหรับการเขียนกราฟซ้อน ๆ กัน

< arc >

แบบ arc(x,y,stangle,enangle,radius)

การดำเนินการ

เขียนส่วนหนึ่งของวงกลมหรือวงกลมที่มีจุดศูนย์กลางอยู่ที่ (x,y) stangle และ enangle คือ มุมเริ่มต้น และมุมสิ้นสุด ส่วน radius ก็คือ ความยาวรัศมี

< ellipse >

แบบ ellipse(x,y,stagle,enangle,xradius,yradius)

การดำเนินการ

เขียนส่วนหนึ่งของวงรี หรือวงรีที่มีจุดศูนย์กลางที่ (x,y) stangle และ enangle คือมุมเริ่มต้น และมุมสิ้นสุด ส่วน xradius และ radius คือความยาวรัศมีทางแกนนอน และแกนตั้ง

< pieslice >

แบบ pieslice(x,y,stangle,enangle,radius)

การดำเนินการ

เขียน และระบายส่วนหนึ่งของวงกลมที่มีจุดศูนย์กลางที่ (x,y) stangle และ enangle คือมุมเริ่มต้น และมุมสิ้นสุด ส่วน radius คือ ความยาวรัศมี

< putpixel >

แบบ putpixel(x,y,color)

การดำเนินการ

เขียนจุดบนจอที่จุด (x,y) ส่วน color นั้นเป็นคำสั่งกำหนดสี

< getpixel >

แบบ getPixel(x,y)

การดำเนินการ

ส่งสีที่จุด x,y ว่าเป็นสีอะไร

< getmaxx , getmaxy >

แบบ getmaxx , getmaxy

การดำเนินการ

ส่งค่าสูงสุดทางแกนนอน และแกนตั้งของจอที่ใช้งาน

< getx , gety >

แบบ getx , gety

การดำเนินการ

ส่งค่าจุดปัจจุบันทางแกนนอน และแกนตั้ง

< **moveto** >

แบบ moveto(x,y)

การดำเนินการ

เลื่อนจุดปัจจุบันไปยังจุด x,y

< **lineto** >

แบบ lineto(x,y)

การดำเนินการ

ลากเส้นจากจุดปัจจุบันไปยังจุด x,y และเปลี่ยนจุดปัจจุบันไปที่จุด x,y

```
moveto(100,100);
lineto(200,200);
```

มีผลเหมือนกับ line(100,100,200,200); moveto(200,200);

< **linerel** >

แบบ linerel(x,y);

การดำเนินการ

ลากเส้นจากจุดปัจจุบันไปตามแกนนอน และแกนตั้ง จำนวน x จุด และ y จุด สมมุติจุดปัจจุบันคือ (5,5) คำสั่ง linerel(10,10) ก็คือการลากเส้นจากจุด (5,5) ไปที่ (15,15)

< **moverel** >

แบบ moverel (x,y)

การดำเนินการ

เลื่อนจุดปัจจุบันไปตามแกนนอน และแกนตั้ง จำนวน x จุดและ y จุด สมมุติว่าปัจจุบันอยู่ที่ (5,5) คำสั่ง moverel(10,10) คือการเปลี่ยนจุดปัจจุบันจากจุด (5,5) ไปที่ (15,15)

< **cleardevice** >

แบบ cleardevice

การดำเนินการ

ลบภาพกราฟฟิกจากจอ

< closegraph >

แบบ closegraph

การดำเนินการ

ออกจากการแสดงกราฟฟิก (ใช้เมื่อเลิกการแสดง)

2) Turtle Graphics และ Recusive Graphics

< Turtle graphics >

Turtle Graphics เป็นการเขียนกราฟฟิก ที่นิยมใช้กันมากในภาษา LOGO โดยเขียนกราฟฟิกด้วยการกำหนดความยาว และมุมเพื่อเขียนเส้น เพื่อทำการเขียนเส้น ดังนั้น จึงต้องสร้างโพรซิเจอร์ pen และ turn เพื่อหาตำแหน่งจุด และลากจากจุดปัจจุบันไปที่ตำแหน่งที่ต้องการ และเปลี่ยนมุมปัจจุบันซึ่งสามารถเขียนได้ดังที่เห็น

```
var iangle:real; (* ให้ iangle เป็นตัวแปร global แทนมุมปัจจุบัน *)

procedure pen(distance:integer);
var angle:real;
   x,y:integer;
begin
   angle:=iangle*pi/180;
   x:=getx+round(distance*cos(engle));
   y:=gety+round(distance*sin(angle));
   lineto(x,y)
end;
procedure turn(angle:real);
begin
   iangle:=ianle+angle;
end;
```

pen(L)

turn (90)

pen(L)

โพรซิเจอร์ทั้งสองจะเก็บไว้ในไฟล์ชื่อ turtle.pas เมื่อต้องการใช้จะเรียกไฟล์ดังกล่าวโดยใช้คำสั่ง {$I turtle.pas}

3. <u>คำสั่งกราฟฟิกที่ใช้ในการกำหนดแบบการเขียนกราฟฟิก</u>

คำสั่งกราฟฟิกที่ใช้ในการกำหนดแบบการเขียนเส้น ระบาย และการเขียนตัวอักษรมีดังนี้

< setlinestyle >

แบบ setlinestyle (linestyle,pattern,thickness)

การดำเนินการ

กำหนดรูปแบบเส้นที่ต้องการลาก (linestyle) แบบ (pattern) และความหนา (thickness) โดยมีหลักการดังนี้คือ

linestyle มีแบบตั้งแต่ 0 ถึง 4 (0 ถึง 3 เป็นแบบที่มีอยู่ 4 เป็นแบบที่ผู้ใช้กำหนด

pattern เป็นตัวแปรแบบ word ผู้ใช้สามารถเลือกตัวเลขภายในขอบเขต $0..ffff หรือ 0..65535 เพื่อออกแบบลักษณะของเส้น (ใช้ได้ในกรณีที่ linestyle เป็น 4 เท่านั้น)

thickness มีแบบ 1 กับ 3 ซึ่งเป็นความหนาของเส้นแบบธรรมดา และแบบหนา

คำสั่งนี้มีผลต่อการลากเส้นตรงต่าง ๆ เช่น line bar drawpoly rectangle เป็นต้น

< setfillstyle >

แบบ setfillstyle (pattern,Color)

การดำเนินการ

กำหนดรูปแบบ (pattern) และสี (color) ที่จะระบาย

pattern คือแบบที่ต้องการจะระบายมี 12 แบบ แบบสุดท้ายคือแบบที่ผู้ใช้สามารถกำหนดเองโดยคำสั่ง setfillpattern สำหรับ color คือสีที่ต้องการระบาย

คำสั่งนี้มีผลต่อคำสั่ง bar bar3d fillpoly pieslice floodfill

< setfillpattern >

แบบ setfillpattern(pattern,color)

การดำเนินการ

กำหนดรูปแบบการระบายโดยผู้ใช้ pattern โดยกำหนดให้ pattern เป็นตัวแปรแบบ fillpatterntype ซึ่งเป็นแบบข้อมูลที่ได้กำหนดไว้ล่วงหน้าแล้วมีลักษณะเป็น

```
fillpatterntype :array [1..8] of byte;
```

การให้ค่ากับ pattern อาจให้ค่าได้มากน้อยขึ้นอยู่ตามความต้องการ ดังตัวอย่าง

```
const pattern:fillpatterntype=($aa,$55,$aa,$55,$aa,
$55,$aa,$55)
```

< settextstyle >

แบบ settextstyle(font,direction,size);

การดำเนินการ

กำหนดแบบตัวอักษร (font) ทิศตั้ง หรือนอน (direction) และขนาด (size)

font	คือแบบตัวอักษรมีตั้งแต่ 0 ถึง 4
direction	คือแนวตัวอักษร 0 คือการเขียนแนวนอน 1 คือการแนวตั้ง
size	คือขนาดตัวอักษร ตั้งแต่ 1 ถึง 10 (จากเล็กไปใหญ่)

คำสั่งนี้มีผลต่อการเขียน หรือตรวจตัวอักษรบนจอกราฟฟิก เช่น outtext texthigh ในแผ่นดิสก์ turbo pascal ต้องมีไฟล์ .chr ด้วย ถ้าไม่มีจะเป็นตัวอักษรธรรมดา

< settextjustify >

แบบ settextjustify(horiz,vert);

การดำเนินการ

กำหนดตำแหน่งที่ต้องการแสดงอักษร horiz และ vert ต่างก็มีค่าตั้งแต่ 0 ถึง 2 horiz มีค่าเป็น 0 หมายถึงแสดงชิดทางซ้าย 1 หมายถึงตรงกลาง และ 2 หมายถึงชิดขวา vert มีค่าเป็น 0 หมายถึงแสดงชิดทางล่าง 1 หมายถึงตรงกลาง และ 2 หมายถึงชิดบน

< outtext,outtextxy >

แบบ outtext(textstring),outtextxy(x,y,textstring)

การดำเนินการ

คำสั่ง outtext เป็นเขียน textstring (ข้อมูลแบบสตริง) ลักษณะ ขนาด และแนวตัวอักษรนั้นกำหนดโดยใช้คำสั่ง settextstyle ตำแหน่งที่เริ่มเขียนคือ ตำแหน่งปัจจุบันเนื่องจากการเขียนตัวอักษรบนจอกราฟฟิกไม่สามารถใช้คำสั่ง write หรือ writeln ได้ นอกจากคำสั่ง outtext แล้ว ยังมีคำสั่ง outtextxy ซึ่งมีแบบเป็น outtextxy(x,y,textstring) x และ y ก็คือ ตำแหน่งที่ต้องการแสดงตัวอักษร

ex

```
settextstyle(1,0,5);      settextjustify(2,0);
outtextxy(getmaxx,getmaxy,'Oh!God');
```

การแสดงตัวอักษรก็จะเป็นแบบแนวนอนขนาดกลาง และอยู่ตัวสุดท้าย (d) อยู่ชิดขวาล่าง

< drawploy >

แบบ drawpoly (numpoint,polypoints)

การดำเนินการ

ลากเส้นจำนวน numpoint เส้น โดยลากจากจุดที่ 1 ไปยังจุดที่ 2 จากจุดที่ 2 ไปจนกระทั่งถึงจุดสุดท้าย polypoints เป็นแบบข้อมูลที่มีลักษณะเป็นดังข้างล่าง

```
type polypoints  =  array [1..numpoint] of record
                                          x : integer;
                                          y : integer;
                                        end;
```

< fillpoly >

แบบ fillpoly

การดำเนินการ

ระบายพื้นที่ภายในรูปหลายเหลี่ยมที่ลากโดยคำสั่ง drawpoly ในกรณีที่เส้นแรกกับเส้นสุดท้ายไม่บรรจบกัน การระบายจะระบายโดยให้จุดแรกกับ

จุดสุดท้ายเชื่อมติดกัน

< floodfill >

แบบ floodfill(x,y,Color:word)

การดำเนินการ

ระบายพื้นที่ที่ตำแหน่ง (x,y) และรอบ ๆ ข้างโดยสี color ถ้าหากตำแหน่ง (x,y) อยู่ภายในรูป จะเป็นการระบายภายในรูป ถ้าตำแหน่งอยู่นอกรูประบายพื้นที่รอบนอก

< setusercharsize >

แบบ setusercharsize(multx,divx,multy,divy)

การดำเนินการ

ปรับขนาดตัวอักษรที่ใช้อยู่ในปัจจุบันให้ใหญ่ เล็กตามความต้องการ เช่น ถ้าต้องการให้ตัวอักษรกว้างขึ้นกว่าปัจจุบันเป็นสองเท่า และสูงเป็นครึ่งหนึ่ง ก็สามารถเขียนได้โดยใช้คำสั่ง

setusercharsize(2,1,1,2)

< fillellipse >

แบบ fillellipse(x,y,xradius,yradius)

การดำเนินการ

เขียนและระบายวงรี โดยมีจุดศูนย์กลางวงรีอยู่ที่ (x,y) และความยาวรัศมีแกน x และแกน y เท่ากับ xradius และ yradius ตามลำดับ

< sector >

แบบ sector(x,y,stangle,endangle,xradius,yradius)

การดำเนินการ

ดำเนินการเช่นเดียวกับคำสั่ง fillellipse และยังสามารถกำหนดมุมเริ่มต้น และมุมสิ้นสุดที่พารามิเตอร์ stangle,endangle ตามลำดับ

< getLinesettings >

แบบ getlinesettings(lineinfo)

การดำเนินการ

ส่งลักษณะของเส้น ซึ่งถูกกำหนดโดยคำสั่ง setlinestyle

ล่าสุดให้กับ lineinfo ซึ่งเป็นข้อมูลแบบ linesettingstype โดยมีโครงสร้างดังที่เห็นข้างล่าง

```
type linesettingstype  =  record
                            linestyle,pattern,thickness:word
                          end;
```

< getfillsettings >

แบบ getfillsettings(fillinfo);

การดำเนินการ

ส่งสี และลักษณะของการระบาย ซึ่งถูกกำหนดโดยคำสั่ง setfillstyle ล่าสุดให้กับตัวแปร fillinfo ซึ่งมีแบบเป็น fillsettingstype โดยมีโครงสร้างดังข้างล่าง

```
type    fillsettingstype = record
                             pattern,color:word
                           end;
```

< gettextsettings >

แบบ gettextsettings(textinfo);

การดำเนินการ

ส่งชนิดตัวอักษร ซึ่งกำหนดโดยคำสั่ง settextstyle ล่าสุดให้กับ textinfo ซึ่งเป็นแบบ textsettingstype โดยมีโครงสร้างดังข้างล่าง

```
type    charsizetype = 1..10;
        textsettingstype
              = record
                  font,direction  : word;
                  charsize        : charsizetype;
                  horiz,vert      : word
                end;
```

horiz และ vert คือ ค่าตำแหน่งที่เริ่มเขียนตัวอักษรทางแกน x และ y ตามลำดับ

< setviewport >

แบบ setviewport(X1,Y1,X2,Y2,clip);

การดำเนินการ

สร้างขอบเขตขึ้นภายในกรอบสี่เหลี่ยม ซึ่งมีตำแหน่งมุมซ้ายบนที่ (X1,Y1) และมุมขวาล่างที่ (X2,Y2) สมมุติว่าใช้คำสั่ง setviewport(100, 100,200,200,true) เมื่อใช้คำสั่ง line(0,0,20,20) เส้นที่ปรากฏจะเป็นเส้นที่ลากจากตำแหน่ง (100,100) ถึง (120,120) สำหรับ clip เป็นข้อมูลแบบบูลีน ถ้าเป็น true การแสดงกราฟฟิกจะไม่เลยกรอบที่ได้กำหนดขึ้นมา

< getviewsetting >

แบบ getviewsettings(viewport);

การดำเนินการ

ส่งค่าของกรอบที่กำหนดโดยคำสั่ง setviewport ล่าสุดให้กับ viewport ซึ่งเป็นตัวแปรแบบ viewporttype โดยมีโครงสร้างดังข้างล่าง

```
type viewporttype = record
                      x1,y1,x2,y2:word; clip:boolen
                    end;
```

< clearviewport >

แบบ clearviewport

การดำเนินการ

ลบกราฟฟิกภายในกรอบที่กำหนดโดยคำสั่ง sertviewport ในกรณีที่ต้องการกำหนดกรอบขึ้นใหม่ ต้องใช้คำสั่ง setviewport ใหม่อีก

< getimage >

แบบ getimage(X1,Y1,x2,Y2,Bitmap);

การดำเนินการ

เก็บภาพบนจอที่อยู่ในกรอบสี่เหลี่ยม (x1,y1,x2,y2) ลงบัฟเฟอร์ bitmap ซึ่งเป็นตัวแปรแบบ pointer (แบบตัวแปรนี้มีอยู่ใน Turbo

Pascal) ชื่อตัวแปรอาจใช้ชื่ออื่นได้ คำสั่งนี้เหมาะสำหรับการเขียนกราฟฟิกให้เคลื่อนไหว โดยเก็บรูปต่าง ๆ ไว้ที่ตัวแปรที่ต่างกัน และแสดงออกเป็นลำดับ

< putimage >

แบบ putimage(x,y,bitmap,bitbit)

การดำเนินการ

แสดงกราฟฟิกที่เก็บไว้ที่ bitmap จากคำสั่ง getimage โดยมุมบนซ้ายของรูปอยู่ที่ตำแหน่ง (x,y) bitbit คือ แบบการแสดงของเซต (1 ถึง 5) รายละเอียดมีดังนี้ คือ

1. แสดงภาพปรกติ

2. แสดงภาพ โดยตรวจดูว่าตำแหน่งที่จะแสดงภาพว่ามีการแสดงภาพอยู่ก่อนแล้วหรือไม่การแสดงภาพจะแสดงในกรณีที่ตำแหน่งดังกล่าวไม่มีการแสดงภาพอยู่ก่อน (xor)

3. แสดงภาพโดยตรวจดูว่าตำแหน่งที่จะแสดงภาพอยู่ก่อนแล้วหรือไม่ การแสดงภาพจะแสดงทุกกรณีไม่ว่าตำแหน่งดังกล่าวมีการแสดงภาพก่อนหรือไม่(or)

4. แสดงภาพ โดยตรวจดูว่าตำแหน่งที่จะแสดงภาพว่ามีการแสดงภาพอยู่ก่อนแล้วหรือไม่ การแสดงภาพจะในกรณีที่ตำแหน่งดังกล่าวมีการแสดงภาพอยู่ก่อน (and)

5. แสดงภาพ ซึ่งกลับข้างกับการแสดงภาพที่มีอยู่ก่อน (not)

< imagesize >

แบบ imagesize(X1,Y1,X2,Y2, : word)

การดำเนินการ

พังก์ชันหาค่าจำนวนไบท์ ที่เก็บส่วนของจอภาพในกรอบสี่เหลี่ยม (X1,Y1,X2,Y2)

< getarccoords >

แบบ getarccoords(arccoords)

การดำเนินการ

ส่งพารามิเตอร์ของคำสั่ง arc ล่าสุดให้กับตัวแปร arccoords ซึ่งมีแบบข้อมูลเป็น arccoordstyle โดยมีโครงสร้างเป็นข้อมูลดังข้างล่าง endx, eny คือ ตำแหน่งของปลายเส้นโค้ง

```
type  arccoordstype = record
```

```
    x,y:integer;
   startx,starty,endx,endy:word
end;
```

< textheight,textwidth >

แบบ textheight(textstring),textwidth(textstring)

การดำเนินการ

ส่งค่าความสูง และความกว้างของสตริง textstring โดยหน่วยเท่ากับจำนวนจุดบนจอ คำสั่งนี้มีประโยชน์ต่อการจัดการตำแหน่งของตัวอักษรบนจอให้เหมาะสม

< setaspectratio >

แบบ setaspectratio(xasp,yasp)

การดำเนินการ

กำหนดอัตรส่วนของการแสดงกราฟฟิกทางแกนนอน และแกนตั้ง คำสั่งนี้มีประโยชน์ในการวาดวงกลมให้กลมจริง ๆ ในกรณีที่คำสั่ง circle ไม่สามารถวาดวงกลมได้ วิธีการกำหนดค่านั้นสามารถเขียนได้ดังนี้

setaspectratio(50,60) ; ในกรณีที่ต้องการให้วงกลมกว้างขึ้น

และ setaspectratio(60,50) ; ในกรณีที่ต้องการให้วงกลมสูงขึ้น

< getaspectratio >

แบบ getaspectratio(xasp,yasp)

การดำเนินการ

การแสดงกราฟฟิกออกจอ สมมุติว่าใช้คำสั่ง rectangle (0, 0,50,50) การเขียนกราฟฟิกไม่เป็นรูปสี่เหลี่ยมจตุรัส คำสั่ง getaspectratio ใช้สำหรับส่งอัตราส่วนแกน x กับ y โดยให้ xasp กับ yasp เป็นตัวแปรแบบ word และ rate เป็นตัวแปรแบบ real เมื่อต้องการเขียนรูปสี่เหลี่ยมจตุรัส หรือลากเส้นแทยงมุมบนจอก็สามารถเขียนดังข้างล่าง

ex

```
getaspectratio(xasp,yasp);
rate:=xasp/yasp;
```

```
        rectangle(0,0,50,round(50*rate));
หรือ    line(0,0,100,round(100*rate));
```

< setactivepage >

แบบ setactivepage(page)

การดำเนินการ

เก็บรูปกราฟฟิกลงในจอที่ต้องการ ดังตัวอย่างข้างล่างนี้

ex

```
setactivepage(1);
rectangle(0,0,100,100);
```

จากตัวอย่าง รูปสี่เหลี่ยมจะถูกเก็บอยู่ที่จอ 1 (โดยปรกติจอที่ใช้อยู่คือจอที่ 0)

< setvisualpage >

แบบ setvisualpage(page)

การดำเนินการ

แสดงกราฟฟิกที่เก็บไว้โดยคำสั่ง setactivepage

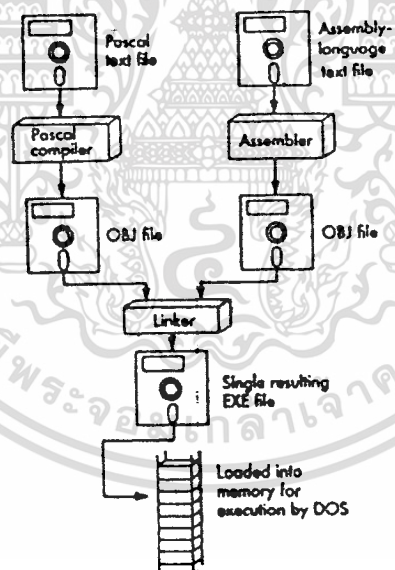
จากตัวอย่างคำสั่ง setactivepage เมื่อใช้คำสั่ง setvisual page(1) รูปสี่เหลี่ยมที่เก็บไว้ที่จอ 1 จะปรากฏบนจอคำสั่งทั้งสองเป็นคำสั่งในการเลือกจอ และแสดงกราฟฟิกของจอนั้น ๆ ซึ่งเหมาะที่จะแสดงรูปกราฟฟิกที่แตกต่างกันโดยทำการเรียกจอสลับไปมา

การอินเตอร์เฟสภาษาแอสเซมบลีกับบาสคาล

การใช้ภาษาแอสเซมบลีร่วมกับภาษาระดับสูง เช่นภาษาเบสิก และภาษาบาสคาลเป็นทางออกที่ดีในการแก้ปัญหาหลาย ๆ แบบ เนื่องจากภาษาแอสเซมบลีทำงานได้เร็ว และสามารถเข้าถึงฮาร์ดแวร์ของคอมพิวเตอร์ได้ ในขณะที่ภาษาระดับสูงเขียนได้ง่าย รวดเร็ว และเข้าใจได้ง่ายกว่าภาษาแอสเซมบลี ดังนั้นผู้เขียนโปรแกรมจึงเอาประโยชน์ของโปรแกรมทั้งสองแบบมาใช้ โดยการเขียนโปรแกรมหลักเป็นภาษาแอสเซมบลีร่วมกับภาษาบาสคาล

1. การอินเตอร์เฟสโปรแกรม

ภาษาบาสคาล และภาษาเบสิกมีวิธีการอินเตอร์เฟสกับภาษาแอสเซมบลีที่ต่างกัน กล่าวคือภาษาบาสคาลเมื่อถูกคอมไพล์หรือแปลจากภาษาบาสคาลเป็นภาษาเครื่อง จะได้ OBJ ไฟล์ ภาษาแอสเซมบลีเมื่อถูกแอสเซมเบลอร์แล้ว ก็จะได้ภาษาเครื่องเป็น OBJ ไฟล์เช่นกัน ต่อมาโปรแกรม LINK ก็จะรวมเอา OBJ ไฟล์หลายหลาย อันเข้าด้วยกัน และสร้าง EXE ไฟล์ขึ้นใหม่เพียงอันเดียว ดังแสดงในรูปที่ 1.6



รูปที่ 1.6 ภาษาแอสเซมบลี และภาษาบาสคาล

2. ปัญหาในการอินเตอร์เฟสโปรแกรม

เมื่อต้องการใช้โปรแกรมซึ่งเกิดจากการรวมโปรแกรมหลายภาษาเข้าด้วยกันมักมีปัญหาเกิดขึ้น ดังนั้นผู้ใช้จำเป็นต้องเข้าใจปัญหาในเบื้องต้นก่อนว่ามีอะไรบ้าง และวิธีการแก้ปัญหาซึ่งแต่ละภาษาอาจมีวิธีการที่แตกต่างกัน ซึ่งจะกล่าวถึงการ

แก้ปัญหาของภาษาเบสิก และภาษาปาสคาลในตอนต่อไป ปัญหาที่อาจเกิดขึ้นในการอินเตอร์เฟสโปรแกรมของภาษากันมีดังนี้

2.1 การจัดสรรเนื้อที่ในหน่วยความจำ

โดยปกติถ้าผู้ใช้เขียนโปรแกรมเป็นภาษาระดับสูง เช่น ภาษาเบสิก หรือภาษาปาสคาล ปัญหาที่ว่าโปรแกรมที่เขียนขึ้นจะอยู่ส่วนใดของหน่วยความจำจะไม่เกิดขึ้น เนื่องจากว่าโปรแกรมแปลภาษาจะทำหน้าที่จัดการในส่วนนี้ให้ แต่เมื่อมีการใช้ภาษาระดับสูงร่วมกับภาษาแอสเซมบลี ก็อาจเกิดปัญหาได้ ดังได้กล่าวตอนที่แล้วเกี่ยวกับการอินเตอร์รัพท์ โปรแกรมภาษาเบสิกกับภาษาแอสเซมบลี ดังนั้นจึงจำเป็นต้องรู้วิธีที่จะเอาโปรแกรมใส่ในหน่วยความจำ พร้อมที่บอกโปรแกรมภาษาเบสิกได้ว่า จะเรียกใช้โปรแกรมภาษาแอสเซมบลีได้จากที่ไหน สำหรับภาษาปาสคาลไม่ค่อยมีปัญหาในเรื่องนี้ เนื่องจากว่าภาษาแอสเซมบลีเมื่อถูกแปลเป็นภาษาเครื่องแล้ว linker ก็ทำหน้าที่รวมโปรแกรมภาษาปาสคาล และภาษาแอสเซมบลี และตัดสินให้ว่าควรอยู่ในส่วนใดของหน่วยความจำให้

2.2 การส่งผ่านการควบคุมไปยังโปรแกรมภาษาแอสเซมบลี

เมื่อโปรแกรมภาษาแอสเซมบลีถูกใช้ร่วมกับโปรแกรมภาษาระดับสูงการส่งผ่านการควบคุมจากโปรแกรมภาษาระดับสูง ไปยังโปรแกรมภาษาแอสเซมบลี มักทำโดยการ CALL ซึ่งมักจะเป็น FAR CALL ในภาษาแอสเซมบลี คือต้องมีการเก็บค่าโคดเซกเมนต์ และค่าของออฟเซตของคำสั่งถัดจากคำสั่ง CALLในโปรแกรมไว้บน stack ก่อนส่งผ่านการควบคุมไป เริ่มทำงานในส่วนของโปรแกรมภาษาแอสเซมบลีในภาษาเบสิกสามารถเรียกใช้โปรแกรมแอสเซมบลี โดยใช้คำสั่ง USR หรือคำสั่ง CALL ในภาษาปาสคาลสามารถเรียกใช้โปรแกรมภาษาแอสเซมบลีโดยใช้คำสั่ง CALL ปัญหาที่อาจเกิดขึ้นก็คือ แล้วโปรแกรมภาษาระดับสูงจะรู้ได้อย่างไรว่าโปรแกรมภาษาแอสเซมบลีอยู่ที่ไหน สำหรับภาษาเบสิกผู้เขียนโปรแกรมต้องรู้ว่าโปรแกรมภาษาแอสเซมบลีจะอยู่ตรงไหน และบอกที่อยู่นี้กับโปรแกรมภาษาเบสิกในคำสั่ง DEF SEG และ DEF USR สำหรับโปรแกรมภาษาปาสคาล linker จะรู้ว่าโปรแกรมภาษาแอสเซมบลีอยู่ที่ไหน แล้วบอกโปรแกรมภาษาปาสคาลให้ โดยผู้เขียนโปรแกรมไม่ต้องยุ่ง

2.3 การส่งผ่าน arguments

ในภาษาระดับสูง เมื่อโปรแกรมหลักเรียกใช้โปรแกรมย่อย บางครั้งต้องมีการส่งผ่านค่าไปมาระหว่างโปรแกรมหลัก และโปรแกรมย่อย กล่าวคือ โปรแกรมหลักต้องส่งค่าเริ่มต้นให้โปรแกรมย่อยก่อน แล้วโปรแกรมก็นำค่าเหล่านี้ไปประมวลผลเมื่อได้ผลลัพธ์แล้วโปรแกรมย่อยก็ต้องส่งค่าเหล่านี้ไปให้โปรแกรมหลัก ซึ่งค่า

ที่ถูกส่งไปมานี้อาจเป็นค่าที่นำมาประมวลผลได้เลย หรือตำแหน่งที่อยู่ของค่า เราเรียกค่าที่ส่งผ่านไปมาระหว่างโปรแกรมหลัก และโปรแกรมย่อยว่า arguments

เมื่อมีการใช้ภาษาระดับสูงร่วมกับภาษาแอสเซมบลี ก็ต้องมีการตกลงกันว่า ถ้ามีการติดต่อกันระหว่างภาษาระดับสูง และภาษาแอสเซมบลี argument ควรเป็นอย่างไร สำหรับภาษาเบสิก คำสั่ง USR สามารถส่ง argument ให้ภาษาแอสเซมบลีได้เพียงตัวเดียวและภาษาแอสเซมบลีสามารถส่งค่าคืนให้ภาษาเบสิกเป็น argument เพียงตัวเดียว โดยค่านี้จะอยู่ในส่วนที่เรียกว่า Floating Point Accumulator หรือ FAC สำหรับคำสั่ง CALL สามารถส่งผ่าน arguments ไปมาได้หลายตัว โดยที่ตำแหน่งที่อยู่ของ arguments จะอยู่บน stack สำหรับภาษาปาสคาล คำสั่ง CALL สามารถส่งผ่าน arguments ไปมาระหว่างภาษาปาสคาลและภาษาแอสเซมบลีได้หลายตัว โดยการส่งตำแหน่งที่อยู่ของ arguments หรืออาจเป็นค่าของ arguments ก็ได้ไว้บน stack

3. <u>การอินเตอร์เฟสภาษาปาสคาล กับภาษาแอสเซมบลี</u>

ภาษาปาสคาลมีวิธีการส่งผ่าน arguments ไปยังภาษาแอสเซมบลีด้วยวิธีการเช่นเดียวกับคำสั่ง call ในภาษาเบสิก คือการใช้ stack ซึ่งในโปรแกรมตัวอย่างจะเป็นการเรียกโปรแกรมภาษาแอสเซมบลีชื่อ TIMER ด้วยพารามิเตอร์ตัวเดียวชื่อ A ซึ่งค่าที่ส่งกลับจาก TIMER จะเป็นเวลา แล้วโปรแกรมภาษาปาสคาลจะคำนวณเวลาเป็น ชั่วโมง นาที และวินาที แล้วพิมพ์เวลาออกมา

ตัวอย่างที่ 1 โปรแกรมภาษาปาสคาลที่คำนวณเวลา

```
program time (input,output);
var a,hours,mins,secs:integer4;
procedure timer (var i:integer4);external;
begin
     timer(a)
     hours      :=   a div65543;
     a          :=   a-hours*65543;
     mins:      :=   div 1092;
     a          :=   a - mins*1092;
     secs       :=   a div 18;
     write(time of day is');
     writen(hours:2,':',mins:2,':',secs:2)
end.
```

ตัวอย่างที่ 2 โปรแกรมภาษาแอสเซมบลีที่ อ่านค่าเวลา

```
                page        ,132
frame           struct
savebp          dw          ?
saverct         dd          ?
a               dd          ?
frame           ends        ?
code-scg        segment
dgroup          group       data
                assume      cs:code-seg,DS:dgroup,
                            ES:dgroup
```



```
                assume      ss:dgroup
timer           proc        far
                public      timer
                push        bp
                mov         bp,sp;establish address
                            to frame
                int         lah  ;call BIOS for time
                            of day
                les         bx,[bp]+a;get pointer in
                            ES:BX to parameter
                mov         ES  :  [bx],dx;store low
                            part of time
                mov         ES:[bx + 2],cx;store high
                            part of time
                pop         bp
                ret         4
timer           endp
code-seg        ends
                end
```

ตัวอย่างการเชื่อมโยงกับภาษา Assembly

```
program addt;
var d:integer;
     procedure plus(a,b : integer; var c : integer;
     external;
     {$L test}
begin
     plus(100,200d);
     writeln('a+b = ',d);
end.
```

สำหรับโพรซิเจอร์ในภาษาแอสเซมบลีสามารถเขียนได้ดังนี้

```
code     segment byte public
         assume cs:code
         public plus
;procedure plus(a,b:integer; var c : integer);external
plus     proc    near
         push    dp
         mov     bp,sp
         mov     ax,[bp+10]          ; A:integer
         add     ax,[bp+8]           ; A:=A+B
         les     dl,dword ptr [bp+4] ; es = segment of
                 variable c
                                     ; di = offset of
                                       variablec
         stosw                       ; store  ax
                                     ; es:[di]
         pop     bp
         ret     4
plus     endp
code     ends
         end
```

เมื่อคอมไพล์ภาษาแอสเซมบลีจะได้ไฟล์ชื่อ test.obj

การใช้ภาษาแอสเซมบลีเชื่อมกับภาษา pascal มีหลักดังนี้

- ส่วนประกาศโพรซิเจอร์ หรือฟังก์ชันมีคำ extermal ต่อท้าย
- ใช้ directive $L ตามด้วยชื่อไฟล์โปรแกรมย่อย (test.obj)
- ใช้ชื่อเซกเมนต์ของคำสั่งว่า code
- ชื่อโพรซิเจอร์ หรือฟังก์ชั่นในภาษา Pascal ต้องอยู่ในคำสั่ง Public และตรงกับชื่อ Proc ในภาษา Assembly
- พิจารณาลักษณะการเรียกโพรซิเจอร์ หรือฟังก์ชั่นว่าเป็น near หรือ far และโปรแกรมย่อย Assembly กับ Pascal อยู่ในเซ็กเมนต์เดียวกัน หรือไม่
- พิจารณาพารามิเตอร์ที่ส่งมาจาก Pascal และที่จะส่งกลับ

<u>การสั่งงานคอมไพเลอร์ (Compiler Directives)</u>

Turbo Pascal สามารถสั่งงานคอมไพเลอร์ได้หลายวิธี ดังที่เห็น

- ในโปรแกรม โดยใช้ {$directive}
- ใน integreated environment (ในเมนู Options/ compiler)
- Command Line สำหรับคอมไพเลอร์ TPC โดยมีแบบฟอร์มดังนี้ >tpc program /$<directive>
- ในไฟล์ tpc.cfg หรือ turbo.tp

1) Compiler Directive อยู่ในวงเล็บปีกกาโดยจะเริ่มต้นด้วยตัวอักษร $ และตามด้วยชื่อหรือสัญญลักษณ์ซึ่งอาจเป็นอักษรเดียว หรือหลายตัวอักษร เพื่อสั่งงานคอมไพเลอร์

Compiler Directive มีอยู่ 3 ชนิดด้วยกัน

1. แบบสวิทช์ มีลักษณะคล้ายสวิทช์เปิดปิดโดยใช้เครื่องหมาย +,- หลัง directive เช่น {$B+} {$R-}

2. แบบมีพารามิเตอร์ โดยมีพารามิเตอร์ตามหลังชื่อ directive ซึ่งอาจเป็นชื่อไฟล์ หรือขนาดของหน่วยความจำ เช่น {$I TYPE.INC} {$M 65520, 8192, 655360}

3. แบบมีเงื่อนไข ใช้ควบคุมการคอมไพล์โปรแกรม โดยสัญญลักษณ์เงื่อนไขที่ได้วางไว้ในลักษณะของ if และ else และเงื่อนไขก็คือการนิยามสัญญลักษณ์เงื่อนไขไว้หรือไม่

ตัวอย่างคือ

```
{$DEFINE Debug}
{$IFDEF Debug}
    .....(ข้อความที่ต้องการให้คอมไพล์)
{SENDIF}
```

หากสัญญลักษณ์เงื่อนไข ไม่เป็นจริงจะไม่คอมไพล์ข้อความดังกล่าว

และในการเขียน Compiler Directives แบบมีตัวแปรต้องมีช่องว่าง หรือเว้นวรรคระหว่างชื่อ Directive กับตัวแปรต่าง ๆ ส่วนแบบที่เหลือห้ามมีเว้นวรรค

Compiler Directive แบบสวิทช์นี้มี 2 ชนิดด้วยกันนั่นคือ

1. ชนิด global จะต้องอยู่หลังโปรแกรม หรือก่อนยูนิต

2. ชนิด local มีลักษณะในทางตรงกันข้ามกับชนิด global นั่นคือ อยู่ที่ใดก็ได้ซึ่งจะมีผลเฉพาะส่วนที่อยู่หลัง Compiler Directive จนกว่าจะมีการเปลี่ยนแปลง

Compiler Directive แบบสวิทช์นี้ จะสามารถรวมอยู่ใน Comment (วงเล็บปีกกา) เดียวกันได้โดยคั่นด้วย จุลภาค ',' ดังเช่น {$B+,R+,$+} โดยใช้คอมมาคั่น

บทที่2

การสร้างยูนิต และโปรแกรมช่วยต่าง ๆ
DIGITAL TRAINER SOFTWARE

การสร้าง และใช้งานยูนิตทั่วไป

ก่อนอื่นควรเข้าใจถึงการสร้าง และวิธีการใช้ยูนิตเสียก่อน ยูนิตคือโพรซิเจอร์ที่คอมไพล์เป็นภาษาเครื่อง การสร้างยูนิตสามารถสร้างได้ดังที่เห็นข้างล่าง (ต้องตั้งชื่อไฟล์เป็นชื่อเดียวกับยูนิต)

```
unit test;
inteface
(* ใส่แบบข้อมูล หรือตัวแปรแบบ global ที่นี่ ถ้าจำเป็น *)
procedure plus(i,j:integer;var k:integer);
(* ใส่โพรซิเจอร์ หรือฟังก์ชันอื่น ๆ ในกรณีที่มีมากกว่าหนึ่งโพรซิ
 เจอร์ *)
implementation
procedure plus;
(* ประกาศตัวแปรแบบ local ที่นี่ ถ้าจำเป็น *)
 begin
    k:=i+j
 end;
(* สร้างโพรซิเจอร์ หรือฟังก์ชันอื่นที่นี่ ในกรณีที่มีมากกว่า 1 โพรซิ
 เจอร์ หรือฟังก์ชัน *)
end.
```

เมื่อคอมไพล์ลงดิสก์ จะปรากฏยูนิตชื่อ test.tpu หลังจากนั้น ก็สามารถใช้โพรซิเจอร์ plus ได้ดังที่เห็นข้างล่าง

ex)
```
program testyou;
uses test;      (* คำสั่งเรียกยูนิต test *)
var L:integer;
begin
   plus(10,20,L);
   writeln(L)
```

```
end.
```

< TPUMOVER >

เมื่อสร้างยูนิตแล้ว อาจเก็บไว้เป็นไฟล์ .TPU ธรรมดา หรือเก็บใส่ไฟล์ TURBO.TPL ก็ได้ วิธีการเก็บใส่ไฟล์ TURBO.TPL สามารถเก็บโดยใช้คำสั่ง

```
> TPUMOVER TURBO TEST
```

ขั้นตอนการเก็บยูนิต (TEST) เข้าใน TURBO.TPL สามารถเขียนได้ดังนี้

1. กด F6 เพื่อย้ายเคอร์เซอร์ไปที่หน้าต่างขวา
2. กด + เพื่อเลือกไฟล์ TEST
3. กด INS เพื่อย้ายยูนิต
4. กด F6 เพื่อย้ายเคอร์เซอร์กลับมาที่หน้าต่างเดิม
5. กด F2 เพื่อบันทึก TURBO.TPL
6. กด Esc เพื่อกลับสู่ DOS

หมายเหตุ
ในกรณีที่ต้องการย้ายยูนิตออก ให้เปลี่ยนจาก Ins เป็น Del

< MAKE >

การใช้คำสั่ง MAKE เริ่มจากการเขียนไฟล์เพื่อใช้งานโดยมีหลักการทั่วไปในการเขียนไฟล์ MAKE ดังนี้

```
TARGET  [TARGET..] : SOURCE [SOURCE]
/ เขียนคอมเมนต์ที่นี่จะอยู่ภายในเครื่องหมาย '/'  /
[COMMAND] (* ต้องย่อหน้าอย่างน้อย 1 ช่องไฟ *)
[COMMAND] (* คำสั่งใน COMMAND จะใช้คำสั่งใด ๆ ก็ได้ เช่น
             cd path ฯลฯ )
...
```

เมื่อสร้างไฟล์เรียบร้อย ปรกติจะเก็บไฟล์ในชื่อของ MAKEFILE และใช้คำสั่ง

```
  MAKE     ได้ทันที
```

หรือจะใช้ไฟล์ที่ตั้งชื่อเอง เช่น testmake.mak และใช้คำสั่ง

```
  MAKE -ftestmake.mak    ก็ได้
```

การเขียนไฟล์ MAKE สามารถเขียนได้หลายวิธีดังต่อไปนี้

1. <u>การเขียนแบบเจาะจง</u>

ex)
```
test1.obj: test1.asm
tasm test1.asm test1.obj

mymake.tpu wymake.pas
tpc mytest /tc:\tp5\bin
```

จากตัวอย่างข้างบน เป็นการสร้างไฟล์ test1.obj จาก test1.asm โดยคำสั่ง tasm (Turbo Assemly) และสร้างไฟล์ mytest.tpu และเก็บไฟล์ดังกล่าวไว้ที่ไดร์ฟ c ในซับไดเร็กทอรี tp5\bin ตามลำดับ

2. <u>การเขียนแบบทั่วไป</u>

ex)
```
.asm.obj:
tasm $*.asm,$*.obj;
```

จากตัวอย่างนี้ จะเป็นการสร้างไฟล์ .obj ขึ้นจากไฟล์ .asm ที่มีอยู่ทุกไฟล์

ที่ส่วนต้นของ COMMAND ยังสามารถกำหนดแบบของการใช้คำสั่ง MAKE ได้ดังตัวอย่างต่อไปนี้

@ เป็นคำสั่งให้ดำเนินการคำสั่ง MAKE โดยไม่แสดงรายละเอียดออกจากจอ

-num เป็นการกำหนดตัวเลข exit code ในกรณี exit code เกินตัวเลขที่กำหนดจะเลิกดำเนินการคำสั่ง MAKE

\- ไม่มีการตรวจ exit code

3. <u>การเขียนโดยใช้ Macros</u>

ในกรณีที่มีการใช้คำสั่ง หรือ path บ่อย ๆ ผู้ใช้สามารถเขียนเป็นเสมือนกับตัวแปรดังตัวอย่าง

ex)
```
TURBO = c:\TP5\bin
test1.exe:test2.pas test3.tpu test4.tpu
tpc test1 / t$(TURBO)
```

ความหมายก็คือให้สร้างไฟล์ test1.exe และเก็บไว้ที่ c:\tp5\bin

หรือจะกำหนดที่คำสั่ง MSKE ก็ได้ดังตัวอย่าง

```
MAKE _DTURBO=C:\TP5\bin
```

สัญญลักษณ์ต่าง ๆ ที่ใช้ใน Macros

1. $d สัญญลักษณ์ตรวจว่ามีการกำหนด Macros แล้วหรือไม่

ex)
```
!if !$d(TURBO)    (* ถ้ายังไม่มีการกำหนด TURBO *)
TURBO=c:\tp5=bin  (* กำหนดให้ TURBO=C:\tp5\bin
                  *)
!endif
```

2. $* สัญญลักษณ์แทนไฟล์ TARGET โดยไม่มีนามสกุลดังตัวอย่าง

ex)
```
test1.exe:test1.pas test2.tpu
tpc  $*
```

ความหมายก็คือ คอมไพล์ไฟล์ test1.pas เป็นไฟล์ test1.exe โดยใช้ test2.tpu

3. $< สัญญลักษณ์แทนไฟล์ TARGET โดยใช้ชื่อเต็ม
4. $: สัญญลักษณ์แทนไฟล์ PATH ของไฟล์ TARGET
5. $. สัญญลักษณ์แทนไฟล์ TARGET โดยไม่มี PATH
6. $& สัญญลักษณ์แทนไฟล์ TARGET โดยไม่มี PATH และนามสกุล

การสั่งงานคำสั่ง MAKE (Directive)

การสั่งงานคำสั่ง MAKE มีดังนี้คือ

1. !include เป็นคำสั่งอ่านไฟล์ ซึ่งมีรูปแบบดังนี้

   !include "ชื่อไฟล์"

   หรือ !include < ชื่อไฟล์ >
2. !if คำสั่งตรวจ นิพจน์ที่ตามมาเป็นจริง หรือเท็จ
3. !else คำสั่งเช่นเดียวกับ if
4. !elif คำสั่งเช่นเดียวกับ else if
5. !endif คำสั่งสิ้นสุดการใช้เงื่อนไข !if
6. !error คำสั่งพิมพ์ข้อความหลัง !error ในกรณีที่คำสั่ง MAKE ดำเนินการไม่ได้
7. !undef คำสั่งกำหนด Macros โดยสามารถเขียนเป็น !

   undef macro name

   เครื่องหมายต่าง ๆ ที่สามารถใช้ในไฟล์ MAKE มีดังนี้คือ

เครื่องหมายสำหรับพจน์เดียว

-	ลบ
~	สลับค่าในแต่ละบิตของข้อมูล
!	ผลเป็นตรงข้าม (แบบบูลีน)

เครื่องหมายสำหรับนิพจน์ 2 นิพจน์

+	บวก	^	คำสั่ง xor แบบเปรียบเทียบทีละบิต
-	ลบ	&&	คำสั่ง end (ผลเป็นบูลีน)
*	คูณ	||	คำสั่ง and (ผลเป็นบูลีน)
/	หาร	>	มากกว่า
%	เศษ	<	น้อยกว่า
>>	เลื่อนบิตไปทางขวา	>=	มากกว่าหรือเท่ากับ
<<	เลื่อนบิตไปทางซ้าย	<=	น้อยกว่าหรือเท่ากับ
&	คำสั่ง and แบบเปรียบเทียบทีละบิต	==	เท่ากับ
|	คำสั่ง or แบบเปรียบเทียบทีละบิต	!=	ไม่เท่ากับ

เครื่องหมายใช้กับนิพจน์ 3 นิพจน์

แบบ E1 ? E2 : e3

ในกรณี E1 ไม่เท่ากับ 0 ผลจะเป็น E2 มิเช่นนั้นจะเป็น E3

< GREP >

โปรแกรมนี้ทำหน้าที่ค้นหาคำหรือข้อความในไฟล์ ลักษณะทั่วไปสามารถเขียนเป็น

GREP [แบบการค้นหา] คำหรือข้อความ [ชื่อไฟล์]

แบบการค้นหามีดังนี้คือ

-C	แสดงจำนวนบรรทัดที่มีคำหรือข้อความนั้น
-D	แสดงคำในไดเร็กทอรีปัจจุบัน และในไดเร็กทอรีย่อยของไดเร็กทอรีปัจจุบัน
-I	ถือตัวอักษรใหญ่เล็กเหมือนกัน
-L	พิมพ์ชื่อไฟล์ที่มีคำหรือข้อความนั้น
-N	พิมพ์หมายเลขบรรทัดที่มีคำหรือข้อความนั้น
-O	แสดงผลเป็นแบบ UNIC
-R	ตัวอักษรในไฟล์ถือเป็นตัวอักษรธรรมดา (ไม่ถือเป็นเท็กซ์)
-U	กำหนดแบบการค้นหาโดยผู้ใช้ดังตัวอย่าง

ex) -UI ที่จอจะปรากฏช่องใส่ชื่อไฟล์ สมมุติว่าใส่ GREP1 เมื่อใช้ GREP1 ค้นหาคำแบบการค้นจะเป็นการค้นโดยถือเอาตัวอักษรใหญ่เล็กเหมือนกัน

-V พิมพ์บรรทัดที่ไม่มีคำหรือข้อความนั้น

-W ค้นหาคำที่ตรงกันจริง ๆ เช่นให้คำ en ถ้าในไฟล์มีคำว่า end ถือว่าไม่ตรงกัน

-Z แสดงชื่อไฟล์ จำนวนบรรทัดและเลขบรรทัดที่มีคำหรือข้อความนั้น

คำหรือข้อความอาจเขียนให้อยู่ภายในเครื่องหมาย (") ได้ และในกรณีที่ใช้แบบการค้น -R ยังสามารถเขียนสัญญลักษณ์หลัง -R เพื่อกำหนดแบบ การค้นหาได้ดังต่อไปนี้ (ต้องเว้นห่างจาก R 1 ช่องไฟ

^ หาคำหรือข้อความที่จุดเริ่มต้นของบรรทัด

$ หาคำหรือข้อความที่จุดท้ายบรรทัด

. สัญญลักษณ์นี้ถือเป็นอักขระใด ๆ ได้ทุกอักขระ

* ค้นหาคำที่อยู่หน้าเครื่องหมาย * ดังตัวอย่าง ถ้ากำหนด fo* จะค้นหาคำ f fo

+ ค้นหาคำที่อยู่หน้าเครื่องหมาย + ดังตัวอย่าง ถ้ากำหนด fo* จะค้นหาคำ fo foo

[] แทนคำหรือข้อความเช่น [xyz],[a-z] (a ถึง z)

/ คำที่จะให้ค้นหานั้นต้องเป็นตัวอักษรใหญ่หรือเล็ก และนเดอร์บาร์ ในกรณีที่ต้องการค้นหา . หรือ , ต้องใช้ / ใส่หน้า . หรือ ,

< TPC >

การคอมไพล์ภาษา Pascal นอกจากจะใช้ TURBO.EXE แล้ว ยังสามารถใช้โปรแกรม TPC.EXE ได้อีกด้วย คำสั่งนี้จะคอมไพล์โปรแกรมบน DOS และสร้างไฟล์ .exe ซึ่งแบบการใช้งานคือ

TPC [แบบการคอมไพล์] ชื่อไฟล์ [แบบการคอมไพล์]

แบบ	ความหมาย
TPC / $A+ filename	$A}
หรือ TPC filename /$A-	

TPC filename /DIOCHECK, {$DEFINE IOCHECK} {$DEFINE DEBUG	DEBUG}
TPC filename /$M1600,0, 655360	กำหนดขนาดแสตก และฮีพ
TPC filename /M	คอมไพล์โดยคำสั่ง MAKE
TPC filename /B	คอมไพล์โดยคำสั่ง BUILD
TPC filename /Fo:11	ค้นหา error ที่ตำแหน่ง 0:11
TPC filename /L	OPtions/Linker/LinkBuffer memory
TPC /Q filename	ไม่แสดงรายละเอียดการคอมไพล์ออกจอ
TPC /TC:\tp filename	เก็บไฟล์ TPC.XE และ TURBO.TPL ไว้ที่ไดร์ฟ C ซับไดเร็กทอรี tp/bin
TPC /EC:\tp filename	กำหนดไดร์ฟเก็บไฟล์ .EXE และไฟล์ .TPU
TPC /IC:\tp filename	กำหนดไดร์ฟของไฟล์ที่ต้องการอ่าน (include)
TPC /UC:\tp filename	กำหนดไดร์ฟของไฟล์ยูนิต
TPC /OC:=tp filename	กำหนดไดร์ฟของไฟล์ .OBJ
TPC /GS filename	/Options/LInker/Mapfile /segnents
TPC /GP filename	/Options/LInker/Mapfile Publics
TPC /GD filename	/Options/LInker/Mapfile Detailed
TPC /V	สร้างไฟล์ .exe ที่สามารถใช้ Turbo Debugger ตรวจหรือแก้ไขได้

< TPC.CFG >

การกำหนดวิธีการคอมไพล์โดยคำสั่ง TPC.EXE นอกจากจะใช้โดย

ตรงดังที่เห็นหน้าที่แล้ว อาจสร้างไฟล์เพื่อเก็บแบบการคอมไพล์ต่าง ๆ ลงไฟล์ TPC.CFG ดังตัวอย่าง

ex) /IC:\TP\INC (* เก็บไว้ในไฟล์ TPC.CFG *)

เมื่อเรียกใช้ TPC filename

จะมีความหมายเป็น TPC filename /IC:\TP\INC

< TPCONFIG.EXE >

โปรแกรมส่งแบบการเก็บข้อมูลระหว่างไฟล์ .TP กับไฟล์ TPC.CFG โดยมีแบบการใช้งานดังนี้

ex) TPCONFIG TURBO.TP TPC.CFG

จากตัวอย่างข้างบน จะเป็นการส่งแบบการคอมไพล์จากไฟล์ TURBO.TP ให้กับไฟล์ TPC.CFG ในกรณีที่ไม่มีไฟล์ TPC.CFG จะสร้างไฟล์ TPC.CFG ขึ้นใหม่ และสามารถส่งแบบคอมไพล์จากไฟล์ TPC.CFG ให้แก่ TURBO.TP ได้เช่นกัน

< UPGRADE >

โปรแกรมแปลงไฟล์ภาษา Pascal ซึ่งเขียนโดยใช้ Version 3.0 ซึ่งโปรแกรมนี้มีแบบการใช้งานดังนี้

UPGRADE [แบบการแก้ไข]

สำหรับแบบการแก้ไขมีดังนี้คือ

/3 ในกรณีที่ใช้คำสั่งของ Turbo3 จะใส่คำสั่ง uses turbo3 โดยอัตโนมัติ

/J ในกรณีที่ต้องการเขียนรายละเอียดลงไฟล์ ชื่อเดียวกับโปรแกรม .JNL

/n ในกรณีที่ไม่ต้องการแสดงรายละเอียด

/O ในกรณีที่ต้องการบันทึกไฟล์ที่แก้ไขไปยัง PATH ที่ต้องการ ดังตัวอย่าง

ex) UPGRADE test c:\turbo5

/U เป็นการแบ่งไฟล์โปรแกรมเป็นยูนิตย่อย ๆ โดยต้องเขียน .u ชื่อยูนิต ไว้ที่ต้นไฟล์

การสร้างยูนิตเกี่ยวกับคีย์บอร์ด

ในที่นี้จะกล่าวถึงหลักการพื้นฐานเทอร์โบปาสคาลในการอ่านข้อมูลจากแป้นพิมพ์ นอกจากนี้จะกล่าวึงการสร้างยูทิลิตี้ และโปรแกรมในการอ่านข้อมูลที่มีประสิทธิภาพสูงขึ้น เช่น

1. โปรแกรมอ่านตัวอักษรที่กำหนดความยาวของการอ่านได้
2. โปรแกรมอ่านข้อมูลเฉพาะตัวเลข
3. โปรแกรมอ่านข้อมูลที่ใช้ปุ่มลูกศรเลื่อนไปแก้ไขข้อมูลตัวใดก็ได้
4. โปรแกรมตรวจการกดฟังก์ชันคีย์เช่นการกดปุ่ม F1-F10

มีอะไรเกิดขึ้นเมื่อเรากดปุ่ม

แป้นพิมพ์ของไอบีเอ็มพีซีมีไมโครโปรเซสเซอร์ 8048 ทำหน้าที่ในการควบคุม และตรวจรับการกดปุ่ม เมื่อผู้ใช้กดปุ่มใดไม่ว่าจะเป็นปุ่มตัวอักษร หรือกดปุ่มรหัสควบคุม เช่น Ctrl A, Ctrl B ฯลฯ 8048 จะส่งรหัสประจำปุ่มไปยังซีพียูทันที รหัสประจำปุ่มบนแป้นพิมพ์นี้เรียกว่า "สแกนโค้ด" (Scan Code) รหัสสแกนโค้ดแต่ละปุ่มบนแป้นพิมพ์ดังแสดงไว้ในรูปที่2.1

จะสังเกตเห็นได้ว่าปุ่มฟังก์ชันต่าง ๆ เช่น F1-F10, Ins, Del, Home, end, PgUp, PgDn หรือปุ่มลูกศรทั้งสี่ทิศทางก็มีสแกนโค้ดประจำตัวเช่นกัน ดังนั้น วิธีการตรวจสอบปุ่มฟังก์ชันเหล่านี้จึงต้องตรวจจากสแกนโค้ดที่อ่านได้ ส่วนปุ่มฟังก์ชันพิเศษ เช่น ปุ่ม Alt-F1 หรือ Ctrl F1 จะมีสแกนโค้ดพิเศษประจำปุ่มที่เรียกว่า "สแกนโค้ดส่วนขยาย" (Extended Scan Code) ดังตารางที่ 1

1. การตรวจสอบการสแกนโค้ด

เทอร์โบปาสคาล 4.0 มีคำสั่งอ่านค่าตัวอักษรด้วยฟังก์ชัน ReadKey ซึ่งจะอ่านการกดแป้นพิมพ์หนึ่งครั้ง เมื่อเรากดปุ่มอักษรค่าที่ได้คือรหัส 1 ไบต์ของตัวอักษรนั้น แต่ถ้าเรากดปุ่มฟังก์ชันเราจะได้รหัสสองไบต์ติดต่อกัน รหัสไบต์แรกจะเห็นรหัส NULL หรือ #00 ส่วนรหัสไบต์ที่สองเป็นรหัสสแกนโค้ดจึงใช้รหัส #00 เป็นตัวตรวจสอบการกดฟังก์ชันคีย์ได้ กระบวนการในการตรวจสอบการกดปุ่มจึงเป็นดังนี้

```
var Key : char;
    FuncKey :boolaen;
procedure ReadFuncKey(var Key:char);
```

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

รูปที่ 2.1 สแกนโค้ดประจำปุ่ม

Scan codes and characters

Key No. (Dec)	Scan Code (Hex)	Base Case	Upper Case	Ctrl	Alt
1	29	(grave)	(tilde)	N.A.	N.A.
2	2	1	!	N.A.	-1
3	3	2	@	-1	-1
4	4	3	#	N.A.	-1
5	5	4	$	N.A.	-1
6	6	5	%	N.A.	-1
7	7	6	^	RS	-1
8	8	7	&	N.A.	-1
9	9	8	*	N.A.	-1
10	A	9	(	N.A.	-1
11	B	0	)	N.A.	-1

Scan codes and characters

Key No. (Dec)	Scan Code (Hex)	Base Case	Upper Case	Ctrl	Alt
12	C	-(hypen)	-	US	-1
13	D	=	+	N.A.	-1
14	2B	\	¦	FS	N.A.
15	E	Backspace	Backspace	DEL	N.A.
16 Tab	F	Tab	-1	N.A.	N.A.
17	10	q	Q	DCI	-1
18	11	w	W	ETB	-1
19	12	e	E	ENQ	-1
20	13	r	R	DC2	-1
21	14	t	T	DC4	-1
22	15	y	Y	EM	-1
23	16	u	U	NAK	-1
24	17	i	I	HT	-1
25	18	o	O	SI	-1
26	19	p	P	DLE	-1
27	1A	[	{	ESC	-1
28	1B	]	}	GS	N.A.
30	1D	N.A.	N.A.	N.A.	N.A.
31 Ctrl	1E	a	A	SOH	-1
32	1F	s	S	DC3	-1
33	20	d	D	EOT	-1
34	21	f	F	CK	-1
35	22	g	G	BEL	-1
36	23	h	H	BS	-1

Scan codes and characters

Key No. (Dec)		Scan Code (Hex)	Base Case	Upper Case	Ctrl	Alt
37		24	j	J	LF	-1
38		25	k	K	VT	-1
39		26	l	L	FF	-1
40		27	;	:	N.A.	N.A.
41		28	'	"	N.A.	N.A.
43	Enter	1C	CR	CR	LF	N.A.
44	Shift	2A	N.A.	N.A.	N.A.	N.A.
46		2C	z	Z	SUB	-1
47		2D	x	X	CAN	-1
48		2E	c	C	EXT	-1
49		2F	v	V	SYN	-1
50		30	b	B	STX	-1
51		31	n	N	SO	-1
52		32	m	M	CR	-1
53		33	,(comma)	<	N.A.	N.A.
54		34	.(period)	>	N.A.	N.A.
55		35	/	?	N.A.	N.A.
57	Shift	36	N.A.	N.A.	N.A.	N.A.
58	Alt	38	N.A.	N.A.	N.A.	N.A.
61	Space	39	SP	SP	SP	SP
64	Caps Locs	3A	N.A.	N.A.	N.A.	N.A.
65	F2	3C	-1	-1	-1	-1
66	F4	3E	-1	-1	-1	-1
67	F6	40	-1	-1	-1	-1

Key No. (Dec)	Scan Code (Hex)	Base Case	Upper Case	Ctrl	Alt
68 F8	42	-1	-1	-1	-1
69 F10	44	-1	-1	-1	-1
70 F1	3B	-1	-1	-1	-1
71 F3	3D	-1	-1	-1	-1
72 F5	3F	-1	-1	-1	-1
73 F7	41	-1	-1	-1	-1
74 F9	43	-1	-1	-1	-1
90 Esc	1	ESC	ESC	ESC	N.A.
95 Num Lock	45	N.A.	N.A.	-2	N.A.
100 Scroll	46	N.A.	N.A.	-2	N.A.
105 Sys Req	54	-2	N.A.	N.A.	N.A.
106	37	*	-2	-1	N.A.

Scan codes and characters for numeric keypad

Key No. (Dec)	Scan Code (Hex)	Base Casse	Num Lock	Ctrl	Alt
91	47	Hom(-1)	7	Clear Screen	N.A.
92	4B	<-- (-1)	4	Reverse Word (-1)	N.A.
93	4F	End(-1)	1	Erase to EOL (-1)	N.A.
96	48	¦(-1)	8	N.A.	N.A.
97	4C	N.A.	5	N.A.	N.A.
98	50	¦(-1)	2	N.A.	N.A.

Key No. (Dec)	Scan Code (Hex)	Base Case	Num Loc	Ctrl	Alt
99	52	Ins(-1,-2)	0	N.A.	N.A.
100	49	PgUp(-1)	9	Top of Text and Home	N.A.
102	4D	-- (-1)	6	Advance Word (-1)	N.A.
103	51	PgDw(-10	3	Erase to EOS (-1)	N.A.
104	53	Del(-1,-2)	.	-2	-2
107	4A	-(minus)	-	N.A.	N.A.
108	4E	+	+	N.A.	N.A.

กระบวนการนี้จะให้หรัส 1 ไบต์อยู่ในตัวแปร Key ผู้ใช้จะตรวจสอบได้ว่ามีการกดปุ่มใดโดยดูจากแฟล็ก Runckey ถ้า Funckey มีค่า False แสดงว่ามีการกดปุ่มอักษรธรรมดา และตัวแปร Key จะเป็นรหัสแอสกีของตัวอักษรนั้น แต่ถ้า funcKey มีค่าเป็น True แสดงว่ามีการกดปุ่มฟังก์ชันและตัวแปร Key จะเก็บรหัสสแกนโค้ดของตัวอักษรนั้น โปรแกรมที่ 1 เป็นตัวอย่างการเรียกใช้กระบวนการ ReadKey เพื่อตรวจการกดปุ่ม

<u>การตรวจสแกนโค้ดโดยใช้อินเตอร์รัพต์</u>

รอมไบออสภายในเครื่อง มีโปรแกรมทำหน้าที่ตรวจรับการกดแป้นพิมพ์ คืออินเตอร์รพต์หมายเลข $16 ซึ่งมีฟังก์ชันทั้งหมด 3 ฟังก์ชันดังตารางที่ 2 ในที่นี้จะนำฟังก์ชัน 0 ซึ่งเป็นการอ่านคีย์บอร์ดมาใช้ ผลการทำงานของฟังก์ชัน 0 จะได้รหัสตัวอักษรจากการกดปุ่มเก็บไว้ในรีจิสเตอร์ AH และรหัสแอสกีอยู่ใน AL ในกรณีที่เรากดปุ่มฟังก์ชันจะได้ค่าในรีจิสเตอร์ AL เป็น $00 ตัวอย่างการใช้อินเตอร์รัพต์ $16 แสดงได้ดังโปรแกรมที่ 2

โปรแกรมอ่านข้อมูลเฉพาะตัวเลข

คำสั่ง Read และ Readln ในเทอร์โบปาสคาลใช้สำหรับการอ่านข้อมูลตัวเลขหรืออักษรก็ได้ ในกรณีที่ใช้คำสั่งอ่านข้อมูลตัวเลข ถ้าข้อมูลที่ป้อนเข้าไปไม่ถูกต้อง เช่น ป้อนข้อมูลเป็นตัวอักษร หรือใส่รหัสพิเศษเข้าไปจะเกิด Run-time Error ทำให้โปรแกรมหยุดทำงาน เหตุการณ์เช่นนี้อาจเกิดขึ้นได้ทั้งจากความตั้งใจหรือความผิดพลาดจากการป้อน อย่างไรก็ตามโปรแกรมที่ดี ควรสามารถตรวจสอบความถูกต้องของข้อมูล หรือป้องกันความผิดพลาดที่ส่งผลให้โปรแกรมหยุดทำงานได้

เทคนิคการตรวจสอบ และป้องกันความผิดพลาดจากการป้อนข้อมูลทำได้โดยการเขียนกระบวนการเพื่ออ่านข้อมูลขึ้นโดยเฉพาะ กระบวนการนี้จะรับข้อมูลเข้ามาอยู่ในรูปของสตริงแล้วจึงใช้ฟังก์ชันในการแปลงสตริงไปเป็นตัวเลข พร้อมกับส่งรหัสเพื่อบ่งบอกถึงความถูกต้องของการแปลงตามมาด้วย คุณสมบัติทั้งหมดนี้มีอยู่ในกระบวนการดังโปรแกรมที่ 3

โปรแกรมที่ 3 ประกอบด้วยกระบวนการที่สามารถอ่านข้อมูลตัวเลขและตรวจสอบได้ในทันทีว่ากดปุ่มตัวเลขหรือไม่ นอกจากนี้ยังสามารถกำหนดความยาวของตัวเลขที่จะป้อนเข้าไปอีกด้วย เช่น ยอมให้มีการป้อนตัวเลขเพียง 5 หลัก เป็นต้น

โปรแกรมที่ 3 มี 3 กระบวนการสำคัญคือ ReadInt, ReadReal และReadChar ทำหน้าที่หลักดังนี้

- ReadInt เป็นการอ่านตัวเลขจำนวนเต็มการป้อนข้อมูลจึงต้องป้อนเฉพาะตัวเลขเท่านั้น จะไม่รับอักขระอื่นใดเลย เครื่องหมายลบจะป้อนตรงตำแหน่งแรก (จะมีการตรวจว่าผู้ใช้กดเครื่องหมายลบ ณ ตำแหน่งแรกหรือยัง โดยใช้แฟล็กชื่อ MinusFlag เป็นตัวเก็บสถานะ กระบวนการนี้สามารถกำหนดความยาวของการป้อนได้ โดยปรกติแล้วตัวเลขจำนวนเต็มจะมีค่าบวกสูงสุดเท่ากับ 32767 คือความยาว 5 ตัวอักษร และมีค่าลบได้ถึง -32767 รวม 6 ตัวอักษร (นับเครื่องหมายลบด้วย) ดังนั้นการป้อนข้อมูลจึงควรกำหนดความยาวไม่เกิน 6 ตัวอักษร

การเรียกใช้จะต้องมีพารามิเตอร์ 3 ตัว ดังตัวอย่างการเรียกใช้ด้วย ReadInt (N,6,Code) พารามิเตอร์ตัวแรก N เป็นตัวแปรที่ใช้เก็บค่าตัวเลขเมื่อป้อนข้อมูลเสร็จ พารามิเตอร์ตัวที่สองในที่นี้คือ 6 หมายถึงความยาวของตัวอักษรที่ป้อนได้ พารามิเตอร์ตัวที่สามคือ Code เป็นตัวเก็บรหัสที่บ่งบอกผลการป้อนตัวเลข

ถ้าการป้อนตัวเลขเป็นไปอย่างถูกต้องตัวแปร N จะเก็บตัวเลขจำนวนเต็มที่ป้อน และตัวแปร Code จะมีค่าเป็น 0 แต่ถ้าป้อนไม่ถูกต้องจะถือว่าค่าในตัวแปร N ไม่นิยามคือ เป็นค่าที่นำไปใช้ไม่ได้ ส่วนตัวแปร Code จะมีค่าเป็น 255

บ่งบอกว่าเกิดความผิดพลาดขึ้น และในกรณีที่ผู้ใช้กดปุ่มรีเทอร์นโดยไม่ป้อนอะไรเลย หรือกดปุ่ม ESC จะถือว่าเป็นการยกเลิกการป้อนข้อมูล ซึ่งในกรณีนี้ตัวแปร Code จะมีค่าเป็น 1 ผลของค่าตัวเลข และรหัสนี้ได้สรุปไว้ดังตารางที่ 3

- ReadReal มีหลักการทำงานคล้ายกับ ReadInt แต่ใช้สำหรับการอ่านตัวเลขจำนวนจริง ช่วงของตัวเลขจำนวนจริงที่อ่านได้เป็นไปตามข้อกำหนดของเทอร์โบคือ -1E-38 ถึง 1E+38 การป้อนตัวเลขจำนวนจริงอาจมีทศนิยมประกอบอยู่ด้วย และป้อนได้หลายแบบ เช่น เมื่อต้องการป้อนตัวเลข 0.45 สามารถป้อนด้วยการพิมพ์ 0.45 หรือ .45 ก็ได้การ ReadReal มีข้อจำกัดตรงที่ไม่สามารถป้อนตัวเลขในรูปของเลขวิทยาศาสตร์ได้ กล่าวคือไม่อาจป้อนในรูปสัญลักษณ์ E เช่น 3E-08 แต่จะต้องป้อนในรูปทศนิยมเสมอ

การเรียกใช้กระบวนการนี้ต้องมีพารามิเตอร์ 3 ตัว เช่นเดียวกับการ ReadInt ส่วนที่แตกต่างกันก็คือพารามิเตอร์ตัวแรกต้องเป็นเลขจำนวนจริง สำหรับรหัสที่บ่งบอกความผิดพลาดจะมีรูปแบบเหมือน ReadInt ในตารางที่ 3

- ReadChar เป็นกระบวนการการอ่านข้อความที่สามารถกำหนดความยาวตัวอักษรที่ต้องการอ่านได้ ตัวอย่างเช่น เมื่อเรียกใช้คำสั่ง ReadCahr(C,10) หมายถึงการอ่านสตริงเข้าไปเก็บในตัวแปร C โดยให้มีความยาวของสตริงที่ป้อนได้ไม่เกิน 10 ตัว ตัวแปร C จะต้องมีแบบข้อมูล (Type) เป็นสตริง

เมื่อกดปุ่ม Return โดยไม่ป้อนตัวอักษรใดเลย พารามิเตอร์ที่ส่งกลับไปจะมีค่าเท่ากับ #13 คือรหัสแอสกีของ Return และในกรณีที่กดปุ่ม ESC พารามิเตอร์ที่ส่งกลับมีค่าเท่ากับ #27 เป็นรหัสแอสกีของ ESC เช่นกัน

กระบวนการทั้งสามนี้จะมีคุณสมบัติอื่นโดยทั่วไปคล้ายกับคำสั่ง Read กล่าวต่อภายหลังเสร็จสิ้นการอ่านแล้วเคอร์เซอร์จะไม่ขึ้นบรรทัดใหม่ กระบวนการเหล่านี้จะมีประโยชน์อย่างมากในการตรวจสอบการป้อนข้อมูล

2. โปรแกรมอ่านข้อมูลที่ใช้ปุ่มลูกศรเลื่อนไปแก้ไขได้

กระบวนการ Read และ ReadIn ในปาสคาล หรือ ReadInt, ReadReal และ ReadChar ที่สร้างขึ้นมีลักษณะการแก้ไขข้อมูลเหมือนกัน คือใช้ได้แต่เพียงปุ่ม BackSpace ลยถอยหลัง หรือลบตัวอักษรที่อยู่ทางซ้ายของเคอร์เซอร์เท่านั้น และก็ลบได้จากตัวอักษรที่อยู่หลังสุด ผู้ใช้ไม่สามารถเลือกตำแหน่งของการลบที่จุดใด ๆ ได้

สแกนโค้ด

ฐานสิบหก	ฐานสิบ	ปุ่ม
#3	3	Ctrl Z
$F	10	Shift Tab
$10-$19	16-25	Alt Q ถึง Alt P
$1E-$26	30-38	Alt A ถึง Alt L
$2C-$32	43-50	Alt Z ถึง Alt M
$54-$5D	84-93	Shift F1 ถึง Shift F10
$5E-$67	94-103	Ctrl F1 ถึง Ctrl F10
$68-$71	104-113	Alt F1 ถึง Alt F10
$27	112	Ctrl Prtsc
$73	113	Ctrl <--
$74	114	Ctrl -->
$75	115	Ctrl End
$76	116	Ctrl PgDn
$77	117	Ctrl Home
$78-$83	118-131	Alt 1 ถึง Alt =
$84	132	Ctrl PgUp

ตารางที่ 1 สแกนโค้ดส่วนขยาย

ฟังก์ชัน	การทำงาน	ค่าที่ได้
0	อ่านอักขระจากบัฟเฟอร์	AL = รหัสแอสกี AH = สแกนโค้ด
1	ตรวจสอบดูว่ามีอักขระอยู่ในบัฟเฟอร์หรือไม่ ฟังก์ชันจะไม่รอคอยการกดแป้นพิมพ์	ถ้ามี ZF = 0 AL = รหัสแอสกี AH = สแกนโค้ด ถ้าไม่มี ZF = 1
2	อ่านสถานภาพของการกดคีย์บอร์ด	บิตต่าง ๆ ในรีจิสเตอร์ AL จะแสดงสภาวะต่อไปนี้ เมื่อบิตนั้นมีค่า "1" ดังนี้ บิต 7 มีการกด Ins 6 มีการกด CapLock 5 มีการกด NumLock 4 มีการกด ScrollLock 3 มีการกด Alt-Shift 2 มีการกด Ctrl-Shift 1 Shift ด้านซ้ายถูกกด 0 Shift ด้านขวาถูกกด

ตารางที่ 2 ฟังก์ชันของคีย์บอร์ดอินเตอร์รัพต์

วิธีการสร้างความยืดหยุ่นตัว และเพิ่มความสะดวกในการป้อนข้อมูลที่ดีขึ้นอาจทำได้โดยนำหลักการของฟูลสกรีนเอดิเตอร์มาประยุกต์ใช้ ในระบบฟูลสกรีนผู้ใช้สามารถเลื่อนเคอร์เซอร์ไปแก้ไข ลบ หรือแทรกตัวอักษรที่จุดใดของจอภาพ นอกจากนี้อาจมีฟังก์ชันพิเศษในการย้ายตำแหน่งของเคอร์เซอร์อย่างรวดเร็ว หรือฟังก์ชันช่วยในการลบอักษรข้อความได้อย่างมีประสิทธิภาพด้วย

การป้อนข้อมูลที่มีลักษณะคล้ายกับฟูลสกรีนเอดิเตอร์จึงอำนวยความสะ-

ดวกต่อผู้ใช้เป็นอย่างมาก สามารถพัฒนาขึ้นเป็นกระบวนการได้โดยง่ายในที่นี้จะออกแบบกระบวนการการป้อนข้อมูลให้มีคุณสมบัติดังนี้

- สามารถใช้ปุ่มลูกศรซ้ายขวา เลื่อนไปที่ตำแหน่งตัวอักษรใด ๆ ก็ได้ในบรรทัดนั้น
- ตัวอักษรที่ป้อนได้มีความยาวไม่เกินขนาดวินโดว์ที่ใช้งานในขณะนั้น
- การป้อนจะอยู่ในสภาวะ Insert หรือแทรกตัวอักษรตลอดเวลา กล่าวคือถ้ามีตัวอักษรอื่นอยู่ทางขวาของตำแหน่งเคอร์เซอร์ ตัวอักษรใหม่ที่พิมพ์เข้าไปจะแทรกที่ตำแหน่งเคอร์เซอร์ และดันตัวอักษรทางขวาออกไป
- สามารถใช้ปุ่ม BackSpace หรือปุ่ม Ctrl H เพื่อลบอักษรทางซ้ายที่จุดใดก็ได้
- สามารถใช้ปุ่ม Del หรือ ปุ่ม Ctrl G เพื่อลดตัวอักษรที่ตำแหน่งเคอร์เซอร์ได้
- สามารถใช้ปุ่ม home หรือปุ่ม Ctrl A เพื่อเลื่อนเคอร์เซอร์อย่างรวดเร็วไปยังอักษรตัวแรกได้
- สามารถใช้ปุ่ม End หรือปุ่ม Ctrl Z เพื่อเลื่อนเคอร์เซอร์อย่างรวดเร็วไปที่ตำแหน่งหลังอักษรตัวสุดท้าย
- สามารถใช้ปุ่ม Ctrl T เพื่อลบตัวอักษรตั้งแต่ตำแหน่งเคอร์เซอร์จนถึงอักษรตัวสุดท้ายได้
- สามารถใช้ปุ่ม Ctrl Y ลบตัวอักษรทั้งหมด หรือยกเลิกเพื่อเริ่มพิมพ์ใหม่ได้

การป้อน	พารามิเตอร์	พารามิเตอร์ Code
การป้อนตัวเลขถูกต้อง	เก็บค่าตัวเลขนั้น	0
ป้อนตัวเลขไม่ถูกต้อง	ไม่นิยาม	255
กดปุ่ม RETURN โดยไม่ป้ออะไร	ไม่นิยาม	1
กดปุ่ม ESC	ไม่นิยาม	1

ตารางที่ 3 พารามิเตอร์แสดงผลการป้อนของ ReadInt และ ReadReal

โปรแกรมที่ 4 ประกอบด้วยกระบวนการหลัก 3 ส่วนซึ่งมีคุณสมบัติดังที่

กล่าวมาแล้วทั้งหมด ได้แก่ ReadIntNum, ReadRealNum และ ReadString กระบวนการทั้ง 3 มีการเรียกใช้ และส่งผ่านพารามิเตอร์เช่นเดียวกับ ReadInt, และ ReadReal และ ReadCahr ตามลำดับ นอกจากนี้กระบวนการ ReadIntNum และ ReadRealNum ยังให้รหัสการทำงานเหมือนตารางที่ 3 อีกด้วย

3. <u>การสร้างยูนิต KEYBOARD.PAS</u>

```
UNIT KEYBOARD;
INTERFACE
USES DOS,CRT;

VAR       CH          :CHAR;
          FUNCKEY     :BOOLEAN;
          FINISHED    :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;
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;
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;
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 READFUNCKEY(VAR KEY :CHAR);
    PROCEDURE TESTFUNCKEY(KEY :CHAR);
    {END OF INTERFACE SECTION}
    IMPLEMENTATION

PROCEDURE READFUNCKEY (VAR KEY:CHAR);

 BEGIN
  KEY := READKEY;
  IF KEY = #0 THEN
     BEGIN
      FUNCKEY := TRUE;
      KEY     := READKEY;
     END
  ELSE
      FUNCKEY := FALSE;
 END;
```

```
PROCEDURE TESTFUNCKEY (KEY :CHAR);
 BEGIN
   CASE  (KEY) OF
         F1_KEY          : WRITELN ('F1');
         F2_KEY          : WRITELN ('F2');
         F3_KEY          : WRITELN ('F3');
         F4_KEY          : WRITELN ('F4');
         F5_KEY          : WRITELN ('F5');
         F6_KEY          : WRITELN ('F6');
         F7_KEY          : WRITELN ('F7');
         F8_KEY          : WRITELN ('F8');
         F9_KEY          : WRITELN ('F9');
         F10_KEY         : WRITELN ('F10');
         HOME_KEY        : WRITELN ('Home');
         UP_KEY          : WRITELN ('Up Arrow');
         PGUP_KEY        : WRITELN ('Pg Up');
         LT_KEY          : WRITELN ('Left Arrow');
         RT_KEY          : WRITELN ('Right Arrow');
         END_KEY         : WRITELN ('End');
         DN_KEY          : WRITELN ('Dn Arrow');
         PGDN_KEY        : WRITELN ('Pg Dn');
         INS_KEY         : WRITELN ('Ins');
         DEL_KEY         : WRITELN ('Del');
         ALT_F1_KEY      : WRITELN ('Alt F1');
         ALT_F2_KEY      : WRITELN ('Alt F2');
         ALT_F3_KEY      : WRITELN ('Alt F3');
         ALT_F4_KEY      : WRITELN ('Alt F4');
         ALT_F5_KEY      : WRITELN ('Alt F5');
         ALT_F6_KEY      : WRITELN ('Alt F6');
         ALT_F7_KEY      : WRITELN ('Alt F7');
         ALT_F8_KEY      : WRITELN ('Alt F8');
         ALT_F9_KEY      : WRITELN ('Alt F9');
         ALT_F10_KEY     : WRITELN ('Alt F10');
```

```
            CTRL_F1_KEY    : WRITELN ('Ctrl F1');
            CTRL_F2_KEY    : WRITELN ('Ctrl F2');
            CTRL_F3_KEY    : WRITELN ('Ctrl F3');
            CTRL_F4_KEY    : WRITELN ('Ctrl F4');
            CTRL_F5_KEY    : WRITELN ('Ctrl F5');
            CTRL_F6_KEY    : WRITELN ('Ctrl F6');
            CTRL_F7_KEY    : WRITELN ('Ctrl F7');
            CTRL_F8_KEY    : WRITELN ('Ctrl F8');
            CTRL_F9_KEY    : WRITELN ('Ctrl F9');
            CTRL_F10_KEY   : WRITELN ('Ctrl F10');
        END;
    END;
END.
```

การสร้างยูนิตเกี่ยวกับจอภาพ

จอภาพเป็นอุปกรณ์เอาต์พุตที่นับว่ามีความสำคัญสูง เปรียบเสมือนหน้าต่างบานแรกที่เปิดเชื่อมระหว่างผู้ใช้กับคอมพิวเตอร์ ผู้ใช้จะเกิดความประทับใจขณะใช้คอมพิวเตอร์มากเพียงไรก็ขึ้นอยู่กับรูปแบบของผลงานบนจอภาพมิใช่น้อย ดังนั้นผู้พัฒนาโปรแกรม หรือยูทิลิตี้ต่าง ๆ จึงเน้นความน่าใช้ของซอฟต์แวร์ด้วยรูปแบบของการแสดงผลที่สวยงาม ตลอดจนความเร็วในการแสดงผลที่รวดเร็ว

เนื้อหาในบทนี้จึงเป็นการกล่าวถึงแนวทางการพัฒนาเครื่องมือและยูทิลิตี้ในการแสดงผลตัวอักษรบนจอภาพทั้งในแง่ของการควบคุมรูปแบบการแสดงผลที่สวยงาม และการแสดงผลด้วยความเร็วสูง

จอภาพของไอบีเอ็มพีซี

เริ่มต้นเมื่อไอบีเอ็มพีซีออกสู่ตลาดใหม่ ๆ ไอบีเอ็มได้ออกการ์ดสำหรับเชื่อมต่อจอภาพไว้ 2 ชนิด คือการ์ดจอโมโนโครม และการ์ดจอสี จนกระทั่งทุกวันนี้มีการ์ดแสดงผลสมรรถนะสูงออกมา ทั้งที่สามารถอีมูเลตตัวเองเป็นทั้งจอโมโนโครมและจอสี คือการ์ดอีจีเอ (Eng\hanced Graphics Adapter)

นอกจากนั้นการ์ดเชื่อมต่อจอภาพที่เป็นมาตรฐานอีกการ์ดหนึ่งได้แก่ การ์ดเฮอร์คิวลิส ซึ่งเป็นการ์ดที่ได้รับความนิยมสูงสุด สำหรับใช้กับจอโมโนโครมและทำให้แสดงผลกราฟิกบนจอโมโนโครมได้ด้วยความละเอียดสูงถึง 720 x 348 จุด

การตรวจชนิดของจอภาพ

แม้ว่าไอบีเอ็มพีวีจะใช้งานร่วมกับการ์ดจอภาพได้หลายแบบ หลายชนิด แต่ซอฟต์แวร์ต่าง ๆ ก็ยังสามารถทำงานร่วมกับการ์ดจอภาพแต่ละชนิดได้ ทั้งนี้เนื่องจากรอมไบออสมีฟังก์ชันการตรวจสอบชนิดจอภาพที่ต่อเชื่อมอยู่ ซอฟต์แวร์จะใช้ข้อมูลที่ได้จากไบออสทำการแยกแยะชนิดของจอภาพ ทำให้ซอฟต์แวร์ทำงานร่วมกับฮาร์ดแวร์ชนิดต่าง ๆ ได้อย่างถูกต้อง

การตรวจสอบชนิดของจอภาพที่ติดตั้งอยู่ ทำได้โดยเรียกอินเตอร์รัพต์ของไบออสนอกจากนี้ยังสามารถตรวจสอบชนิดของจอภาพโดยติดต่อกับหน่วยความจำบริเวณหนึ่งที่เรียกว่า "พื้นที่ข้อมูลไบออส" (Bios Data Area) ซึ่งจะกล่าวในหัวข้อถัดไป

สำหรับการสแดงผลตัวอักษรด้วยความเร็วสูงทำได้โดยไม่จำเป็นต้อง

เรียกผ่านไบออส แต่ใช้วิธีติดต่อกับตัวจอภาพโดยตรงทำให้การแสดงผลเป็นไปด้วยความเร็วสูงมาก ซึ่งเป็นวิธีที่ซอฟต์แวร์มีชื่อเสียงทั้งหลาย เช่น ดีเบสทรี โลตัส เทอร์โบซี ฯลฯ ได้พัฒนายูทิลิตี้นี้ขึ้นมาเช่นกันแต่ผู้อ่านจำเป็นต้องเรียนรู้ถึงระบบควบคุมการแสดงผลของจอภาพเสียก่อน

1. <u>วิดีโอแรม</u>

วิธีแสดงอักขระออกจอภาพที่นิยมใช้ คือจัดหน่วยความจำบริเวณหนึ่งให้เป็นหน่วยความจำจอภาพเรียกว่า "วิดีโอแรม" (Video RAM) ถ้านำรหัสตัวอักษรใดก็ตามมาไว้ที่หน่วนความจำบริเวณนี้ จะทำให้เกิดตัวอักษรนั้นบนจอภาพทันที

ผู้ออกแบบไอบีเอ็มพีซีก็ใช้หลักการของวิพีโอแรมนี้เองก็คงต้องมาหาคำตอบว่าหน่วยความจำนี้อยู่ที่ใด หรือถ้าใช้จอโมโนโครม กับจอสีแล้วตำแหน่งหน่วยความจำจะอยู่ที่เดียวกันหรือไม่

โดยปรกติวิดีโอแรมของจอภาพต่างชนิดกันจะอยู่คนละตำแหน่งกันบนจอ โมโนโครมมีจุดเริ่มต้นของวิดีโอแรมที่แอดเดรส $B000:$0000 ส่วนวิดีโอแรมของจอสีจะเริ่มต้นที่แอดเดรส $B800:$0000 เป็นต้นไป

ขนาดของวิดีโอแรมสามารถคำนวณได้จากขนาดของจอภาพ เนื่องจากจอภาพโมโนโครมมีขนาด 80 x 25 หมายถึงแสดงผลได้ 25 บรรทัด ๆ ละ 80 ตัวอักษร จำนวนตัวอักษรทั้งหมดบนหน้าจอจึงมีจำนวน 80 x 25 = 2000 ตัวอักษร ขนาดของวิดีโอแรมที่ใช้จึงควรมีขนาด 2000 ไบต์ด้วยคือตั้งแต่ $B000:$0000 ถึง $B000:$07CF แต่ความจริงไม่เป็นเช่นนั้น เนื่องจากตัวอักษรหนึ่งตัวจะใช้เนื้อที่หน่วยความจำในวิดีโอแรมถึง 2 ไบต์ ดังนั้นวิดีโอแรมจึงมีขนาด 4000 ไบต์ ตั้งแต่ $B000:$0000 ถึง $B000:$0F9F สาเหตุที่ตัวอักษรหนึ่งตัวกินเนื้อที่ถึง 2 ไบต์นั้น เพราะว่าผู้ออกแบบได้เพิ่มความสามารถในการแสดงผลตัวอักษรแบบต่างๆ เช่น ตัวกระพริบ ขีดเส้นใต้ ตัวรีเวอร์ส ฯลฯ ตัวอักษรแต่ละตัวจึงใช้เนื้อที่อีก 1 ไบต์ เพื่อบ่งบอกถึงลักษณะเฉพาะของตัวเองที่เรียกว่า "แอตทริบิวต์" (Attribute) แอตทริบิวต์จะมีตำแหน่งติดกับรหัสตัวอักษร ยกตัวอย่าง เช่น สำหรับจอโมโนโครม ตัวอักษรตัวแรกบนจอภาพที่มุมบนซ้ายสุดใช้วิดีโอแรมที่ $B000:$0000 แอตทริบิวต์ของอักษรตัวนี้อยู่ที่ $B000:$0001 ตัวอักษรตัวที่สองจะอยู่ที่ $B000:$0002 และแอตทริบิวต์ที่ $B000:$0003 เช่นนี้ไปเรื่อย ๆ สังเกตได้ว่าแอดเดรสไบต์คู่จะใช้เก็บรหัสตัวอักษร ส่วนแอดเดรสไบต์คี่จะใช้เก็บแอตทริบิวต์ของตัวอักษรตำแหน่งของตัวอักษรบนจอภาพที่ตรงกับแอดเดรสในวิดีโอแรมได้แสดงไว้ในรูปที่2.2

	$00	$02	$04	$06	$08	$0A	$9E
$B000:0000	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(80,1)
$B000:00A0	(1,2)						
$B000:0140	(1,3)						
$B000:01E0	(1,4)						
$B000:0280	(1,5)						
$B000:0320	(1,6)						
$B000:03C0	(1,7)						
$B000:0460	(1,8)						
$B000:0500	(1,9)						
$B000:05A0	(1,10)						
$B000:0640	(1,11)						
$B000:06E0	(1,12)						
$B000:0780	(1,13)						
$B000:0820	(1.14)						
$B000:08C0	(1,15)						
$B000:0960	(1,16)						
$B000:0A00	(1,17)						
$B000:0AA0	(1,18)						
$B000:0B40	(1,19)						
$B000:0BE0	(1,20)						
$B000:0C80	(1,21)						
$B000:0D20	(1,22)						
$B000:0DC0	(1,23)						
$B000:0E60	(1,24)						
$B000:0F00	(1,25)						(80,25)

รูปที่ 2.2 แผนภาพแสดงตำแหน่งอักษรบนจอภาพที่ตรงกับตำแหน่งหน่วยความจำวิดีโอแรมของจอโมโนโครมขนาด 80 x 25

	$00	02	$04	$06	$08	$0A	$9E
$B800:0000	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(80,1)
$B800:00A0	(1,2)						
$B800:0140	(1,3)						
$B800:01E0	(1,4)						
$B800:0280	(1,5)						
$B800:0320	(1,6)						
$B800:03C0	(1,7)						
$B800:0460	(1,8)						
$B800:0500	(1,9)						
$B000:05A0	(1,10)						
$B800:0640	(1,11)						
$B800:06E0	(1,12)						
$B800:0780	(1,13)						
$B800:0820	(1,14)						
$B800:08C0	(1,15)						
$B800:0960	(1,16)						
$B800:0A00	(1,17)						
$B800:0AA0	(1,18)						
$B800:0B40	(1,19)						
$B800:0BE0	(1,20)						
$B800:0C80	(1,21)						
$B800:0D20	(1,22)						
$B800:0DC0	(1,23)						
$B800:0E60	(1,24)						
$B800:0F00	(1,25)						(80,25)

รูปที่ 2.3 แผนภาพแสดงตำแหน่งอักษรบนจอภาพที่ตรงกับตำแหน่งหน่วยความจำวิดีโอแรมของจอสีขนาด 80 x 25

	$00	$02	$04	$06	$08	$0A	$4E
$B800:0000	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(40,1)
$B800:0050	(1,2)						
$B800:00A0	(1,3)						
$B800:00F0	(1,4)						
$B800:0140	(1,5)						
$B800:0190	(1,6)						
$B800:01E0	(1,7)						
$B800:0230	(1,8)						
$B800:0280	(1,9)						
$B800:02D0	(1,10)						
$B800:0320	(1,11)						
$B800:0370	(1,12)						
$B800:03C0	(1,13)						
$B800:0410	(1,14)						
$B800:0460	(1,15)						
$B800:04B8	(1,16)						
$B800:0500	(1,17)						
$B800:0550	(1,18)						
$B800:05A0	(1,19)						
$B800:05F0	(1,20)						
$B800:0640	(1,21)						
$B800:0690	(1,22)						
$B800:06E0	(1,23)						
$B800:0730	(1,24)						
$B800:0780	(1,25)						(40,25)

รูปที่ 2.4 แผนภาพแสดงตำแหน่งอักษรบนจอภาพที่ตรงกับตำแหน่งหน่วยความจำวิดีโอแรมของจอสีขนาด 40 x 25

สำหรับจอสีจะมีหลักการเช่นเดียวกันเพียงแต่จุดเริ่มต้นอยู่ที่ $B800:$0000 ไปจนถึง B800L$0F9F ดังรูปที่ 2.3อย่างไรก็ตามการแสดงผลบนจอสีสามารถปรับเป็นขนาด 40 x 25 ได้ การใช้วิดีโอแรมจึงสิ้นสุดที่ $B800:$07CF เท่านั้น ดังรูปที่ 2.4ส่วนการแสดงผลในโหมดกราฟิกนั้นรูปแบบการจัดวิดีโอแรมจะแตกต่างกัน ซึ่งจะได้กล่าวในบทต่อไป

2. แอตทริบิวต์ไบต์

แอตทริบิวต์เป็นสิ่งบ่งบอกถึงลักษระการแสดงผลของตัวอักษร แต่ละบิตในแอตทริบิวต์ไบต์จะมีความหมายเฉพาะตัว สำหรับจอภาพโมโนโครม จะเป็นแบบการแสดงผลรีเวอร์ส ขีดเส้นใต้ กะพริบ หรือความเข้มในขณะที่แอตทริบิวต์จอสีจะบ่งบอกถึงสีของตัวอักษร ความหมายของแต่ละบิตในแอตทริบิวต์ ของจอโมโนโครมและจอสีแสดงไว้ ดังรูปที่ 2.5 และ 2.6 ตามลำดับ

B1	XXX	I	XXU	แอตทริบิวต์
0	000	0	000	ไม่แสดงผลตัวอักษร
0	000	0	111	ตัวอักษรจาง
0	000	1	111	ตัวอักษรเข้ม
0	000	0	001	ตัวจางขีดเส้นใต้
0	000	1	001	ตัวเข้มขีดเส้นใต้
0	111	0	000	ตัวจางรีเวอร์ส
0	111	1	000	ตัวเข้มรีเวอร์ส
1	000	0	111	ตัวจางกะพริบ
1	000	1	111	ตัวเข้มกะพริบ
1	000	0	001	ตัวจางขีดเส้นใต้กะพริบ
1	000	1	001	ตัวเข้มขีดเส้นใต้กะพริบ
1	111	0	000	ตัวจางรีเวอร์สกะพริบ
1	111	1	000	ตัวเข้มรีเวอร์สกะพริบ

รูปที่ 2.5 แอตทริบิวต์ ของจอโมโนโครม

I	R	G	B	แอตทริบิวต์
0	0	0	0	ดำ
0	0	0	1	น้ำเงิน
0	0	1	0	เขียว
0	0	1	1	น้ำทะเล
0	1	0	0	แดง
0	1	0	1	บานเย็น
0	1	1	0	น้ำตาล
0	1	1	1	เทาอ่อน
1	0	0	0	เทาแก่
1	0	0	1	น้ำเงินสด
1	0	1	0	เขียวสด
1	0	1	1	น้ำทะเลสด
1	1	0	0	แดงสด
1	1	0	1	บานเย็นสด
1	1	1	0	น้ำตาลสด
1	1	1	1	ขาว

รูปที่ 2.6 แอตทริบิวต์ไบต์ของจอสี

3. <u>การเขียนโปรแกรมเชื่อมต่อกับวิดีโอแรม</u>

การแสดงผลโดยเชื่อมต่อกับวิดีโอแรมโดยตรงเป็นวิธีแสดงผลรวดเร็วที่สุด และเป็นวิธีที่ซอฟต์แวร์ที่มีความสามารถสูงนิยมใช้กัน เนื่องจากวิดีโอแรมเป็นส่วนหนึ่งของหน่วยความจำ การติดต่อกับวิดีโอแรมด้วยเทอร์โบปาสคาลจึงทำได้โดยสร้างตัวแปรแบบแอบโซลุตที่วิดีโอแรม ตัวแปรแอบโซลุตจะเป็นอะเรย์ของตัวอักษรบนจอภาพด้วยการนิยามดังนี้

```
type  Screen_Array  =  array[1..25,1..80,1..2] of char;
      Mono_Screen:   Screen_Array Absolute  $B000:$0000;

      Colo_Screen:   Screen_Array Absolute  $B800:$0000;
```

การกำหนดตัวแปรจอภาพตามแบบข้างต้น จะช่วยให้ผู้ใช้ติดต่อกับวิดีโอแรมได้โดยตรง ตัวอย่างเช่น ถ้าต้องการแสดงตัวอักษร "A" ให้ปรากฏในบรรทัดที่ 12 คอลัมน์ที่ 1 ของจอโมโนโครม และมีแอตทริบิวต์ตัวเข้มสามารถทำได้โดยใช้คำสั่ง

```
Mono_Screen[12,1,1]: = 'A';
Mono_Screen[12,1,2]: = #15;
```

เมื่อ #15 เป็นแอตทริบิวต์ตัวอักษรเข้มอย่างไรก็ตามสิ่งที่จำเป็นต้องทราบก็คือ เมื่อไรควรจะใช้ monoscreen หรือ colorscreen ซอฟต์แวร์ที่เขียนขึ้นควรตรวจสอบได้ว่า จอภาพที่ใช้ในขณะนั้นเป็นจอชนิดใด วิธีตรวจสอบนี้จะตรวจได้จากพื้นที่ข้อมูลไบออสตำแหน่งที่ $0040:$0049 ดังโปรแกรมทางขวามือ

แอดเดรส $0040:$0049 ถ้ามีค่าเท่ากับ 7 หมายถึงกำลังใช้จอโมโนโครม แต่ถ้าใช้จอสีค่าของไบออสจะเท่ากับ 3

```
var CRT_Type : byte Absolute $0040:$0049;
begin
   if (CRT_Type = 7) then
     begin
        Mono_Screen[12,1,1] := 'A';
        Mono_Screen[12,1,2] := #15;
     end
  else
     begin
        Colo_Screen[12,1,1] := 'A';
        Colo_Screen[12,1,2] := #15;
     end;
end;
```

<u>การแสดงผลตัวอักษรทั้งจอภาพ</u>

การเชื่อมต่อกับวิดีโอแรมโดยตรง ทำให้ผู้ใช้สร้างโปรแกรมแสดงผลได้รวดเร็วใกล้เคียงกับความเร็วในการเคลื่อนย้ายข้อมูลจากตำแหน่งหนึ่งไปอีกตำ-

แหน่งหนึ่ง วิธีเช่นนี้ใช้สำหรับการปรับข้อความบนจอภาพทั้งจอให้เปลี่ยนแปลงในทันทีทันใด

ตัวอย่างการปรับข้อความบนจอภาพได้แสดงไว้ดังโปรแกรมที่ 1 โดยกำหนดค่าคงที่ ScreenData เป็นอะเรย์ 2 มิติขนาด 25 x 80 เท่ากับขนาดของจำนวนบรรทัดคูณจำนวนอักษรแล้วใส่ข้อความไว้

โปรแกรมที่ 1 จะเคลื่อนย้ายอักษรจากตัวแปร ScreenData เข้าสู่วิดีโอแรมทีละตัวตามด้วยแอตทริบิวต์ที่กำหนด จนกระทั่งอักษรตัวสุดท้ายถูกส่งไปความเร็วการแสดงผลข้อมูลต่อหนึ่งหน้าจอกินเวลาประมาณ 0.1 วินาที (จับเวลาบนเครื่องไอบีเอ็มเอทีทำงานที่ความถี่ 8 MHz)

โปรแกรมที่ 1 โปรแกรมแสดงผลข้อมูลทั้งจอ

```
Program UpdatesScreen;
Uses    Screen80    =  array [1..25,1..80,1..2] of char;
        ScreenArray = array [1..25] of String[80];

var     MonoScreen : Screen80 Absolute $B000:$0000;
        ColoScreen : Screen80 Absolute $B800:$0000;
Const   ScreenData : ScreenArray  =

KKK      KKK  MMMMMM      MMMMMM  IIIIIIIIII   TTTTTTTTTT
KKK     KKK   MMM  MM    MM  MMM      III          TTT
KKK    KKK    MMM   MM MM   MMM      III          TTT
KKK   KKK     MMM    MMM    MMM      III          TTT
KKKKKK        MMM    MMM    MMM      III          TTT
KKKKK         MMM           MMM      III          TTT
KKKKKK        MMM           MMM      III          TTT
KKK   KKK     MMM           MMM      III          TTT
KKK    KKK    MMM           MMM      III          TTT
KKK     KKK   MMM           MMM  IIIIIIIIII       TTT

**************************************************
**************************************************
 ;
```

```
var CrtType     : byte     Absolute $0040:$0049;
    X,Y,Len     : byte;

begin
  Clrscr;
  for Y   :=1 to 25 do
   begin
     Len  := length(ScreenData[Y]);
     for X:=1 to len do
       if  (CrtType = 7) then
          begin
            MonoScreen[Y,X,1] :=ScreenData[Y][X];
            MonoScreen[Y,X,2] :=#15;  { Attribute }
          end
      else
          begin
            ColoScreen[Y,X,1] :=ScreenData[Y][X];
            ColoScreen[Y,X,2] :=#15;  { Attribute }
          end;
   end;
end.
```

4. <u>การสร้างไฟล์ของจอภาพ</u>

การประยุกต์ใช้งานจริงสำหรับการแสดงอักษรทั้งจอภาพ มักเป็นโปรแกรมประเภทสไลด์โชว์ หรือโปรแกรม CAI ที่ต้องการแสดงข้อความเป็นหน้า ๆ ซึ่งจะออกแบบข้อความบนจอภาพแล้วบันทึกเก็บเป็นไฟล์ จากนั้นจึงอ่านไฟล์มาไว้ในตัวแปรแล้วนำข้อความในตัวแปรส่งไปยังวิดีโอแรม ตัวอย่างโปรแกรมในกรณีนี้จึงแบ่งออกเป็น 2 ส่วน คือ ส่วนแรกเป็นโปรแกรมสร้างไฟล์ของจอภาพดังโปรแกรมที่ 2 ส่วนที่สองเป็นโปรแกรมอ่านไฟล์มาแสดงผลดังโปรแกรมที่ 3

โปรแกรมที่ 2 ตัวอย่างการสร้างไฟล์จอภาพ

```
Program CreateScreen;
Type Screen80       = array [1..25,1..80,1..2] of char;
     ScreenArray    = array [1..25] of String[80];

var  MonoScreen  :  Screen80 Absolute $B000:$0000;
     coloScreen  :  Screen80 Absolute $B800:$0000;
Const  ScreenData : ScreenArray  =


KKK      KKK  MMMMMM      MMMMMM  IIIIIIIIII  TTTTTTTTTT
KKK     KKK   MMM  MM    MM  MMM     III         TTT
KKK    KKK    MMM   MM MM   MMM     III         TTT
KKK  KKK      MMM    MMM     MMM     III         TTT
KKKKKK        MMM    MMM     MMM     III         TTT
KKKKK         MMM            MMM     III         TTT
KKKKKK        MMM            MMM     III         TTT
KKK    KKK    MMM            MMM     III         TTT
KKK     KKK   MMM            MMM     III         TTT
KKK      KKK  MMM            MMM  IIIIIIIIII     TTT

**************************************************
**************************************************

;


Var  ScrFile   :  File of ScreenArray;
begin
   Assign   (ScrFile, 'SCR.TXT');

   Rewrite  (ScrFile);
```

```
    Write    (ScrFile,ScreenData);
  end.
```

โปรแกรมที่ 3 การอ่านไฟล์จอภาพมาแสดงผล

```
Program Display;
Uses Crt;
Type Screen80        = array [1..25,1..80,1..2] of cahr;
     ScreenArray     = array [1..25] of String[80];

var  MonoScreen   : Screen80 Absolute $B000:$0000;
     ColoScreen   : Screen80 Absolute $B800:$0000;
     CharScreen   : ScreenArray ;
     CrtType      : Byte    absolute  $0040:$0049;
     X,Y,Len      : Byte;
     ScrFile      : File of ScreenArray;

procedure ReadScreenFile;
begin
  Assign (ScrFile, 'SCR.TXT');
  Reset  (ScrFile);
  Read   (ScrFile,CharScreen);
end;
procedure DisplayScreen;
begin
  for Y  :=1 to 25 do
    begin
       Len  := length(CharScreen[Y];
       for X:=1 to len do

         if  (CrtType =7) then
          begin

            MonoScreen[Y,X,1] :=CahrScreen[Y][X];
```

```
                MonoScreen[Y,X,2] :=#15  { Attribute }
              end
              else
                begom
                   ColoScreen[Y,X,1]  :=CharScreen[Y][X];
                   ColoScreen[Y,X,2]  :=#15;  { Attribute }
                end;
          end;
      end;
      begin
        ClrScr;
        ReadScreenFile;
        DisplayScreen;
      end.
```

5. การปรับแอตทริบิวต์จอโมโนโครม

การแสดงผลตัวอักษรแต่ละตัวให้มีแอตทริบิวต์ที่ต้องการทำได้โดยใส่ค่าแอตทริบิวต์ลงในวิดีโอแรม และส่วนที่เป็นแอตทริบิวต์ไบต์โดยตรง ในเทอร์โบบาสคาลเองมีคำสั่งเปลี่ยนแอตทริบิวต์ของการแสดงผลได้ 2 แบบคือคำสั่ง Highvideo และ LowVideo เป็นการปรับบิตที่ 3 ของแอตทริบิวต์ทำให้ตัวอักษรมีความเข้มมากและน้อยตามลำดับ เทอร์โบบาสคาล 4.0 ยังได้สร้างตัวแทน TexAttr ใช้เก็บสถานะของแอตทริบิวต์ตัวอักษร การเปลี่ยนแปลงค่าตัวแปร TextAttr จะทำให้เกิดแอตทริบิวต์แบบต่าง ๆ ได้ตามต้องการ

แอตทริบิวต์ที่เป็นไปได้ของจอโมโนโครมได้สรุปไว้ในตารางที่ 1 ค่าแอตทริบิวต์ในช่องขวาสุดของตารางจะใช้เป็นค่าให้กับตัวแปร TextAttr เป็นผลให้คำสั่ง Write หรือ Writeln สร้างตัวอักษรที่มีแอตทริบิวต์แบบต่าง ๆ ได้ดังโปรแกรมที่ 4

ชื่อแอตทริบิวต์	ลักษณะแอตทริบิวต์	ค่าแอตทริบิวต์
Nodisplay	ไม่แสดงตัวอักษร	$00
lowdisplay	ตัวอักษรจาง	$07
Highdisplay	ตัวอักษรเข้ม	$0F
UnderlineLow	ตัวจางขีดเส้นใต้	$01
UnderlineHigh	ตัวเข้มขีดเส้นใต้	$09
ReverseLow	ตัวจางรีเวอร์ส	$70
ReverseHigh	ตัวเข้มรีเวอร์ส	$78
BlinkLow	ตัวจางกะพริบ	$17
BlinkHigh	ตัวเข้มกะพริบ	$1F
UndBlinkLow	ตัวจางขีดเส้นใต้กะพริบ	$81
UndBlinkHigh	ตัวเข้มขีดเส้นใต้กะพริบ	$89
RevBlinkLow	ตัวจางรีเวอร์สกะพริบ	$F0
RevBlinkHigh	ตัวเข้มรีเวอร์สกะพริบ	#F8

ตารางที่ 1 แอตทริบิวต์ที่เป็นไปได้ของจอโมโนโครม

โปรแกรมที่ 4 โปรแกรมปรับแอตทริบิวต์

```
program   DisplayAttr;
uses      Crt;
type      Attr          = Byte;
const     Nodisplay     =  $00;
          LowDisplay    =  $07;
          HighDisplay   =  $0F;
          UnderLinelow  =  $01;

          UnderLineHigh =  $09;

          ReverseLow    =  $70;
```

```
            ReverseJogj     =  $78;
            BlinkLoe        =  $87;
            BlinkHigh       =  $8F;
            UndBlinkLow     =  $81;
            UndBlinkHigh    =  $89;
            RevBlinkLow     =  $F0;
            RevBlinkHigh    =  $F8;
var         Ch    : Char;
{
 PROCEDURE  SetAttr
PURPOSE     :  Set Attribute of display character
INPUT       :  Attribute type
OUTPUT      :  none
Monochrome  Character Attributes byte
     bit no.  7    6    5    4    3    2    1    0
              B    x    x    x    I    x    x    U
}
procedure SetAttr (A: Attr);
begin
  Case    A  of
            Nodisplay       :  TextAttr    :=  $00;
            LowDisplay      :  TextAttr    :=  $07;
            HighDisplay     :  TextAttr    :=  $0F;
            ReverseLow      :  TextAttr    :=  $70;
            ReverseHigh     :  TextAttr    :=  $78;
            UnderLinelow    :  TextAttr    :=  $01;
            UnderLineHihg   :  TextAttr    :=  $09;
            BlinkLow        :  TextAttr    :=  $87;
            BlinkHihg       :  TextAttr    :=  $8F;


            UndBlinkLow     :  TextAttr    :=  $81;
            UndBlinkHigh    :  TextAttr    :=  $89;
```

```
          RevBlinkLow    :  TextAttr    :=  $F0;
          RevBlinkHigh   :  TextAttr    :=  $F8;
     end;
end;
pprocedure Display(X,Y  :Byte;  S  :String);
begin
  GotoXY  ( X,Y);
  Write   (S);
end;
{ Main program }
begin
  ClrScr;
  SetAttr(LowDisplay);      Display(10,04,'LowVideo');
  SetAttr(HighDisplay);     Display(10,06,'HighVideo');
  SetAttr(UnderLineLow);    Display(10,08,'UnderLineLow
                            Video');
  SetAttr(UnderLineHigh);   Display(10,10,'UnderLineHigh
                            video');
 SetAttr(ReverseLow);       Display(10,12,'ReverseLow
                            Video');
 SetAttr(ReverseHigh);      Display(10,14,'ReverseHigh
                            Video');
 SetAttr(BlinkLow);         Display(40,04,'BlinkLow
                            Video');
 SetAttr(BlinkHigh);        Display(40,06,'BlinkHigh
                            Video');
 SetAttr(UndBlinkLow);      Display(40,08,'UnderLineBlink
                            LowVideo');
 SetAttr(UndBlinkHigh);     Display(40,10,'UnderlineBlink

                            HighVideo');
 SetAttr(RevBlinkLow);      Display(40,12,'Reverseblink
                            LowVideo');
```

```
    SetAttr(RevBlinkHigh);     Display(40,14,'ReverseBlink
                               HighVideo');

    SetAttr(LowDisplay);
    Display (25,18,'PRESS ANY KEY TO SYSTEM');
    Ch  :=  ReadKey;
    Writeln;
    Writeln ('Program terminated');
end.
```

6. <u>การแสดงผลความเร็วสูง</u>

เทอร์โบปาสคาลมีคำสั่ง Write และ WriteLn ในการแสดงผลตัวตัวอักษรออกจอภาพทั้งสองคำสั่งเป็นคำสั่งที่เชื่อมต่อกับไบออสด้วยอินเตอร์รัพต์ $10 ความเร็วการแสดงผลตัวอักษรจึงไม่สูงนัก แต่การแสดงผลตัวอักษรด้วยความเร็วสูงอาจทำขึ้นเองได้โดยเขียนโปรแกรมภาษาแอสเซมบลีใช้คำสั่ง Inline เพื่อติดต่อกับวิดีโอแรมโดยตรง หรืออาจใช้คำสั่ง Mem ก็ได้

เทอร์โบ 4.0 มองเห็นความสำคัญในจุดนี้มากจึงออกแบบคำสั่ง Write และ WriteLn ให้สามารถส่งตัวอักษรไปยังวิดีโอแรมโดยตรงได้ เทอร์โบ 4.0 ได้สร้างตัวแปร DirectVideo ซึ่งมีแบบข้อมูลเป็นบูลีนทำหน้าที่เป็นแฟล็กสำหรับเลือกโหมดการทำงาน ในสภาพปกติตัวแปร DirecVideo มีค่าเป็น false ทำให้คำสั่ง write และ WriteLn อยู่ในโหมดการทำงานติดต่อกับไบออส แต่ถ้าตัวแปร DirectVideo มีค่าเป็น true จะทำให้คำสั่ง Write และ WriteLn อยู่ในโหมดการติดต่อกับวิดีโอแรมโดยตรงตัวอักษรที่จะแสดงผลจะถูกนำไปยังวิดีโอแรมทันทีไม่ต้องผ่านไบออส ทำให้ความเร็วเพิ่มสูงขึ้นอย่างมาก

อย่างไรก็ตามปัญหาในการส่งตัวอักษรตรงไปยังวิดีโอแรมจะทำให้เกิดบนจอภาพที่เรียกว่า สโนว์ (Snow) โดยเฉพาะจอสี เทอร์โบจึงได้ออกแบบตัวแปรบูลีน CheckSnow เพื่อใช้ในการตรวจสอบจังหวะการส่งตัวอักษรเพื่อป้องกันสโนว์

ถ้าตัวแปร CheckSnow มีค่าเป็น false หมายถึงไม่มีการตรวจสอบ แต่ถ้า CheckSnow มีค่าเป็น true จะมีการตรวจสอบเพื่อป้องกันสโนว์ และทำให้ความเร็วการแสดงผลลดลงไปเล็กน้อย การตรวจสอบสโนว์นี้ถ้าเป็นจอโมโนโครมก็ไม่มีความจำเป็น แต่ถ้าเป็นจอสีจะจำเป็นอย่างยิ่ง ดังนั้นถ้าหากต้องการให้โปร

แกรมที่เขียนขึ้นใช้งานได้กับจจอโมโนโครม และจอสีอย่างสมบูรณ์ก็ควรให้ค่า true กับ CheckSnow ทุกครั้ง ตัวอย่างการใช้ Write และ WriteLn ในโหมดของวิดีโอแรมนี้แสดงไว้ในโปรแกรมที่ 5

โปรแกรมที่ 5 โปรแกรมแสดงผลความเร็วสูง

```
program  DemoFastDisplay;
uses     Crt;
type     Attr              =  Byte;
         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;
         VideoSeg          :  Wore        =  $B000;

var  Ch  :  char;

     I   :  byte;
{
 PROCEDURE  SetAttr
 PURPOSE    :  Set attribute of display character
 INPUT      :  Attribute type
```

```
  OUTPUT     :  none
  Monochrome Character Attributes Bytes
      bit  no.  7    6    5    4    3    2    1    0
                B    X    X    X    I    X    X    U
}
procedure SetAttr  (A:attr);
begin
  case    A of
          NoSisplay       :    TextAttr  :=  $00;
          LowDisplay      :    TextAttr  :=  $07;
          HighDisplay     :    TextAttr  :=  $0F;
          ReverseLow      :    TextAttr  :=  $70;
          ReverseHigh     :    TextAttr  :=  $78;
          UnderLineLow    :    TextAttr  :=  $01;
          UnderLineHigh   :    TextAttr  :=  $09;
          BlinkLow        :    TextAttr  :=  $87;
          BlinkHigh       :    TextAttr  :=  $8F;
          UndBlinkLow     :    TextAttr  :=  $81;
          UndBlinkHigh    :    TextAttr  :=  $89;
          RevBlinkLow     :    TextAttr  :=  $F0;
          RevBlinkHigh    :    TextAttr  :=  $F8;
     end;
end;

procedure DirectVideoOn;
begin

     DirectVideo     :=  true;
     CheckSnow       :=  true;
end;

{Main program }
begin
```

```
        SetAttr(HighDisplay);
        Clrscr;
        Writeln ('Screen display with Write');
        Writeln ('Press any key when you ready..');
        Ch  := Readkey;
        DirectVideoOff;
        for  I  := 1 to 24 do
        begin
          GotoXY(1,I);
          Write   ('STRING  STRING  STRING  STRING  STRING
                    STRING STRING STRING STRING');
        end;
        Writeln;
        Write ('Oress any key to continue..');
        Ch  := ReadKey;

        Clrscr;
        Writeln;('Screen display with Write DirectVideo');
        Writeln ('Press any key when you ready..');
        DirectVideoOn;
        for  I  := 1 to 24 do
           begin
             GotoXY(1,I);
             Write  ('STRING  STRING  STRING STRING STRING
                      STRING STRING STRING STRING');

          end;
       DirectVideoOff;
end.
```

7. การปรับเคอร์เซอร์

การควบคุมลักษณะเคอร์เซอร์เป็นสิ่งสำคัญประการหนึ่ง ในบางเวลา

การให้เคอร์เซอร์ปรากฏเพื่อบ่งบอกถึงตำแหน่งข้อมูลที่จะป้อน แต่บางครั้งก็ไม่ต้องการให้เคอร์เซอร์ปรากฏ หรือต้องการควบคุมขนาดของเคอร์เซอร์ให้มีขนาดเล็กหรือใหญ่ เพื่อบ่งบอกถึงลักษณะการป้อนข้อมูลบางประการอีกด้วย

การควบคุมเคอร์เซอร์อาจทำได้โดยใช้อินเตอร์รัพต์ $10 ด้วยฟังก์ชันหมายเลข1 แต่ในที่นี้จะควบคุมเคอร์เซอร์โดยการโปรแกรมผ่านตัวควบคุมการแสดงผลจอภาพคือไอซี 6845 หรือที่เรียกว่า CRT-Controller

6845 มีรีจิสเตอร์อยู่ 18 ตัว ชื่อ R0 ถึง R17 รีจิสเตอร์เหล่านี้จะเก็บข้อมูลของการแสดงผลเปลี่ยนไปด้วยรีจิสเตอร์แต่ละตัวแสดงไว้ดังตารางที่ 2

การโปรแกรมรีจิสเตอร์เหล่านี้ทำการติดต่อกับพอร์ต 2 พอร์ตคือ พอร์ต $3B4 เพื่อใช้ในการเลือกหมายเลขรีจิสเตอร์ แล้วจึงติดต่อกับพอร์ต $3B5 เพื่อให้ค่ากับรีจิสเตอร์ที่เลือกไว้ ตัวอย่างเช่น ถ้าต้องการเปลี่ยนขนาดเคอร์เซอร์ให้ใหญ่ขึ้นโดยมีเส้นเริ่มต้นที่เส้นที่ 1 และเส้นสุดท้ายที่ 11 ก็จะต้องเขียนคำสั่งดังนี้

```
Part[$3B4]: = 11;        {Select R11}
Part[$3B5]: = 1;         {R11 = 1}
Part[$3B4]: = 12;        {Select R12}
Part[$3B5]: = 11;        {R12 = 11}
```

หมายเลขพอร์ต $3B4 และ $3B5 นี้ใช้สำหรับจอโมโนโครม ส่วนจอสีจะใช้พอร์ต $3B4 และ $3B5 แทนตามลำดับ อย่างไรก็ตามการเลือกหมายเลขพอร์ตอาจใช้วิธีสร้างตัวแปรแอบโซลุตไว้ที่แอดเดรส $0040:$0003 ซึ่งเป็นแอดเดรสของพื้นที่ข้อมูลไบออสที่เก็บหมายเลขพอร์ตของจอภาพที่ใช้งานอยู่โปรแกรมการกำหนดขนาดเคอร์เซอร์แสดงไว้ในโปรแกรมที่ 6 จะสังเกตเห็นได้ว่าในกรณีที่เป็นจอสีจะมีการลดขนาดความละเอียด ด้วนการแสดงผลตัวอักษรน้อยกว่าจอโมโนโครม

ขนาดของเคอร์เซอร์ที่เป็นไปได้จะมีค่าตั้งแต่เส้น 0 ถึง 13 ดังรูปที่ 2.7

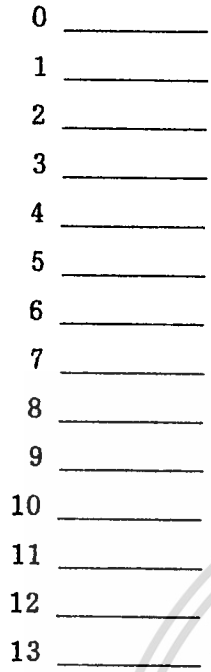
นอกจากนี้ยังมีการพัฒนาโปรแกรมปิดเปิดเคอร์เซอร์ได้ตั้งแต่โปรแกรมที่ 7 ซึ่งมีกระบวนการสำคัญ 2 กระบวนการคือ CursorOn และ CursorOff กระบวนการทั้งสองจะไม่มีผลกระทบกับขนาดของเคอร์เซอร์ที่เป็นอยู่ ตัวอย่าง เช่น ถ้าเคอร์เซอร์มีจุดเริ่มที่เส้น 1 และจุดสุดท้ายที่เส้น 11 เมื่อผู้ใช้ปิดเคอร์เซอร์ด้วย CursorOff และเปิดเคอร์เซอร์ด้วย CursorOn โปรแกรมจะปรับเคอร์เซอร์ให้เป็นตามเดิมได้

หมายเลขรีจิสเตอร์	รีจิสเตอร์เก็บค่า	ค่ากำหนดมาให้จอโมโนโครม	ค่ากำหนดมาให้จอสี
0	จำนวนตัวอักษรทั้งหมดต่อบรรทัด	$61	$71
1	จำนวนตัวอักษรต่อบรรทัด	$50	$50
2	ตำแหน่งสัญญาณ HSYNC	$52	$5A
3	ความกว้างสัญญาณ HSYNC	$0F	$0A
4	จำนวนแถวทั้งหมดบนหน้าจอ	$19	$1F
5	ค่าสแกนไลน์	$06	$06
6	จำนวนแถวต่อหน้าจอ	$19	$19
7	ตำแหน่งสัญญาณ VSYNC	$19	$1C
8	อินเตอร์เลซโหมด	$02	$02
9	จำนวนสแกนไลน์ต่อบรรทัด	$0D	$07
10	เส้นเริ่มเคอร์เซอร์	$0B	$06
11	เส้นสุดท้ายของเคอร์เซอร์	$0C	$07
12	แอดเดรสแถวของจอ (H)	$00	#00
13	แอดเดรสแถวของจอ (L)	$00	$00
14	ตำแหน่งเคอร์เซอร์ (H)	$00	$00
15	ตำแหน่งเคอร์เซอร์ (L)	$00	$00
16	ตำแหน่งไลต์เพน (H)	ไม่ได้กำหนด	อ่านได้เท่านั้น
17	ตำแหน่งไลต์เพน (L)	ไม่ได้กำหนด	อ่านได้เท่านั้น

ตารางที่ 2 รีจิสเตอร์ของ 6845 และค่าปกติของรีจิสเตอร์แต่ละตัว

8. <u>โปรแกรม SCREEN.INC</u>

ในที่นี้ได้รวบรวมกระบวนการต่าง ๆ ที่เกี่ยวข้องกับการทำงานของจอภาพเพื่อสร้างเป็นยูนิตชื่อ SCREEN.PAS ดังโปรแกรมที่ 8



Start = 12 Start = 7 Start = 1
Stop = 13 Stop = 13 Stop = 13

รูปที่ 2.7 เคอร์เซอร์ขนาดต่าง ๆ

โปรแกรมที่ 6 โปรแกรมปรับขนาดเคอร์เซอร์

```
program AdjustCursorShape;
uses      Crt;
const     ColorSeg  =  $B800;

          MonoSeg   =  $B000;
          VideoSeg  :  Word  =  $B000;
var       Ch  :Cahr;

procedure IndentifyCRT;

var CrtType : byte Absolute $0040:0049;
```

```
begin

    if (CrtType = 2) or (CrtType = 3) then
       VideoSeg := ColorSeg;  { Color Graphics Adapter }
    if (CrtType = 7) then
       VideoSeg  :=  MonoSeg; { MonoChrome Graphics
                                Adapter }
end;
{
 PROCEDURE SetCursor
PURPOSE    :    Set cursor's shape
INPUT      :    Top and bottom line
OUTPUT     :    none
}
procedure SetCursor  (Top,Bottom :byte);
var   Vport      :  integer Absolute $0040:$0063;
      CursorMode :  integer Absolute $0040:$0060;
begin
   if VideoSeg = ColorSeg then
   begin
     Top      :=  Top - 4;
     Bottom   :=  Bottom - 4;

  end;
  port[Vport]    := 10;

  port[Vport+1]  := Top;
  port[Vport]    := 11;
  port[Vport+1]  := Bottom;
  CursorMode     := (Top Shl 8 ) or Bottom; { Store
                    Current shape }
end;
{ Main program }
```

```
begin
  Clrscr;

  IdentifyCRT;
  SetCursor(10,11);

  Clrscr;
  Writeln ('We adjust cursor : Start line = 0, End line
           = 13');
  Writeln ('Press any key to continue..');
  SetCursor (0,13);
  GotoXY (1,3(; Write ('A'); GotoXY (1,3);
  Ch := ReadKey;

  Clrscr;
  Writeln ('We adjust cursor : Start line = 6, End line
           = 11');
  Writeln ('Press any key to continue..');
  SetCursor (6,11);
  GotoXY) (1,3); Write('A'); GotoXY (1,3);
  Ch := ReadKey;


  Clrscr;

  Writeln ('We adjust cursor : Start line = 1, End line
          = 4');
  Writeln ('Press any key to continue..');
  SetCursor (1,4);
  GotoXY (1,3); Write('A'); GotoXY (1,3);
Ch  := ReadKey;

Clrscr;
```

```
    SetCursor (11,12);
    Writeln ('We adjust cursor : Start line = 11, End line
             = 12');

    Writeln ('Press any key terminate..');
    SetCursor (11,12);
    GotoXY (1,3); Write('A'); GotoXY (1,3);
    Ch := Readkey;

    Writeln ('Program Terminated..');
end.
```

โปรแกรมที่ 7 โปรแกรมปิด/เปิดเคอร์เซอร์

```
program SwitchCursor;
uses     Crt;
const    ColorSeg  =  $B800;
         MonoSeg   =  $B000;
         VideoSeg  :  Word = $B000;
var      Ch  :  Char;
Procedure IdentifyCrt;
var CrtType : byte Absolute $0040:$0049;


begin
   if (CrtType = 2) or (CrtType = 3) then
       VideoSeg := ColorSeg; { Color Graphics Adapter }
   if  (CrtType = 7 then
       VideoSeg := MonoSeg;  { monoChrome Graphics
                               Adapter }
end;
```

```
{
 PROCEDURE SetCursor
 PURPOSE   :   Set cursor's shape
 INPUT     :   Top and Bottom line
 OUTPUT    :   none

}
procedure SetCursor (Top,Bottom :byte);
var  Vport          : integer Absolute $0040 : $0063;
     CursorMode     : integer Absolute $0040 : $0060;
begin
   if VideoSeg = ColorSeg then
   begin
     Top      := Top - 4;
     Bottom   := Bottom - 4;
   end;
   Port[Vport]   := 10;
   Port[Vport+1] := Top;
   Port[Vport1]  := 11;
   Port[Vport+1] := Bottom;
   CursorMode    := (Top Shl 8 ) or Bottom;  { Store
                    current  shape }
end;


procedure CursorOn;
var  Vport       := integer Absolute $0040 : $0063;
     CursorMode  := integer Absolute $0040 : $0060;
begin
   Port[Vport    := 10;
   Port[Vport+1] := Hi(CursorMode) and $DF;
   Port[Vport]   := 11;
   Port[Vport+1] := Lo(Cursormode)
```

```
end;

procedure Cursoroff;
var  Vport      : integer Absolute $0040 : $0063;
     CursoMode  : integer Absolute $0040 : $0060;
begin

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

begin
   Clrscr;
   identifyCrt;
   Writeln ('Cursor adjust :Start line = 0 , End line =
            13');
   Writeln ('Press any key to continue..');
   SetCursor (0,13);
   Ch := Readkey;

   Clrscr;
   Writeln ('Cursor OFF);

   Writeln ('Press any key to continue..');
   CursorOFF;
   Ch := ReadKey;
   Clrscr;
   Writeln ('Cursor ON again with old shape');
   Writeln ('Press any key to continue..');
   CursorON;
   Ch := Readkey;
```

```
    Clrscr;
    Writeln ('Back to normal cursor;);
    Writeln ('Press any key to terminate..');
    SetCursor (11,12);
    Ch := ReadKey;

 end.
```

8. การสร้างยูนิต SCREEN.PAS

```
 (*-------SCREEN.PAS--------*)

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 SETCURSOR (TOP,BOTTOM :BYTE);
PROCEDURE CURSORON;
PROCEDURE CURSOROFF;

  (*-------END  OF  INTERFACE SECTION--------*)

IMPLEMENTATION

  VAR REGS :REGISTERS;

  PROCEDURE SETATTR (ATTRIB: BYTE);
    BEGIN

      TEXTATTR := ATTRIB;

    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 IDENTIFYCRT;

  BEGIN

    CASE CRTTYPE OF
       0..3   : VIDEOSEG := COLORSEG;
          7   : VIDEOSEG := MONOSEG;
    END;

  END;

BEGIN
```

```
  IDENTIFYCRT;

END. (*----- END  OF UNIT -----*)
```

การสร้างยูนิตเกี่ยวกับ WINDOW

ซอฟต์แวร์แพคเกจสมัยใหม่ หรือโปรแกรมต่าง ๆ ในปัจจุบันนิยมทำจอภาพเป็นช่องหน้าต่างหลาย ๆ ช่องที่เรียกว่า **วินโดว์** แต่ละช่องหน้าต่างอาจใช้แสดงเรื่องราวหลายเรื่องได้ในเวลาเดียวกัน ลักษณะของหน้าต่างอาจมีการซ้อนทับกัน หรือแยกจากกัน แต่ที่สำคัญก็คือ หลังจากการลบหน้าต่างที่ไม่ได้ใช้งานออกไปแล้ว จอภาพเดิมที่ถูกหน้าต่างทับอยู่จะต้องคืนสภาพดังเดิม

แบบของจอหน้าต่าง

ลักษณะหน้าต่างแบ่งออกได้เป็น 2 ประเภทคือ หน้าต่างที่มีการซ้อนกันหลาย ๆ ชั้น และหน้าต่างที่แยกจากกันคือ ไม่มีส่วนใดทับกัน

หน้าต่างแบบซ้อนกัน เหมาะสำหรับงานที่แจกแจงเป็นรายละเอียดย่อย ทำให้ผู้ใช้รู้สึกว่ากำลังใช้งานย่อยลึกลงไปเรื่อย ๆ ตามจำนวนชั้นของการซ้อน ส่วนหน้าต่างที่ไม่ซ้อนกันจะดูเหมือนว่าจอภาพถูกแบ่งออกเป็นจอย่อยหลายจอ หน้าต่างแบบนี้จึงนิยมใช้แสดงเรื่องราวหรือข้อความหลาย ๆ อย่างในเวลาเดียวกัน แต่อย่างไรก็ตามการใช้งานจะต้องอยู่ที่หน้าต่างใดหน้าต่างหนึ่งเท่านั้น

การแสดงผลแบบจอหน้าต่างจะมีส่วนช่วยให้ผู้ใช้ใช้งานได้สะดวก และคล่องตัว เราสามารถนำเทคนิคของหน้าต่างไปใช้สร้างโปรแกรมประเภทป๊อบ - อัพ (Pop-Up)อย่างเช่นไซด์คิกเมื่อกดปุ่ม Ctrl-Alt แล้วมีหน้าต่างปรากฏ หรือสร้างระบบพูลดาวน์เมนู (PullDon Menu) อย่างเช่น การใช้ ASSIST ในดีเบส หรือเอดิเตอร์ของเทอร์โบซี หรือเทอร์โบบาสคาล 4.0

1. วินโดว์ในเทอร์โบบาสคาล

เทอร์โบบาสคาลมีคำสั่งสร้างหน้าต่าง โดยการกำหนดตำแหน่งมุมบน-ซ้าย และมุมล่างขวาของกรอบหน้าต่างด้วยคำสั่ง

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

เมื่อ (x1,y1) เป็นจุดมุมบนซ้าย และ (x2,y2) เป็นจุดมุมล่างขวา

ผลการใช้คำสั่ง Window จะทำให้มีการจำกัดขอบเขตการแสดงผลให้อยู่ภายในกรอบหน้าต่างเท่านั้น ดังนั้นคำสั่งที่เกี่ยวข้องกับการแสดงผลบนจอภาพได้แก่ Read, Readln, Write, Writeln, clrScr, GotoXY, หรือการเลื่อนจอภาพ จะเกิดผลในบริเวณหน้าต่างที่สร้างขึ้นเท่านั้น ในการปรับให้ช่องหน้าต่างกลับคืนสภาวะปกติก็เพียงแต่สร้างกรอบหน้าต่างให้เท่ากับจอปกติด้วยคำสั่ง

Window (1,1,80,25)

เมื่อ (1,1) คือจุดขอบบนซ้ายสุดของจอภาพ และ (80,25) คือจุดขอบล่างขวาสุดของจอภาพ

คำสั่ง Window ของเทอร์โบปาสคาลมีข้อจำกัดใช้งานอยู่มาก ข้อที่นับว่าเป็นจุดด้อยอย่างมากก็คือ ไม่มีการเก็บข้อความเดิมบนจอภาพที่ช่องหน้าต่างจะเกิดขึ้นการเขียนข้อความใหม่ลงไปในช่องหน้าต่างจึงเท่ากับเป็นการทำลายข้อความเดิมบนจอ ข้อจำกัดอื่นก็คือ ไม่สามารถปิดช่องหน้าต่างที่เพิ่งสร้างขึ้น เราจะทำได้เพียงแต่สร้างช่องหน้าต่างใหม่ โดยที่ข้อความของช่องหน้าต่างเดิมยังคงที่ไม่เปลี่ยนแปลง ยิ่งไปกว่านั้นเราไม่สามารถสร้างหน้าต่างซ้อนกันหลาย ๆ ชั้นได้ จะเห็นได้ว่าคำสั่ง Window ของเทอร์โบก็เป็นเพียงคำสั่งพื้นฐานที่ต้องนำไปใช้สร้างระบบหน้าต่างที่แท้จริงด้วยตัวเอง

โครงงานสร้างระบบหน้าต่าง

ก่อนที่จะสร้างระบบหน้าต่าง เราจะต้องกำหนดความสามารถของระบบหน้าต่างเสียก่อน เพื่อชดเชยจุดด้วย และเพิ่มข้อความสามารถที่ไม่มีอยู่ในคำสั่ง Window ของเทอร์โบดังนี้

* ช่องหน้าต่างที่เกิดขึ้นจะมีกรอบปรากฏบนจอภาพ เพื่อบ่งบอกขอบเขตช่องหน้าต่างได้ชัดเจน
* สามารถเลือกแบบของกรอบหน้าต่างได้หลายแบบ
* สามารถเก็บข้อความเดิมและเรียกข้อความเดิมที่เกิดช่องหน้าต่างกลับมาได้ รวมทั้งตำแหน่งเคอร์เซอร์จะไม่เปลี่ยนแปลงด้วย
* หน้าต่างที่สร้างขึ้นสามารถซ้อนกันเป็นชั้น ๆ ได้
* ทุกส่วนของหน้าต่างไม่ว่าจะเป็นส่วนของกรอบ ส่วนหัวของกรอบ พื้นของช่องหน้าต่าง และตัวอักษรที่ปรากฏบนช่องหน้าต่าง สามารถมีแอตทริบิวต์แบบใดก็ได้ตามแบบของแอตทริบิวต์ที่กล่าวมาแล้วในเรื่อง "สร้างยูนิตจอภาพ" โดยขึ้นอยู่กับผู้ใช้กำหนด

โครงสร้างช่องหน้าต่าง

จากข้อกำหนดที่กล่าวข้างต้น การออกแบบโครงสร้างของช่องหน้าต่างให้มีคุณสมบัติครบถ้วนตามกำหนด เราจำเป็นต้องเก็บข้อมูลช่องหน้าต่างทุก ๆ ช่องไว้ เพื่อให้ตัวหน้าต่างสามารถซ้อนกันหลาย ๆ ชั้นได้ สิ่งที่เราจำเป็นต้องเก็บมีดังนี้

- ค่า X1, Y1, X2, Y2 ซึ่งกำหนดกรอบหน้าต่าง
- ค่า X, Y เก็บตำแหน่งเคอร์เซอร์ปัจจุบัน
- ค่า ID บอกหมายเลขของช่องหน้าต่าง
- BackLink เป็นพอยน์เตอร์เชื่อมโยงหน้าต่าง
- ScreenContents เก็บข้อความเดิมทั้งหมดของช่องหน้าต่าง

การสร้างหน้าต่างให้มีคุณสมบัติซ้อนกันได้หลายชั้น ต้องใช้หลักการ - โครงสร้างข้อมูลเข้าช่วย เพื่อให้หน้าต่างเชื่อมโยงกันได้ทั้งนี้เมื่อยกเลิกหน้าต่างปัจจุบันโปรแกรมต้องย้อนไปเปิดหน้าต่างถัดไปได้ โครงสร้างข้อมูลที่เหมาะสม ซึ่งเรานำมาใช้ได้ก็คือลิงก์ลิสต์เนื่องจากเราไม่ทราบว่าจะมีการเปิดใช้ช่องหน้าต่างเป็นจำนวนเท่าใด เราจะใช้คุณสมบัติตัวแปรพอยน์เตอร์ของปาสคาล สำหรับใช้เชื่อมโยงจากหน้าต่างหนึ่งไปยังอักหน้าต่างหนึ่ง ทำให้เกิดช่องหน้าต่างที่ซ้อนกันได้โครงสร้างหน้าต่างที่ออกแบบจะอยู่ในรูปของเรคอร์ดข้างล่างดังนี้

```
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  : byte;
                                X,Y          : byte;
                                Id           : byte;
                                BackLink     : Window
                                               Link;
                              ScreenContents  :
                                          ScreenBlock;
                            end;
```

ปัญหาสำคัญที่เราต้องหาคำตอบก็คือ ช่องหน้าต่างแต่ละช่องมีขนาดแตกต่างกันได้ เราจึงควรใช้วิธีการเก็บข้อมูลในช่องหน้าต่างอย่างประหยัดที่สุด โดยเก็บข้อความเฉพาะหน้าต่างนั้นโดยไม่เก็บทั้งจอภาพ วิธีนี้ทำให้เราใช้หน่วยความจำเท่ากับขนาดช่องหน้าต่างในการเปิดหน้าต่างแต่ละครั้ง

ถ้าเราย้อนกลับไปพิจารณาเรคอร์ด WindowControlBlock จะเห็น

ว่า ScreenContents ซึ่งใช้เป็นที่เก็บข้อความช่องหน้าต่าง มีแบบข้อมูลเป็นอะเรย์ของหน้าจอภาพพอดี การขอใช้หน่วยความจำไดนามิกมาสร้างลิงก์ลิสต์ด้วยคำสั่ง

หน้าต่างปัจจุบัน

หน้าต่าง #4 <-> หน้าต่าง #3 <-> หน้าต่าง #2 <-> หน้าต่าง #1

รูปที่ 2.8 โครงสร้างลิงค์ของช่องหน้าต่างในรูปนี้แสดงตัวอย่างเมื่อเปิดถึงหน้าต่าง 4

สำหรับลักษณะลิงก์ลิสต์ของหน้าต่างจะเป็นดังรูปที่ 2.8

<u>การเก็บช่องหน้าต่าง</u>

เมื่อเราสร้างหน้าต่างแล้ว ต้องมีการเก็บข้อความเดิมไว้ เนื่องจากช่องหน้าต่างแต่ละช่องอาจมีขนาดไม่เท่ากัน จำนวนตัวอักษรที่ต้องเก็บจึงไม่แน่นอนขึ้นอยู่กับขนาดหน้าต่างโดยตรงเราอาจใช้วิธีง่าย ๆ คือเก็บทั้งจอภาพ ซึ่งวิธีนี้จะเขียนโปรแกรมง่าย แต่เปลืองเนื้อที่หน่วยความจำค่อนข้างมาก ดังนั้นในที่นี้เราจึงออกแบบโปรแกรมที่จะเก็บตัวอักษรเฉพาะช่องหน้าต่างหนึ่งเท่านั้น

New (P)

จะได้ขนาดหน่วยความจำเท่ากับขนาดของแบบข้อมูลที่กำหนดไว้ ซึ่งไม่ตรงกับความต้องการของเรา อย่างไรก็ตามเทอร์โบปาสคาลมีคำสั่งการขอใช้หน่วยความจำตามขนาดที่เราต้องการโดยใช้คำสั่ง

FreeMem (P,I)

การเก็บข้อมูลในช่องหน้าต่าง จึงต้องคำนวณขนาดหน้าต่าง ที่ต้องเก็บจากสูตร

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

เมื่อ (X1,Y1) และ (X2,Y2) เป็นจุดมุมซ้ายบน และมุมขวาล่างของกรอบหน้าต่างผลคูณ (X2-X1+1) กับ (Y2-Y1+1) คือ จำนวนตัวอักษรทั้งหมดและเรายังต้องเก็บแอตทริบิวต์ของตัวอักษรแต่ละตัว จึงต้องนำ 2 มาคูณเข้าไปหลังจากที่เราคำนวณขนาดหน้าต่าง และให้เทอร์โบจัดสรรเนื้อที่หน่วยความจำ โดย

ใช้คำสั่ง GetMem แล้วจึงเก็บข้อมูลตัวอักษรกับแอตทริบิวต์ของช่องหน้าต่างไว้ในหน่วยความจำที่ได้มาด้วยคำสั่ง Move ก่อนที่จะนำไปสร้างลิงก์ลิสต์ช่องหน้าต่าง

การตรวจสอบขนาดหน่วยความจำไดนามิก

หน่วยความจำไดนามิกที่ปาสคาลจัดให้ จะมีขนาดจำกัดคือไม่เกิน 1 เซกเมนต์ หรือ 64 กิโลไบต์ ก่อนการใช้จึงจำเป็นต้องตรวจสอบว่าหน่วยความจำที่เหลืออยู่มีขนาดพอเพียงกับที่โปรแกรมต้องการใช้หรือไม่ เทอร์โบปาสคาลมีฟังก์ชัน - MemAvail บอกขนาดหน่วยความจำที่เหลืออยู่

ฟังก์ชัน MemAvail ให้ค่าเป็น LongInt ในการตรวจสอบก่อนการเปิดช่องหน้าต่าง เราจึงใช้ขนาดของช่องหน้าต่างที่คำนวณได้ซึ่งเก็บอยู่ในตัวแปร - WindowSize มาเปรียบเทียบกับ MemAvail

การเรียกคืนข้อความเมื่อยกเลิกช่องหน้าต่าง

ในกรณีของการปิด หรือเลือกหน้าต่างจำเป็นต้องเรียกค่าที่เก็บไว้กลับคืนมา ขั้นตอนก็จะกลับกันคือคำนวณขนาดหน้าต่างแล้วนำข้อมูลที่เก็บไว้กลับมาแสดงผลบนจอภาพ โดยใช้คำสั่ง Move ย้ายข้อมูลไปสู่วิดีโอแรมแล้วจึงคืนค่าหน่วยความจำกลับคืนให้เทอร์โบโดยใช้คำสั่ง FreeMem

การสร้างกรอบ

กรอบหน้าต่างมีประโยชน์สำหรับบ่งบอกขอบเขตหน้าต่างแบบของกรอบหน้าต่างในที่นี้สร้างไว้รวม 4 แบบ ดังรูปที่ 2.9 ผู้ใช้สามารถออกแบบเพิ่มเติมได้อีก

2. การสร้างยูนิต WIN.PAS

โปรแกรม WIN.PAS ตัวโปรแกรมที่ 1 เป็นโปรแกรมยูนิตที่รวมกระบวนการสำหรับสร้างหน้าต่างได้แก่ กระบวนการ WindowOpen, WindowClose และ InitWin รวมทั้งกระบวนการในการตั้งค่าพารามิเตอร์ของหน้าต่างได้แก่ SetWinHeader, SetWinAttr, SetBoxAttr, SetHeadAttr, SetChar-Attr และ SetBoxStyle ซึ่งมีการใช้งานดังนี้

WindowOpen เป็นกระบวนการเปิดช่องหน้าต่างโดยกำหนดจุดโคออร์ดิเนต (X1,Y1) และ (X2,Y2) กระบวนการนี้จะสร้างกรอบหน้าต่าง และแอตทริบิวต์ต่าง ๆ ของช่องหน้าต่างให้ ตลอดจนตรวจสอบความถูกต้องของโครงสร้างหน้า-

ต่าง รวมทั้งขนาดหน่วยความจำไดนามิกที่เหลืออยู่ว่าพอเพียงต่อการเก็บข้อความช่องหน้าต่างหรือไม่ โดยให้รหัสความผิดพลาดอยู่ในตัวแปร ErrorWindow

ก. แบบ Single ข. แบบ Double

ค. แบบ Mix1 ง. แบบ Mix2

รูปที่ 2.9 แบบของกรอบหน้าต่าง 4 แบบที่สร้างไว้ให้ก่อน

WindowClose เป็นการยกเลิกการใช้ช่องหน้าต่างแล้วย้อนกลับไปยังช่องหน้าต่างก่อนหน้านั้น

InitWin สำหรับกำหนดค่าเริ่มต้นที่ใช้ในการสร้างหน้าต่าง กระบวนการ InitWin ต้องเรียกใช้ก่อนเสมอ (เพียงครั้งเดียวก่อนที่จะมีการใช้ระบบหน้าต่าง)

SetWinHeader เป็นกระบวนการกำหนดเฮดเดอร์ของวินโดว์

SetWinAttr เป็นกระบวนการปรับอแตทริบิวต์ของกรอบหน้าต่าง

SetBoxAttr ปรับแอตทริบิวต์ของเฮดเดอร์

SetCharattr เป็นการปรับแอตทริบิวต์ของอักษรในจอหน้าต่าง

SetBoxStyle เป็นการเลือกแบบกรอบหน้าต่าง

ส่วนแบบข้อมูล ค่าคงที่ ตัวแปรของยูนิต WIN,PAS ได้แสดงไว้ในตาราง ที่ 1 ถึง 3 ตามลำดับ สำหรับโปรแกรมที่ 2 เป็นโปรแกรมทดสอบระบบหน้าต่าง โปรแกรมนี้จะสร้างหน้าต่างบนจอภาพ 5 หน้าต่าง โดยใช้กรอบหน้าต่างแบบต่าง ๆ กัน

ตารางที่ 1 แบบข้อมูลใน WIN.PAS

ชื่อแบบข้อมูล	การใช้งาน
ScreenLine	อะเรย์ของจำนวนเต็มขนาด 80 หน่วย เป็นแบบข้อมูลของจอภาพ 1 บรรทัด ซึ่งประกอบด้วยตัวอักษร และแอตทริบิวต์
ScreenArray	อะเรย์ขนาด 25 หน่วยของ ScreenLine ใช้เป็นแบบข้อมูลของจอภาพหนึ่งหน้าจอ
ScreenBlock	อะเรย์ของตัวเลขจำนวนเต็ม 2,000 ตัวใช้เป็นแบบข้อมูลของจอภาพเช่นเดียวกับ ScreenArray แต่มองจอภาพเป็นบล็อกขนาดใหญ่ต่อเนื่องกันไป
WindowLink	พอยน์เตอร์ของ WindowControlBlock
WindowControl-Block	เรคอร์ดของหน้าต่างประกอบด้วย X1,Y1,X2,Y2 เป็นขอบมุมบนซ้าย และมุมล่างขวาของหน้าต่าง X,Y เป็นตำแหน่งเคอร์เซอร์ปัจจุบัน ID เป็นหมายเลขหน้าต่าง Backlink เป็นพอยน์เตอร์ชี้ไปยังหน้าต่างก่อนหน้านั้น ScreenContent เป็นที่เก็บข้อมูลของจอภาพ
HeadString	สตริงขนาด 80 ตัวอักษรสำหรับเป็นเฮดเดอร์ของจอหน้าต่าง

ตารางที่ 2 ค่าคงตัวใน WIN.PAS

ชื่อค่าคงตัว	การใช้งาน
Single	เส้นกรอบหน้าต่างแบบเส้นเดี่ยว
Double	เส้นกรอบหน้าต่างแบบเส้นคู่
Mix1	เส้นกรอบหน้าต่างแบบเส้นเดี่ยวผสมเส้นคู่แบบที่ 1
Mix2	เส้นกรอบหน้าต่างแบบเส้นเดี่ยวผสมเส้นคู่แบบที่ 2
AttrofWindow	แอตทริบิวต์ของพื้นหน้าต่างค่าปกติเป็น HighDisplay
AttrOfBox	แอตทริบิวต์ของเส้นกรอบหน้าต่างค่าปกติเป็น HighDisplay
AttrOfHeader	แอตทริบิวต์ของเฮดเดอร์ ค่าปกติเป็น HighDisplay
AttrOfChar	แอตทริบิวต์ของตัวอักษรในจอหน้าต่างค่าปกติเป็น HighDisplay
HeaderOfWindow	สตริงของเฮดเดอร์ ค่าปกติเป็นสตริงความยาวศูนย์
TypeOfBox	อะเรย์ของเส้นกรอบหน้าต่าง

ตารางที่ 3 ตัวแปรใน WIN.PAS

ชื่อตัวแปร	การใช้งาน
ActiveWindow	เป็นพอยน์เตอร์ชี้หน้าต่างที่ใช้งานปัจจุบัน
ScreenPtr	พอยน์เตอร์ชี้วิดีโอแรม
FixedSize	ตัวแปรจำนวนเต็มเก็บขนาดข้อมูลที่ไม่เปลี่ยนแปลงในวินโดว์ ซึ่งได้แก่ X1,X2,Y1,y2.X,Y,ID และ BackLink
ErrorWindow	เป็นตัวแปรบ่งบอกรหัสความผิดพลาดการปิดช่องหน้าต่าง ถ้ามีค่า 0 ไม่มีความผิดพลาด ถ้ามีค่า 1 กำหนดกรอบไม่ถูกต้อง ถ้ามีค่า 2 หน่วยความจำไม่พอต่อการเก็บช่องหน้าต่าง

```
{************************************************
 *              PROGRAM....WIN.PAS              *
 *           ----------------------             *
 ************************************************}

UNIT WIN ;

INTERFACE
USES SCREEN,CRT ;
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 ; {window boundaries}
     X,Y              : INTEGER ; {cursor loaction}
     ID               : BYTE    ; {id number}
     BACKLINK         : WINDOWLINK ;
     SCREENCONTENTS : SCREENBLOCK ;
                        END;
     HEADSTRING = STRING [80];

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 : HEADSTRING = '';
      TYPEOFBOX      : ARRAY [1..4,1..8] OF CHAR =

                ( (' ','ı','ı',' ', ',','ı','ı', '),

                (' ','=','=',' ',' ','=','=',' '),

                   );

VAR   ACTIVEWINDOW   : WINDOWLINK ;
      SCREENPTR      : ^SCREENARRAY ;
      FIXEDSIZE      : INTEGER ;
      WINDOWCOUNT    : BYTE ;
      ERRORWINDOW    : BYTE ;


PROCEDURE WINDOWBOX  (X1,Y1,X2,Y2 :BYTE);
PROCEDURE WINDOWOPEN (X1,Y1,X2,Y2 :BYTE);
PROCEDURE WINDOWCLOSE;
PROCEDURE INITWIN;
PROCEDURE SETBOXSTYLE  (NAME :BYTE);
PROCEDURE SETWINHEADER (ST   :HEADSTRING);
PROCEDURE SETWINATTR   (NAME :BYTE);
PROCEDURE SETBOXATTR   (NAME :BYTE);
PROCEDURE SETHEADATTR  (NAME :BYTE);
PROCEDURE SETCHARATTR  (NAME :BYTE);

{*--------END OF INTERFACE SECTION---------*}

IMPLEMENTATION

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 ;

  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
     BEGIN
       WRITE(TYPEOFBOX [BOXSTYLE,TOP] );
     END;
    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] );

(*--MARK IT THE CURRENT WINDOW AND LOCATE CURSOR TO TOP--*)
      SETATTR (ATTROFHEADER) ;
      GOTOXY ( (X1+X2-LENGTH(HEADEROFWINDOW)) DIV 2, Y1) ;
      WRITE (HEADEROFWINDOW);
      WINDOW(X1+1,Y1+1,X2-1,Y2-1) ;
      SETATTR (ATTROFCHAR);
  END;  {END OF PROCEDURE BOXWIN}

 PROCEDURE WINDOWOPEN (X1,Y1,X2,Y2 :BYTE) ;
 VAR  BLOCK       : WINDOWLINK ;
      LINELENGTH,
      WINDOWSIZE,
      I           : INTEGER ;
      Y           : BYTE ;
 BEGIN
   LINELENGTH := X2-X1+1 ;
   WINDOWSIZE := LINELENGTH*(Y2-Y1+1)*2+FIXEDSIZE;
       { CHECK WINDOW VALID }
   IF (X2>80) OR (Y2>25) OR (X2-X1<2) OR (Y2-Y1<2) 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 DO
          BEGIN
           MOVE(SCREENPTR^[Y,X1],BLOCK^.SCREENCONTENTS[I]
           ,LINELENGTH*2);
           I := I+LINELENGTH ;
          END;
        WINDOWBOX (X1,Y1,X2,Y2) ;
        END ;
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+1;
       WINDOWSIZE    := LINELENGTH*(BLOCK^.Y2-BLOCK^
                        .Y1+1)*2+FIXEDSIZE ;
       WINDOWCOUNT   := WINDOWCOUNT-1 ;
```

```
        I             := 1 ;
        FOR Y := BLOCK^.Y1 TO BLOCK^.Y2 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;
VAR MODE : BYTE ABSOLUTE $0040:$0049;

  BEGIN
    ACTIVEWINDOW := NIL ;
    FIXEDSIZE    := SIZEOF(WINDOWCONTROLBLOCK)
                    -SIZEOF(SCREENBLOCK);
    IF (MODE=2) OR (MODE=3) THEN
       SCREENPTR := PTR($B800,0)
    ELSE
       SCREENPTR := PTR($B000,0);
    WINDOW (1,1,80,25);
    WINDOWCOUNT := 0;
  END;
```

```
PROCEDURE SETBOXSTYLE (NAME :BYTE) ;
   BEGIN
     BOXSTYLE := NAME;
   END;

PROCEDURE SETWINHEADER (ST :HEADSTRING);
   BEGIN
     HEADEROFWINDOW := ST;
   END;

PROCEDURE SETWINATTR (NAME :BYTE) ;
   BEGIN
     ATTROFWINDOW := NAME;
   END;

PROCEDURE SETBOXATTR (NAME :BYTE) ;
   BEGIN
     ATTROFBOX := NAME ;
   END;

 PROCEDURE SETHEADATTR (NAME :BYTE) ;
    BEGIN
      ATTROFHEADER := NAME ;
    END;

 PROCEDURE SETCHARATTR (NAME :BYTE) ;
    BEGIN
      ATTROFCHAR := NAME;
    END;

    END.{OF UNIT}
```

การสร้างพูลดาวน์เมนู

ความประทับใจของผู้ใช้ที่มีต่อซอฟต์แวร์ หรือแพคเกจใด เริ่มตั้งแต่ข้อความแรกที่มองเห็นบนจอภาพ และความสะดวกในการใช้งาน ซอฟต์แวร์ใหม่บนไมโครคอมพิวเตอร์ปัจจุบันจึงมีการปรับปรุงทั้งในด้านเทคนิคการแสดงผลบนจอภาพการใช้สีสัน ความเร็วการทำงาน และใช้งานง่าย สิ่งเหล่านี้เป็นหัวใจสำคัญสูงสุดที่ผู้ออแบบซอฟต์แวร์ไมโครคอมพิวเตอร์คำนึงถึงมาก ซอฟต์แวร์ที่ผลิตออกมา ต้องใช้งานได้ง่ายผู้ใช้ไม่จำเป็นต้องใช้เวลาในการศึกษามากนัก

โปรแกรม หรือแพคเกจใหม่ ที่ออกมาในระยะหลังจึงเพิ่มความสะดวกต่อการใช้งานในรูปของระบบใหม่ เช่น ป๊อบ-อัพ (Pop-Up) หรือ พูล-ดาวน์ (Pull-Down) โปรแกรมระบบป๊อบ-อัพ มักเป็นโปรแกรมเรสซิเดนต์ อย่างเช่น ไซด์คิก ส่วนพูลดาวน์จะมีใช้อย่างมากในแพคเกจทั่วไป เช่น ดีเบส หรือ เวิร์ดโปรเซสเวอร์รุ่นใหม่ ๆ

ระบบพูล-ดาวน์เมนู ในที่นี้ เป็นการสร้างโปรแกรมต้นแบบให้นำไปดัดแปลงเข้ากับความต้องการได้สะดวก และนำไปใช้เป็นเครื่องมือพัฒนาโปรแกรมพูล-ดาวน์ ได้ง่ายอีกด้วย

1) <u>พูล-ดาวน์เมนู</u>

หลายท่านคงจะคุ้นเคยอยู่กับโปรแกรมที่มีระบบการทำงานเป็นพูล-ดาวน์นี้มาบ้างแล้ว ที่เห็นชัดที่สุดก็คือ เอดเตอร์ของเทอร์โบปาสคาล 4.0 เอง

เอดิเตอร์ของเทอร์โบปาสคาล 4.0 เป็นระบบเมนูให้เลือกคำสั่งโดยใช้ปุ่มลูกศร 4 ทิศทาง ในจอภาพหนึ่งจอจะมีได้หลายเมนู แต่จะมีเพียงเมนูเพียงที่แอคตีฟ หรือกำลังเลือกใช้อยู่ หากจะเรียกใช้เมนูอื่น ๆ ก็ต้องกดปุ่มลูกศรซ้าย หรือขวาเมนูเก่าจะหายไป บนจอภาพจะปรากฏเมนูใหม่ขึ้นมาแทน ส่วนข้อเลือกในแต่ละเมนูก็ต้องใช้ลูกศรขึ้นหรือลง ข้อไหนถูกเลือกอยู่จะเห็นเป็นรีเวิร์ส หรือขีดเส้นใต้ หรือแบบอื่น ๆ สุดแต่ผู้ออกแบบโปรแกรมจะกำหนด

แนวความคิดที่ใช้กับระบบพูล-ดาวน์เมนู จะเป็นการเก็บสถานะของเมนู และข้อเลือกไว้ในตัวแปร เมื่อมีการเลือกเมนู หรือเปลี่ยนข้อย่อยโดยใช้ปุ่มลูกศรก็จะมีการปรับสถานะนี้ตามไปด้วยเสมอ ในขณะเดียวกันก็มีการปรับสถานะบนจอภาพตามไปด้วย ทำให้โปรแกรมตรวจสอบได้ตลอดเวลาว่า ขณะนี้ผู้ใช้กำลังเลือกเมนูใด และเลือกข้อย่อยใดอยู่

<u>ออกแบบโครงสร้างข้อมูลเมนู</u>

การออกแบบโครงสร้างของโปรแกรมนี้ มีเป้าหมายให้โปรแกรมยืดหยุ่นตัวต่อการดัดแปลง ส่วนสำคัญจุดนี้อยู่ที่ตัวเมนู ซึ่งเป็นส่วนที่ต้องแก้ไขดัดแปลงทั้งข้อความ และตำแหน่งแสดงผลอยู่เสมอ

การสร้างเมนูให้อยู่ในส่วนของโปรแกรมจะทำให้แก้ไขยุ่งยาก เมนูควรจะอยู่ในรูปของข้อมูลมากกว่า แล้วมีโปรแกรมเป็นตัวจัดการอีกต่อหนึ่ง เมื่อจะดัดแปลง หรือแก้ไขเมนูก็เพียงแต่แก้ที่ข้อมูลแทนการแก้โปรแกรม

โครงสร้างเมนูจึงถูกออกแบบให้อยู่ในรูปของข้อมูลเป็นเรคอร์ดต่อไปนี้

```
const MaxChoice = 15;
type StructureMenu = record
     Win   :array (1..4) of byte;
     Col   :array (1..MaxChoice) of byte;
     Row   :array (0..MaxChoice) of byte;
     Msg   :array (0..MaxChoice) of byte;
     LastChoice  :byte;
end;
```

ในหนึ่งเรคอร์ดของเมนูประกอบไปด้วย

Win เป็นอะเรย์ของตัวกำหนดกรอบของจอหน้าต่าง

Col เป็นอะเรย์ของตำแหน่งข้อความทางแกน X

Row เป็นอะเรย์ของตำแหน่งข้อความทางแกน Y

Msg เป็นอะเรย์ของข้อความหรือตัวข้อเลือกในเมนู

LastChoice เป็นตัวกำหนดว่าเมนูนี้มีข้อเลือกข้อสุดท้ายหมายเลขใด ในหนึ่งเมนูจะมีข้อเลือกได้ไม่เกินค่า MaxChoice ซึ่งเป็นค่าคงตัวที่ต้องกำหนดไว้ก่อน ซึ่งในที่นี้กำหนดให้มีค่าเป็น 15

เมื่อเราได้โครงสร้างของเมนูแล้ว เราจำเป็นต้องมองต่อไปว่าเมนูในระบบมีได้หลายเมนู วิธีจัดการกับเมนูได้สะดวกก็คือมองเมนูทั้งหมดในรูปอะเรย์ โดยให้ชื่อว่า StrucMenu และมีโครงสร้างดังนี้

```
const MaxMenu      =   4;
type StrucMenu     =   array (1..MaxMenu) of StructureMenu
```

แบบข้อมูล StrucMenu ข้างต้นเป็นอะเรย์ของ StructureMenu โดยมีค่า MaxMenu เป็นตัวกำหนดจำนวนเมนูที่มีได้ ในที่นี้ให้ค่าไว้เท่ากับ 4 ดังนั้นโครง

สร้างของเมนูที่เรากำหนดขึ้นจึงเป็นการสร้างระบบพูล-ดาวน์ที่มีเมนูสูงสุดได้ 4 เมนู มีข้อเลือกได้ไม่เกิน 15 ข้อ

<u>การกำหนดข้อความในเมนู</u>

เมื่อสร้างแบบข้อมูลของเมนูได้แล้วขั้นต่อไปจะเป็นการกำหนดตัวข้อมูลในเมนู ซึ่งได้แก่กำหนดตำแหน่งของกรอบหน้าต่าง กำหนดข้อความในแต่ละเมนู พร้อมทั้งตำแหน่งการแสดงผล และตัวบ่งบอกถึงจำนวนข้อเลือกที่มีอยู่ในเมนูนั้น การกำหนดตัวข้อมูลลงเมนูในรูปที่ 2.10 ต่อไปนี้ เป็นการสร้างค่าคงที่ตัว Menu ให้มีแบบข้อมูลเป็น StrucMenu โดยให้

เมนูที่ 1 มีข้อเลือก 5 ข้อ
เมนูที่ 2 มีข้อเลือก 4 ข้อ
เมนูที่ 3 และ 4 มีข้อเลือกเมนูละ 3 ข้อ

โครงสร้างเมนูที่ออกแบบจึงมีทรีของเมนูดังรูปที่ 2.11

```
Const menu : StrucMenu = (
                (Win:(01,02,22,08);
                Col   :(01,02,02,02,02,02,00,00,00,
                        00,00,00,00,00,00,00);
                Row   :(01,01,02,03,04,05,00,00,00,
                        00,00,00,00,00,00,00);
                Msg   :(' Menu1 '
                        ' OPTIONS  NUMBER 1'
                        ' OPTIONS  NUMBER 2'
                        ' OPTIONS  NUMBER 3'
                        ' OPTIONS  NUMBER 4'
                        ' OPTIONS  NUMBER 5'
               LastChoice:5)
               (Win  :(19,02,40,07);
               Col   :(19,02,40,07);
               Row   :(01,01,02,03,04,00,00,00,00,
                       00,00,00,00,00,00,00);
               Msg   :('  Menu2  '
                       '  OPTIONS  NUMBER 1
```

```
                    ' . OPTIONS  NUMBER 2
                    '  OPTIONS  NUMBER 3
                    '  OPTIONS  NUMBER 4

                LastChoice:4)
                 (Win  :(39,02,60,06);
                 Col   :(39,02,02,02,00,00,00,00,00,
                         00,00,00,00,00,00,00);
                 Row   :(01,01,02,03,00,00,00,00,00,
                         00,00,00,00,00,00,00);
```

รูปที่ 2.10 การกำหนดตัวข้อมูลของเมนู

```
                LastChoice:3)

                 (Win  :(59,02,80,06);
                 Col   :(59,02,02,02,00,00,00,00,00,
                         00,00,00,00,00,00,00);
                 Row   :(01,01,02,03,00,00,00,00,00,
                         00,00,00,00,00,00,00);
                Msg    :('  Menu4  '
                          OPTIONS  NUMBER1
                          OPTIONS  NUMBER2
                          OPTIONS  NUMBER3

                LastChoice:3)
                );
```

Menu

Menu1	Menu2	Menu3	Menu4
Choice #1	Choice #1	Choice #1	Choice #1
Choice #2	Choice #2	Choice #2	Choice #2
Choice #3	Choice #3	Choice #3	Choice #3
Choice #4	Choice #4		
Choice #5			

รูปที่ 2.11 ทรีของเมนูที่ออกแบบ

<u>การอ้างอิงถึงข้อมูลในเมนู</u>

การอ้างอิงถึงข้อมูลเมนูใดจะทำได้ง่ายเนื่องจากเราออกแบบเมนูเป็นอะเรย์อีกด้วย การอ้างอิงจึงใช้อินเด็กซ์บ่งบอก ตัวอย่างเช่น ถ้าต้องการอ้างถึงข้อย่อยที่ 3 ของเมนูที่ 1 ก็สามารถอ้างอิงตำแหน่ง และข้อความดังนี้

```
Menu (1).Col(3)
Menu (1).Row(3)
Menu (1).Msg(3)
```

หรือถ้าจะพิมพ์เมนูที่ 1 ออกมาทางจอภาพก็เพียงแต่ใช้อุปกรณ์การพิมพ์ดังนี้

```
With Menu(1) do
     for I:= 1 to LastChoice do
     begin
          GotoXY (Col(I),Row
          (I));
          Write (Msg(I);
     end;
```

<u>การตรวจสอบฟังก์ชันคีย์</u>

การใช้พูล-ดาวน์เมนูจะใช้การควบคุมเมนูผ่านทางฟังก์ชันคีย์ เทคนิคการตรวจสอบการกดฟังก์ชันได้กล่าวไว้แล้ว ฟังก์ชันคีย์ที่ใช้ในระบบพูล-ดาวน์นี้กำหนดไว้ 18 ปุ่มดังตารางที่ 1

2. <u>วินาโดว์กับการควบคุมระบบพูล-ดาวน์</u>

ลักษณะของการใช้พูล-ดาวน์เมนูที่เราพบเห็นก็คือ เมื่อกดปุ่มควบคุมจะมีการปรับเมนูไปตามการกด ถ้าเรากดปุ่มลูกศรขึ้น หรือลงจะไม่ย้ายเมนู แต่เป็นการเลื่อนข้อเลือกขึ้นหรือลงไปตามการกด ซึ่งเราสังเกตได้จากแถบข้อความทึบที่ปรากฏขึ้นเป็นเสมือนแถบเคอร์เซอร์ เทคนิคที่ใช้ก็เป็นเพียงการสร้างแอตทริบิวต์ของข้อเลือกให้เด่นชัดขึ้นมา ดังนั้นเมื่อเรากดปุ่มลูกศรขึ้นหรือลงหนึ่งครั้งก็จะเกิดการปรับแอตทริบิวต์ของข้อเลือก 2 ข้อ คือ ปรับแอตทริบิวต์เดิมให้กลับสู่สภาพปกติ และปรับแอตทริบิวต์ของข้อเลือกใหม่ที่เลื่อนไปหาให้เด่นขึ้น

การเลื่อนแถบเคอร์เซอร์ยังมีลักษณะวนรอบได้ กล่าวคือ ถ้าเลื่อนลงไปจนสุดข้อเลือกสุดท้าย แถบเคอร์เซอร์จะวนไปปรากฏที่ข้อเลือกแรก หรือถ้าเลื่อนแถบเคอร์เซอร์ขึ้นไปวนเลยข้อเลือกแรก ก็มีผลให้แถบเคอร์เซอร์กลับมาปรากฏที่ข้อเลือกสุดท้าย

สำหรับในกรณีที่มีการกดปุ่มลูกศรซ้ายหรือขวา จะมีความหมายถึงการย้ายจากเมนูหนึ่งไปยังอีกเมนูหนึ่ง สิ่งที่เกิดขึ้นบนจอภาพคือเมนูเดิมหายไป และมีเมนูใหม่ปรากฏขึ้น เทคนิคที่ใช้จึงเป็นระบบวินาโดว์ คือขณะที่เราแสดงเมนูใดอยู่เมื่อมีการย้ายเมนูก็ต้องปิดช่องหน้าต่างนั้นแล้วเปิดช่องหน้าต่างสำหรับเมนูใหม่ สำหรับการเลื่อนเมนูจะมีคุณสมบัติวนรอบได้เช่นเดียวกับการเลื่อนข้อเลือก

ปุ่ม	การทำงาน
ลูกศร ขึ้น	เลื่อนแถบเคอร์เซอร์ขึ้น
" ลง	เลื่อนแถบเคอร์เซอร์ลง
" ขวา	เลือกเมนูทางขวา
" ซ้าย	เลือกเมนูทางซ้าย
PgUp	เลื่อนแถบเคอร์เซอร์ไปที่ข้อเลือกแรกของเมนูนั้น
PgDn	เลื่อนเคอร์เซอร์ไปที่ข้อเลือกสุดท้ายของเมนูนั้น
Home	เลือกเมนูแรก
End	เลือกเมนูสุดท้าย
F1-F10	ปุ่มฟังก์ชันคีย์ตามแถบฟังก์ชันคีย์

ตารางที่ 1 ปุ่มฟังก์ชันคีย์ที่ใช้ในระบบพูลดาวน์

3. โปรแกรมพูล-ดาวน์ และการดัดแปลง

โปรแกรมบทที่ 3 เป็นตัวโปรแกรมพูล-ดาวน์เมนูซึ่งเขียนอยู่ในรูปของโปรแกรมที่สมบูรณ์ โปรแกรมนี้จะยืดหยุ่นตัวต่อการแก้ไขดัดแปลง เพิ่มเติมหรือตัดทอน การปรับปรุงโปรแกรมให้มีเมนู หรือข้อเลือกเป็นแบบอื่นจะทำได้สะดวก โปรแกรมนี้จึงเป็นเสมือนเครื่องมือ (Tools)

สำหรับการดัดแปลงโปรแกรมทำได้โดยการเปลี่ยนค่าต่าง ๆ ในส่วนของข้อมูลเมนูซึ่งได้แก่ค่าคงที่ MaxChoice, MaxMenu และตัวเมนูซึ่งประกอบด้วยตำแหน่งกรอบหน้าต่าง ตัวข้อความ พร้อมกับตำแหน่ง X, Y นอกจากนี้ก็ต้องปรับข้อกำหนดฟังก์ชันคีย์ (ค่าคงตัว FKey) รวมทั้งกระบวนความ DoMenu ให้เหมาะสม

```
{ ************************************************************
  *                                                          *
  *         THIS PROGRAM IS..DTS_MAIN (PULLDOWN MENU)        *
  *                                                          *
  ************************************************************}
    PROGRAM PULLDOWNMENU ;
     { PURPOSED       :PROTOTYPE OF THE PULL-DOWN MENU SYSTEM }
     { WRITEN DATE    :AUGUST,27 1990                         }
     {LAST  UPDATE    :AUGUST,10 1990                         }
     USES CRT,SCREEN,KEYBOARD,WIN, GRAPH,DATA_PIN,PICTURE,INTRO
     ;
     CONST MAXCHOICE = 20;
           MAXMENU   =  4;
     TYPE STRING6    = STRING [6];
          STRING80   = STRING [80];
          STRUCTUREMENU   = RECORD
                              WIN    :ARRAY [1..4]          OF BYTE;
                              COL    :ARRAY [0..MAXCHOICE] OF BYTE;
                              ROW    :ARRAY [0..MAXCHOICE] OF BYTE;
                              MSG    :ARRAY [0..MAXCHOICE] OF STRING80;
                              LASTCHOICE : BYTE;
                            END;
          STRUCMENU = ARRAY [1..MAXMENU] OF STRUCTUREMENU;
          FUNCTIONMENU= RECORD
                              COL    : ARRAY [1..10] OF BYTE;
                              ROW    : ARRAY [1..10] OF BYTE;
                              MSG    : ARRAY [1..10] OF STRING6;
                            END;
          CONST MENU :STRUCMENU =(
                 (WIN:(02,03,21,18);
                 COL :(01,02,02,02,02,02,02,02,02,02,
                       02,02,02,02,02,02,02,02,02,02);
                 ROW :(01,01,02,03,04,05,06,07,08,09,
                       10,11,12,13,14,15,16,17,18,19,20);
                 MSG :('  INTRODUCTION MENU ',
                       ' OR_GATE     ',
                       ' AND_GATE    ',
                       ' NAND_GATE   ',
                       ' NOR_GATE    ',
                       ' NOT_GATE    ',
                       ' EX-OR_GATE  ',
                       ' EX-NOR_GATE',
                       ' D_FIFLOP    ',
                       ' SR_FIFLOP   ',
                       ' JK_FIFLOP   ',
                        '',
                       '','','','','','','','','','');
                       LASTCHOICE:10),
```

```
                (WIN:(20,03,54,24);
                COL :(21,02,02,02,02,02,02,02,02,02,
                      02,02,02,02,02,02,02,02,02,02,02);
                ROW :(01,01,02,03,04,05,06,07,08,09,
                      10,11,12,13,14,15,16,17,18,19,20);
MSG :('    DATA PIN IC MENU    ',
           ' #7400 ( 2 INPUT NAND_GATE...) ',
           ' #7402 ( 2 INPUT NOR_GATE....) ',
           ' #7404 ( HEX INVERTER_GATE...) ',
           ' #7408 ( 2 INPUT AND_GATE....) ',
           ' #7432 ( 2 INPUT OR_GATE.....) ',
           ' #7474 ( D_FLIP FLOP.........) ',
           ' #7486 ( EX_OR_GATE..........) ',
           ' #74152( 8 TO 1 LINE.........) ',
           ' #74164( 8 BIT SHIFTREG......) ',
           ' #74393( 4 BIT BINARY COUNT..) ',
           ' #7485 ( 4 BIT COMPARATOR....) ',
           ' #74112( JK_FLIP FLOP........) ',
           ' #74131( 3TO8 DECODER........) ',
           ' #74147( 10 TO 4 PRIORITY....) ',
           ' #74165( 8 BIT SHIFT_PISO....) ',
           ' #74169( UP DOWN DECADE COUNT) ',
           ' #74195( 4 BIT SHIFT PIPO....) ',
           ' #74283( 4 BIT FULL_ADDER....) ',
           ' #74390( DECADE COUNTER......) ',
           ' #74447( BCD TO 7 SEGMENT....) ' );
                      LASTCHOICE:20) ,
                (WIN:(40,03,65,24);
                COL :(43,02,02,02,02,02,02,02,02,02,
                      02,02,02,02,02,02,02,02,02,02,02);
                ROW :(01,01,02,03,04,05,06,07,08,09,
                      10,11,12,13,14,15,16,17,18,19,20);
                MSG :('    TEST IC MENU    ',
                      '  # 7400  ( 14_PIN ) ',
                      '  # 7402  ( 14_PIN ) ',
                      '  # 7404  ( 14_PIN ) ',
                      '  # 7408  ( 14_PIN ) ',
                      '  # 7432  ( 14_PIN ) ',
                      '  # 7474  ( 14_PIN ) ',
                      '  # 7486  ( 14_PIN ) ',
                      '  # 74152 ( 14_PIN ) ',
                      '  # 74164 ( 14_PIN ) ',
                      '  # 74393 ( 14_PIN ) ',
                      '  # 7485  ( 16_PIN ) ',
                      '  # 74112 ( 16_PIN ) ',
                      '  # 74131 ( 16_PIN ) ',
                      '  # 74147 ( 16_PIN ) ',
                      '  # 74165 ( 16_PIN ) ',
                      '  # 74169 ( 16_PIN ) ',
```

```
                    '  # 74283 ( 16_PIN ) ',
                    '  # 74390 ( 16_PIN ) ',
                    '  # 74447 ( 16_PIN ) '
                   );
                     LASTCHOICE:20),
               (WIN:(50,10,79,16);
                COL:(60,02,02,00,00,00,00,00,00,00,
                     00,00,00,00,00,00,00,00,00,00,00);
                ROW:(01,02,04,00,00,00,00,00,00,00,
                     00,00,00,00,00,00,00,00,00,00,00);
                MSG:(' NUMBER SLOT# MENU ',
                     ' SELECT SLOT# NUMBER.. 1  ',
                     ' SELECT SLOT# NUMBER.. 2  ',
                     '','','','','','','','','','',
                     '','','','','','','');
                     LASTCHOICE:2)
                                 ) ;
(*--FUNCTION KEY MEASSAGE--*)
TKEY   :FUNCTIONMENU  =(COL :(01,09,17,25,33,41,49,57,65,72);
                        ROW :(25,25,25,25,25,25,25,25,25,25);
                        MSG :(' HELP ',
                              'DISP_1',
                              'DISP_2',
                              'DISP_3',
                              'FUNC5 ',
                              'FUNC6 ',
                              'FUNC7 ',
                              'FUNC8 ',
                              'FUNC9 ',
                              'QUIT ')
                              );
CONST CHOICEMENU   : ARRAY[1..MAXMENU] OF 1..MAXCHOICE =(1,1,1,1);
      STATUSMENU   : 1..MAXMENU = 1;
      FINISH       : BOOLEAN    = FALSE;   { RUN FLAG }
      SET_STYLEWIN : BYTE       = LOWDISPLAY;
      SET_STYLEBOX : BYTE       = REVERSELOW;
VAR   KEY          : CHAR;                 { KEEP KEYBORD CODE}
PROCEDURE DISPLAYTITLE;
BEGIN
   SETWINATTR(REVERSELOW);
   SETBOXATTR(REVERSELOW);
   SETCHARATTR(REVERSELOW);
   SETBOXSTYLE(SINGLE);
   WINDOWOPEN(10,07,70,16);
   GOTOXY(19,3);
   WRITE( ' THE DIGITAL TRAINNER'  );
   GOTOXY(19,4);
   WRITE('                    ');
```

```
   GOTOXY(22,6);
   WRITE('WRITE BY.....           ');
   GOTOXY(19,8);
   WRITE('1. K. AMNOUY          ');
   GOTOXY(19,10);
   WRITE('2. C.SOMBOON          ');
   REPEAT UNTIL KEYPRESSED;
   DELAY(1500);END;
PROCEDURE DISPLAYFUNCKEY;
VAR  I :BYTE ;
BEGIN
   WINDOW(1,1,80,25);3
   WITH FKEY DO BEGIN
      FOR I := 1 TO 10 DO BEGIN
         {WRITE NUMBER  }
         GOTOXY(COL[I],ROW[I]);
         SETATTR(LOWDISPLAY);
         WRITE(I);
         { WRITE MESSAGE }
         SETATTR(REVERSELOW);
           WRITE(MSG[I]);
         END;
      END;
  END;
PROCEDURE DISPLAYMAINMENU;
VAR I :BYTE ;
BEGIN
   CURSOROFF;
   SETWINATTR(SET_STYLEWIN);
   SETBOXATTR(SET_STYLEBOX);
   WINDOWOPEN(1,1,80,24);
   SETBOXSTYLE(SINGLE);
   SETATTR(LOWDISPLAY);
   FOR I:= 1 TO MAXMENU DO
        WITH MENU[I] DO
        BEGIN
           GOTOXY(COL[0],ROW[0]);
           WRITE(MSG[0]);
        END;
END;
PROCEDURE CHOICEACTIVE(OLD,NEW:BYTE);
BEGIN
   WITH MENU[STATUSMENU] DO
   BEGIN
      { DISPLAY OLD CHOICE WITH NORMAL ATTRIBUTE }
      SETATTR(LOWDISPLAY);
      GOTOXY(COL[OLD],ROW[OLD]);
      WRITE(MSG[OLD]);
      {   DISPLAY NEW CHOICE WITH REVERSE ATTRBUTE   }
```

```
        SETATTR(REVERSELOW);
        GOTOXY(COL[NEW],ROW[NEW]);
        WRITE(MSG[NEW]);
      END;
  END;
  PROCEDURE MENUACTIVE(NEW:BYTE);
  VAR I:BYTE;
  BEGIN
     { DISPLAL MENU HEADER }
     DISPLAYMAINMENU;
     SETATTR(REVERSELOW);
     WITH MENU[NEW] DO
     BEGIN
        GOTOXY(COL[0],ROW[0]);
        WRITE(MSG[0]);
        { CREATE WINDOW  }
        SETWINATTR(LOWDISPLAY);
        SETBOXATTR(LOWDISPLAY);
        SETCHARATTR(LOWDISPLAY);
        SETBOXSTYLE(SINGLE);
        SETWINHEADER('');
        WINDOWOPEN(WIN[1],WIN[2],WIN[3],WIN[4]);
        { DISPLAY CHOICES OF MENU }
        FOR I:= 1 TO LASTCHOICE DO
        BEGIN
          GOTOXY(COL[I],ROW[I]);
          WRITE(MSG[I]);
          END;
        END;
        CHOICEACTIVE(1,CHOICEMENU[NEW]);END;
  PROCEDURE MOVEUP;
  VAR CURRENTCHOICE  :BYTE;
  BEGIN
     CURRENTCHOICE :=CHOICEMENU[STATUSMENU];
     IF CURRENTCHOICE = 1 THEN
        CHOICEMENU[STATUSMENU]  := MENU[STATUSMENU].LASTCHOICE
     ELSE
        CHOICEMENU[STATUSMENU] := CURRENTCHOICE-1;
     CHOICEACTIVE(CURRENTCHOICE,CHOICEMENU[STATUSMENU]);
  END;
  PROCEDURE MOVEDOWN;
  VAR CURRENTCHOICE :BYTE;
  BEGIN
     CURRENTCHOICE := CHOICEMENU[STATUSMENU];
     IF CURRENTCHOICE = MENU[STATUSMENU].LASTCHOICE THEN
        CHOICEMENU[STATUSMENU] := 1
     ELSE
        CHOICEMENU[STATUSMENU] := CURRENTCHOICE+1;
     CHOICEACTIVE(CURRENTCHOICE,CHOICEMENU[STATUSMENU]);
```

```
        END;
        PROCEDURE MOVEFORWARD;
        BEGIN
           WINDOWCLOSE;
           IF STATUSMENU+1 >MAXMENU THEN
              STATUSMENU := 1
           ELSE
           STATUSMENU:=STATUSMENU+1;
           MENUACTIVE(STATUSMENU);
         END;
        PROCEDURE MOVEBACK;
        BEGIN
   WINDOWCLOSE;
           IF  STATUSMENU = 1 THEN
               STATUSMENU := MAXMENU
           ELSE
              STATUSMENU := STATUSMENU-1;
           MENUACTIVE(STATUSMENU);
        END;
        PROCEDURE MOVETOFRISTCHOICE;
BEGIN
           CHOICEACTIVE(CHOICEMENU[STATUSMENU],1);
           CHOICEMENU[STATUSMENU] := 1;
END;
        PROCEDURE MOVETOLASTCHOICE;
        BEGIN
           WITH MENU[STATUSMENU] DO BEGIN
              CHOICEACTIVE (CHOICEMENU[STATUSMENU],LASTCHOICE);
              CHOICEMENU[STATUSMENU]:=LASTCHOICE;
           END;
        END;
        PROCEDURE MOVETOFIRSTMENU;
        BEGIN
   WINDOWCLOSE;
           STATUSMENU := 1;
           MENUACTIVE(1);
        END;
        PROCEDURE MOVETOLASTMENU;
        BEGIN
           WINDOWCLOSE;
           STATUSMENU := MAXMENU;
   MENUACTIVE(MAXMENU);
        END;

                    {*****************************}
                    {*  START PROCEDURE MENU_1   *}
                    {*****************************}
   PROCEDURE NO_1;
```

```
        VAR I,J,XC,YC :INTEGER;
        BEGIN
          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
          INTRO_OR(XC,YC);
          REPEAT UNTIL KEYPRESSED;
          CLOSEGRAPH ;
          MENUACTIVE (1);
END;
        PROCEDURE NO_2;
        VAR I,J,XC,YC :INTEGER;
        BEGIN

          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
          INTRO_AND(XC,YC);
          REPEAT UNTIL KEYPRESSED;
          CLOSEGRAPH ;
          MENUACTIVE (1);
        END;

        PROCEDURE NO_3;
        VAR I,J,XC,YC :INTEGER;
        BEGIN

          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
          INTRO_NAND(XC,YC);
          REPEAT UNTIL KEYPRESSED;
          CLOSEGRAPH ;
          MENUACTIVE (1);
        END;

        PROCEDURE NO_4;
        VAR I,J,XC,YC :INTEGER;
        BEGIN

          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
          INTRO_NOR(XC,YC);
```

```
        REPEAT UNTIL KEYPRESSED;
        OSEGRAPH ;
        MENUACTIVE (1);
END;
        PROCEDURE NO_5;
        VAR I,J,XC,YC :INTEGER;
        BEGIN

        I := DETECT;
        INITGRAPH (I,J,'');
        XC:= GETMAXX DIV 2;
        YC:= GETMAXY DIV 2;
        INTRO_NOT(XC,YC);
        REPEAT UNTIL KEYPRESSED;
        CLOSEGRAPH ;
        MENUACTIVE (1);
END;
        PROCEDURE NO_6;
        VAR I,J,XC,YC :INTEGER;
        BEGIN

        I := DETECT;
        INITGRAPH (I,J,'');
        XC:= GETMAXX DIV 2;
        YC:= GETMAXY DIV 2;
        INTRO_EXNOR(XC,YC);
        REPEAT UNTIL KEYPRESSED;
        CLOSEGRAPH ;
        MENUACTIVE (1);
        END;

        PROCEDURE NO_7;
        VAR I,J,XC,YC :INTEGER;
        BEGIN

        I := DETECT;
        INITGRAPH (I,J,'');
        XC:= GETMAXX DIV 2;
        YC:= GETMAXY DIV 2;
        INTRO_EXNOR(XC,YC);
        REPEAT UNTIL KEYPRESSED;
        CLOSEGRAPH ;
        MENUACTIVE (1);
END;
        PROCEDURE NO_8;
        VAR I,J,XC,YC :INTEGER;
        BEGIN
```

```
  I := DETECT;
     INITGRAPH (I,J,'');
     XC:= GETMAXX DIV 2;
     YC:= GETMAXY DIV 2;
     INTRO_D(XC,YC);
     REPEAT UNTIL KEYPRESSED;
     CLOSEGRAPH ;
     MENUACTIVE (1);
   END;

   PROCEDURE NO_9;
   VAR I,J,XC,YC :INTEGER;
   BEGIN
     I := DETECT;
     INITGRAPH (I,J,'');
     XC:= GETMAXX DIV 2;
     YC:= GETMAXY DIV 2;
     INTRO_RS (XC,YC);
     REPEAT UNTIL KEYPRESSED;
     CLOSEGRAPH ;
     MENUACTIVE (1);
END;
   PROCEDURE NO_10 ;
   VAR I,J,XC,YC :INTEGER;
   BEGIN

     I := DETECT;
     INITGRAPH (I,J,'');
     XC:= GETMAXX DIV 2;
     YC:= GETMAXY DIV 2;
     INTRO_OK (XC,YC);
     REPEAT UNTIL KEYPRESSED;
     CLOSEGRAPH ;
     MENUACTIVE (1);
   END;
   PROCEDURE CONTROL (A,B:INTEGER; VAR C:INTEGER);EXTERNAL;
   {$L BPS_LINK}

   PROCEDURE OUT1_SOUND;
   BEGIN
       sound(770);
       delay(200);
       sound(720);
       delay(200);
       sound(670);
       delay(250);
       sound(620);
       delay(200);
       sound(570);
```

```
        delay(200);
        sound(520);
        delay(200);
        sound(470);
        delay(200);
        sound(420);
        delay(250);
        sound(370);
        delay(200);
        sound(320);
        delay(200);
        sound(480);
        delay(500);
        nosound;END;
    PROCEDURE PROCESSMENU( NUMBER :INTEGER);
    VAR PORT,DATA,D:INTEGER;
    BEGIN
       SETWINATTR(REVERSELO;
       SETBOXATTR(REVERSELOW);
       SETCHARATTR(REVERSELOW);
       SETBOXSTYLE(SINGLE);
WINDOWOPEN(20,09,50,14);
       GOTOXY(03,2);
       WRITE(' SLOT#  NUMBER..... ',CHOICEMENU[STATUSMENU]);
       GOTOXY(03,3);
       WRITE(' IC     NUMBER.....7',NUMBER);
       IF CHOICEMENU[STATUSMENU] = 1  THEN
         BEGIN
           CASE NUMBER OF
            400  : BEGIN  DATA:= 01; PORT:= $F8 ; END;
            402  : BEGIN  DATA:= 02; PORT:= $F8 ; END;
            404  : BEGIN  DATA:= 03; PORT:= $F8 ; END;
            408  : BEGIN  DATA:= 04; PORT:= $F8 ; END;
            432  : BEGIN  DATA:= 05; PORT:= $F8 ; END;
            474  : BEGIN  DATA:= 06; PORT:= $F8 ; END;
            486  : BEGIN  DATA:= 07; PORT:= $F8 ; END;
            4132 : BEGIN  DATA:= 08; PORT:= $F8 ; END;
            4164 : BEGIN  DATA:= 09; PORT:= $F8 ; END;
            4393 : BEGIN  DATA:= 10; PORT:= $F8 ; END;
            485  : BEGIN  DATA:= 01; PORT:= $F9 ; END;
            4112 : BEGIN  DATA:= 02; PORT:= $F9 ; END;
            4131 : BEGIN  DATA:= 03; PORT:= $F9 ; END;
            4147 : BEGIN  DATA:= 04; PORT:= $F9 ; END;
            4165 : BEGIN  DATA:= 05; PORT:= $F9 ; END;
            4169 : BEGIN  DATA:= 06; PORT:= $F9 ; END;
            4195 : BEGIN  DATA:= 07; PORT:= $F9 ; END;
            4283 : BEGIN  DATA:= 08; PORT:= $F9 ; END;
            4390 : BEGIN  DATA:= 09; PORT:= $F9 ; END;
            4447 : BEGIN  DATA:= 10; PORT:= $F9 ; END;END;END;
```

```
      IF CHOICEMENU[STATUSMENU] = 2  THEN
       BEGIN
         CASE NUMBER OF
          400  : BEGIN  DATA:= 01; PORT:= $FA ; END;
          402  : BEGIN  DATA:= 02; PORT:= $FA ; END;
          404  : BEGIN  DATA:= 03; PORT:= $FA ; END;
          408  : BEGIN  DATA:= 04; PORT:= $FA ; END;
          432  : BEGIN  DATA:= 05; PORT:= $FA ; END;
          474  : BEGIN  DATA:= 06; PORT:= $FA ; END;
          486  : BEGIN  DATA:= 07; PORT:= $FA ; END;
          4152 : BEGIN  DATA:= 08; PORT:= $FA ; END;
          4164 : BEGIN  DATA:= 09; PORT:= $FA ; END;
          4393 : BEGIN  DATA:= 10; PORT:= $FA ; END;
          485  : BEGIN  DATA:= 01; PORT:= $FB ; END;
          4112 : BEGIN  DATA:= 02; PORT:= $FB ; END;
          4131 : BEGIN  DATA:= 03; PORT:= $FB ; END;
          4147 : BEGIN  DATA:= 04; PORT:= $FB ; END;
          4165 : BEGIN  DATA:= 05; PORT:= $FB ; END;
          4169 : BEGIN  DATA:= 06; PORT:= $FB ; END;
          4195 : BEGIN  DATA:= 07; PORT:= $FB ; END;
          4283 : BEGIN  DATA:= 08; PORT:= $FB ; END;
          4390 : BEGIN  DATA:= 09; PORT:= $FB ; END;
          4447 : BEGIN  DATA:= 10; PORT:= $FB ; END;END;END;END;
      OUT1_SOUND;
      CONTROL(DATA PORT,D);
      DELAY(800);
      WINDOWCLOSE;
      MOVEBACK;END;
  PROCEDURE NUM_IC(VAR NUMBER:INTEGER );
   BEGIN
 CASE NUMBER OF
    1 : NUMBER :=400 ;
    2 : NUMBER :=402 ;
    3 : NUMBER :=404 ;
    4 : NUMBER :=408 ;
    5 : NUMBER :=432 ;
    6 : NUMBER :=474 ;
    7 : NUMBER :=486 ;
    8 : NUMBER :=4152;
    9 : NUMBER :=4164;
    10: NUMBER :=4393;
    11: NUMBER :=485 ;
    12: NUMBER :=4112;
    13: NUMBER :=4131;
    14: NUMBER :=4147;
    15: NUMBER :=4165;
    16: NUMBER :=4169;
    17: NUMBER :=4195;
    18: NUMBER :=4283;
```

```
   19: NUMBER :=4390;
   20: NUMBER :=4447;
    END;
  END;

PROCEDURE DOMENU1;
VAR NUMBER:INTEGER;
BEGIN
NUMBER:= CHOICEMENU[STATUSMENU];
 CASE NUMBER OF
  1 : NO_1 ;
  2 : NO_2 ;
  3 : NO_3 ;
  4 : NO_4 ;
  5 : NO_5 ;
  6 : NO_6 ;
  7 : NO_7 ;
  8 : NO_8 ;
  9 : NO_9 ;
  10: NO_10;
END;
END;
          {*****************************}
          {*  START PROCEDURE DATA_PIN *}
          {*****************************}
PROCEDURE IC_7400;
VAR I,J,XC,YC :INTEGER;
BEGIN
  I := DETECT;
  INITGRAPH (I,J,'');
  XC:= GETMAXX DIV 2;
  YC:= GETMAXY DIV 2;
  PIN_7400 (XC,YC);
  REPEAT UNTIL KEYPRESSED;
  CLOSEGRAPH ;
  MENUACTIVE (2);
END;
PROCEDURE IC_7402;
VAR I,J,XC,YC :INTEGER;
BEGIN
  I := DETECT;
  INITGRAPH (I,J,'');
  XC:= GETMAXX DIV 2;
  YC:= GETMAXY DIV 2;
  PIN_7402(XC,YC);
  REPEAT UNTIL KEYPRESSED;
  CLOSEGRAPH ;
  MENUACTIVE (2);
END;
```

```
        PROCEDURE IC_7404;
        VAR I,J,XC,YC :INTEGER;
        BEGIN
          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
          PIN_7404(XC,YC);
          REPEAT UNTIL KEYPRESSED;
          CLOSEGRAPH ;
          MENUACTIVE (2);
END;
        PROCEDURE IC_7408;
        VAR I,J,XC,YC :INTEGER;
        BEGIN
          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
          PIN_7408(XC,YC);
          REPEAT UNTIL KEYPRESSED;
          CLOSEGRAPH ;
          MENUACTIVE (2);
        END;
        PROCEDURE IC_7432;
        VAR I,J,XC,YC :INTEGER;
        BEGIN
          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
          PIN_7432(XC,YC);
          REPEAT UNTIL KEYPRESSED;
          CLOSEGRAPH ;
          MENUACTIVE (2);
        END;
        PROCEDURE IC_7474;
        VAR I,J,XC,YC :INTEGER;
        BEGIN
          I := DETECT;
          INITGRAPH (I,J,'');
          XC:= GETMAXX DIV 2;
          YC:= GETMAXY DIV 2;
        PIN_7474(XC,YC);
          REPEAT UNTIL KEYPRESSED;
          CLOSEGRAPH ;
          MENUACTIVE (2);
        END;
        PROCEDURE IC_7486;
```

```
VAR I,J,XC,YC :INTEGER;
    BEGIN
      I := DETECT;
      INITGRAPH (I,J,'');
      XC:= GETMAXX DIV 2;
      YC:= GETMAXY DIV 2;
      PIN_7486(XC,YC);
      REPEAT UNTIL KEYPRESSED;
      CLOSEGRAPH ;
      MENUACTIVE (2);
    END;
    PROCEDURE IC_74152;
    VAR I,J,XC,YC :INTEGER;
    BEGIN
      I := DETECT;
      INITGRAPH (I,J,'');
      XC:= GETMAXX DIV 2;
      YC:= GETMAXY DIV 2;
      PIN_74152 (XC,YC);
      REPEAT UNTIL KEYPRESSED;
      CLOSEGRAPH ;
      MENUACTIVE (2);
END;
    PROCEDURE IC_74164;
    VAR I,J,XC,YC :INTEGER;
    BEGIN
      I := DETECT;
      INITGRAPH (I,J,'');
      XC:= GETMAXX DIV 2;
      YC:= GETMAXY DIV 2;
      PIN_74164(XC,YC);
      REPEAT UNTIL KEYPRESSED;
      CLOSEGRAPH ;
      MENUACTIVE (2);
    END;
    PROCEDURE IC_74393;
    VAR I,J,XC,YC :INTEGER;
    BEGIN
      I := DETECT;
      INITGRAPH (I,J,'');
      XC:= GETMAXX DIV 2;
      YC:= GETMAXY DIV 2;
      PIN_74393(XC,YC);
      REPEAT UNTIL KEYPRESSED;
      CLOSEGRAPH ;
      MENUACTIVE (2);
    END;
    PROCEDURE IC_7485;
    VAR I,J,XC,YC :INTEGER;
```

```
BEGIN
  I := DETECT;
  INITGRAPH (I,J,'');
  XC:= GETMAXX DIV 2;
  YC:= GETMAXY DIV 2;
  PIN_7485(XC,YC);
  REPEAT UNTIL KEYPRESSED;
CLOSEGRAPH ;
  MENUACTIVE (2);
END;
PROCEDURE IC_74112;
VAR I,J,XC,YC :INTEGER;
BEGIN
  I := DETECT;
  INITGRAPH (I,J,'');
  XC:= GETMAXX DIV 2;
  YC:= GETMAXY DIV 2;
  PIN_74112(XC,YC);
  REPEAT UNTIL KEYPRESSED;
  CLOSEGRAPH ;
  MENUACTIVE (2);
END;
PROCEDURE IC_74131;
VAR I,J,XC,YC :INTEGER;
BEGIN
  I := DETECT;
  INITGRAPH (I,J,'');
  XC:= GETMAXX DIV 2;
  YC:= GETMAXY DIV 2;
  PIN_74131(XC,YC);
  REPEAT UNTIL KEYPRESSED;
  CLOSEGRAPH ;
  MENUACTIVE (2);
END;
PROCEDURE IC_74147;
VAR I,J,XC,YC :INTEGER;
BEGIN
  I := DETECT;
  INITGRAPH (I,J,'');
  XC:= GETMAXX DIV 2;
  YC:= GETMAXY DIV 2;
  PIN_74147(XC,YC);
  REPEAT UNTIL KEYPRESSED;
  CLOSEGRAPH ;
  MENUACTIVE (2);
END;
PROCEDURE IC_74165;
VAR I,J,XC,YC :INTEGER;
BEGIN
```

```
    I := DETECT;
    INITGRAPH (I,J,'');
    XC:= GETMAXX DIV 2;
    YC:= GETMAXY DIV 2;
    PIN_74165(XC,YC);
    REPEAT UNTIL KEYPRESSED;
  CLOSEGRAPH ;
    MENUACTIVE (2);
END;
  PROCEDURE IC_74169;
  VAR I,J,XC,YC :INTEGER;
  BEGIN
    I := DETECT;
    INITGRAPH (I,J,'');
    XC:= GETMAXX DIV 2;
    YC:= GETMAXY DIV 2;
    PIN_74169(XC,YC);
    REPEAT UNTIL KEYPRESSED;
    CLOSEGRAPH ;
    MENUACTIVE (2);
  END;
  PROCEDURE IC_74195;
  VAR I,J,XC,YC :INTEGER;
  BEGIN
    I := DETECT;
    INITGRAPH (I,J,'');
    XC:= GETMAXX DIV 2;
    YC:= GETMAXY DIV 2;
    PIN_74195(XC,YC);
    REPEAT UNTIL KEYPRESSED;
    CLOSEGRAPH ;
    MENUACTIVE (2);
  END;
  PROCEDURE IC_74283;
  VAR I,J,XC,YC :INTEGER;
  BEGIN
    I := DETECT;
    INITGRAPH (I,J,'');
    XC:= GETMAXX DIV 2;
    YC:= GETMAXY DIV 2;
    PIN_74283(XC,YC);
    REPEAT UNTIL KEYPRESSED;
    CLOSEGRAPH ;
    MENUACTIVE (2);
END;
  PROCEDURE IC_74390;
  VAR I,J,XC,YC :INTEGER;
  BEGIN
    I := DETECT;
```

```
    INITGRAPH (I,J,'');
    XC:= GETMAXX DIV 2;
    YC:= GETMAXY DIV 2;
    PIN_74390(XC,YC);
    REPEAT UNTIL KEYPRESSED;
    CLOSEGRAPH ;
  MENUACTIVE (2);
  END;
  PROCEDURE IC_74447;
  VAR I,J,XC,YC :INTEGER;
  BEGIN
    I := DETECT;
    INITGRAPH (I,J,'');
    XC:= GETMAXX DIV 2;
    YC:= GETMAXY DIV 2;
    PIN_74447(XC,YC);
    REPEAT UNTIL KEYPRESSED;
    CLOSEGRAPH ;
    MENUACTIVE (2);
  END;
            {*****************************}
            {*        DOMENU_2           *}
            {*****************************}
  PROCEDURE DOMENU2;
  VAR NUMBER: INTEGER;
BEGIN
    NUMBER:= CHOICEMENU[STATUSMENU];
    NUM_IC(NUMBER);
    CASE NUMBER OF
    400 : IC_7400;
    402 : IC_7402;
    404 : IC_7404;
    408 : IC_7408;
    432 : IC_7432;
    474 : IC_7474;
    486 : IC_7486;
    4152: IC_74152;
    4164: IC_74164;
    4393: IC_74393;
    485 : IC_7485;
    4112: IC_74112;
    4131: IC_74131;
    4147: IC_74147;
    4165: IC_74165;
    4169: IC_74169;
    4195: IC_74195;
    4283: IC_74283;
    4390: IC_74390;
    4447: IC_74447;END;
```

```
        END;
        PROCEDURE DOMENU3;
         VAR NUMBER: INTEGER;
             KEY : CHAR;
        BEGIN
         NUMBER := CHOICEMENU[STATUSMENU];
          MOVEFORWARD ;
          READFUNCKEY (KEY);
          IF KEY = RETURN_KEY THEN
            BEGIN
              NUM_IC(NUMBER) ;
              PROCESSMENU(NUMBER);
            END;
         END;
        PROCEDURE DOMENU4;
         BEGIN
         MOVEBACK;
         END;
        PROCEDURE DOMENU;
        BEGIN
           CASE STATUSMENU OF
      1 : DOMENU1;
      2 : DOMENU2;
            3 : DOMENU3;
            4 : DOMENU4;
         END;
        END;
        PROCEDURE PROCESSFUNCTION(NUM :BYTE);
VAR CH :CHAR ;
        BEGIN
           SETWINATTR(REVERSELOW);
           SETBOXATTR(REVERSELOW);
   SETCHARATTR(REVERSELOW);
   SETBOXSTYLE(SINGLE);
           WINDOWOPEN(25,12,55,14);
           GOTOXY(08,2);
           WRITE('     PROCESS FUNCTION  ',NUM);
           CH := READKEY;
           WINDOWCLOSE;
        END;

        PROCEDURE DOF1;
        BEGIN
         PROCESSFUNCTION(1);
        END;

PROCEDURE  DOF2;
BEGIN
```

```
   SET_STYLEWIN := LOWDISPLAY;
         SET_STYLEBOX := LOWDISPLAY;
         DISPLAYMAINMENU;
         MOVETOFIRSTMENU;
       END;
       PROCEDURE  DOF3;
       BEGIN
         SET_STYLEWIN := LOWDISPLAY;
      SET_STYLEBOX := REVERSELOW;
         DISPLAYMAINMENU;
         MOVETOFIRSTMENU;
       END;
       PROCEDURE  DOF4;
       BEGIN
         SET_STYLEWIN := REVERSELOW;
         SET_STYLEBOX := REVERSELOW;
         DISPLAYMAINMENU;
         MOVETOFIRSTMENU;
       END;
       PROCEDURE  DOF5;
BEGIN
         PROCESSFUNCTION(5);
       END;
       PROCEDURE  DOF6;
       BEGIN
  PROCESSFUNCTION(6);
       END;

       PROCEDURE  DOF7;
       BEGIN
         PROCESSFUNCTION(7);
END;
       PROCEDURE  DOF8;
       BEGIN
         PROCESSFUNCTION(8);
       END;

       PROCEDURE  DOF9;
       BEGIN
         PROCESSFUNCTION(9);
       END;

       PROCEDURE QUIT;
       BEGIN
         SETATTR(HIGHDISPLAY);
         WINDOWCLOSE;
         CURSORON;
         CLRSCR;
  FINISH := TRUE;
```

```
END;
PROCEDURE TESTKEY(KEY:CHAR);
BEGIN
   IF FUNCKEY THEN
              CASE (KEY) OF
                 HOME_KEY : MOVETOFIRSTMENU;
                 END_KEY  : MOVETOLASTMENU;
                 UP_KEY   : MOVEUP;
                 LT_KEY   : MOVEBACK;
 RT_KEY    : MOVEFORWARD;
 DN_KEY    : MOVEDOWN;
                 PGUP_KEY : MOVETOFRISTCHOICE;
                PGDN_KEY : MOVETOLASTCHOICE;
                 F1_KEY   : DOF1;
                 F2_KEY   : DOF2;
                 F3_KEY   : DOF3;
                 F4_KEY   : DOF4;
                 F5_KEY   : DOF5;
                 F6_KEY   : DOF6;
                 F7_KEY   : DOF7;
                 F8_KEY   : DOF8;
 F9_KEY    : DOF9;
 F10_KEY   : QUIT;
              END  { OF CASE }
              ELSE
                 IF KEY = RETURN_KEY THEN DOMENU;
              END;
         PROCEDURE INTRO_1;
          VAR I,J,XC,YC :INTEGER;
          BEGIN
           I:=DETECT;
           INITGRAPH(I,J,'');
           XC:=GETMAXX DIV 2-20;YC:=GETMAXY DIV 2-25;
           SETTEXTSTYLE(1,0,5);
           OUTTEXTXY(XC-170,YC-60,'WELCOME TO ......');
           SETTEXTSTYLE(1,0,6);
           OUTTEXTXY(XC-190,YC-17,'THE');
           SETTEXTSTYLE(1,0,8);
           OUTTEXTXY(XC-155,YC-15,'  DIGITAL');
           OUTTEXTXY(XC-185,YC+50,' TRAINER.');
           SETTEXTSTYLE(1,0,1);
           OUTTEXTXY(XC+100,YC+135,'K.M.I.T.L (1990)');
           REPEAT UNTIL KEYPRESSED;
           CLOSEGRAPH ;
          END ;
          PROCEDURE INTRO_2;
           VAR I,J,XC,YC:INTEGER;
           BEGIN
           I:=DETECT;
```

```
  INITGRAPH (I,J,'');
  XC:= GETMAXX DIV 2-20;YC:= GETMAXY DIV 2-25;
  SETTEXTSTYLE(1,0,8);
  OUTTEXTXY(XC-185,YC,'THANK YOU');
  DELAY(2000);
  CLOSEGRAPH;
  CLRSCR;END;
 PROCEDURE OUT2_SOUND;
BEGIN
   sound(770);
   delay(500);
   sound(720);
   delay(200);
   sound(670);
   delay(250);
   sound(620);
   delay(300);
   sound(770);
   delay(500);
   sound(720);
   delay(200);
   sound(670);
   delay(300);
   sound(580);
   delay(350);
   sound(670);
   delay(300);
   sound(680);
   delay(200);
   sound(600);
   delay(100);
   sound(505);
   delay(500);
   nosound;END;
VAR D:INTEGER;
BEGIN
 CLRSCR;
 CONTROL(00,$F8,D);
 CONTROL(00,$F9,D);
 CONTROL(00,$FA,D);
 CONTROL(00,$FB,D);
 INTRO_1;
 OUT2_SOUND;
 DISPLAYTITLE;
 DIRECTVIDEO := TRUE;
 CHECKSNOW   := TRUE;
    { DISPLAY MENU }
 INITWIN;
 DISPLAYFUNCKEY;
```

```
            DISPLAYMAINMENU;
            MENUACTIVE(1);
    CURSOROFF;
       { LOOP READ CONTROL KEY }
    REPEAT
      READFUNEY(KEY);
      TESTKEY(KEY);
            UNTIL FINISH;
            INTRO_2 ;
END.
```

การ UNIX เกี่ยวกับรูปภาพ

การสร้าง UNIX เกี่ยวกับรูปภาพเป็น การสร้าง PROGRAM ที่ใช้ในการสร้างรูปภาพทั้งของ DIGITAL TRAINER SOFTWARE ซึ่งจะเป็นการสร้างรูปภาพและคำสั่งต่างๆที่ไม่มีในคำสั่งของ TURBO PASCAL GRAPH สำหรับการสร้างรูปภาพและคำสั่งต่างๆ ก็เป็นการนำคำสั่งหลักที่มีอยู่มาสร้างคำสั่งใหม่ตามความต้องการ ตัวอย่างในการสร้างรูป NOT GATE เราต้องอาศัยคำสั่งหลักที่มีอยู่คือคำสั่ง LINE (XC1,YC1;XC2,YC2) ซึ่งเป็นคำสั่งในการลากเส้นตรงจากจุด XC1,YC1 ไป XC2,YC2 และ คำสั่ง CIRCLE (XC,YC,R) ซึ่งเป็นคำสั่งในการสร้างรูปวงกลมที่มีจุดศูนย์กลางอยู่ที่ XC,YC และมีรัศมีเท่ากับ R โดยการนำคำสั่งทั้ง 2 มาสร้างคำสั่งใหม่ดังนี้

```
PROCEDURE NOT_GATE(XC,YC);

BEGIN
LINE(XC,YC,XC+10,YC+8);          (ลากเส้นเฉียงบน)
LINE(XC,YC,XC-10,YC-8);          (ลากเส้นเฉียงล่าง)
LINE(XC-10,YC-8,XC+10,YC+8);     (ลากเส้นปิดหลัง)
CIRCLE(XC+3,YC,5);               (สร้างวงกลมที่หัว)
END;
```

จาก PROGRAM ดังกล่าวเมื่อต้องการสร้างรูป NOT GATE ที่ตำแหน่งใดก็สามารถทำได้โดยใช้คำสั่ง NOT_GATE(XC,YC) และต้องกำหนดตำแหน่งไปด้วย เช่น NOT_GATE(100,100); หมายถึงการสร้างรูป NOT GATE ที่ตำแหน่ง XC=100 และ YC=100 สำหรับคำสั่งอื่นที่ใช้ใน DIGITAL TRAINER SOFTWARE นั้นจะมีวิธีการเหมือนกับการสร้างรูป NOT GATE ดังที่ได้กล่าวมาแล้ว

<u>UNIX ในการสร้างรูปภาพประกอบด้วย PROGRAM หลัก 3 PROGRAM คือ</u>

- PROGRAM PICTURE.PAS (เป็น PROGRAM ที่รวมคำสั่งที่สร้างขึ้นใหม่)
- PROGRAM INTRO.PAS (เป็น PROGRAM ที่ใช้สร้างภาพใน MENU 1)
- PROGRAM DATA_PIN.PAS (เป็น PROGRAM ที่ใช้สร้างภาพใน MENU 2)

```
{*************************************************}
{*                                               *}
{*      THIS PROGRAM IS CATALOG OF GRAPHIC GATE  *}
{*                                               *}
{*************************************************}


UNIT PICTURE;
INTERFACE

USES GRAPH,CRT;
VAR I,J,XC,YC: INTEGER;
PROCEDURE NAND_GATE  (XC,YC:INTEGER);
PROCEDURE AND_GATE   (XC,YC:INTEGER);
PROCEDURE NOR_GATE   (XC,YC:INTEGER);
PROCEDURE OR_GATE    (XC,YC:INTEGER);
PROCEDURE NOR_GATE2   (XC,YC:INTEGER);
PROCEDURE EXOR_GATE  (XC,YC:INTEGER);
PROCEDURE EXNOR_GATE (XC,YC:INTEGER);
PROCEDURE NOT_GATE   (XC,YC:INTEGER);
PROCEDURE BLOCK_14   (XC,YC:INTEGER);
PROCEDURE BLOCK_16   (XC,YC:INTEGER);
PROCEDURE RS_FF      (XC,YC:INTEGER);
PROCEDURE JK_FF      (XC,YC:INTEGER);
PROCEDURE D_FF       (XC,YC:INTEGER);
PROCEDURE D1_FF      (XC,YC:INTEGER);
PROCEDURE JK1_FF     (XC,YC:INTEGER);
PROCEDURE D2_FF      (XC,YC:INTEGER);
PROCEDURE JK2_FF     (XC,YC:INTEGER);
PROCEDURE TRUTH_NAND (XC,YC:INTEGER);
PROCEDURE TRUTH_AND  (XC,YC:INTEGER);
PROCEDURE TRUTH_NOR  (XC,YC:INTEGER);
PROCEDURE TRUTH_OR   (XC,YC:INTEGER);
PROCEDURE TRUTH_EXOR (XC,YC:INTEGER);
PROCEDURE TRUTH_EXNOR(XC,YC:INTEGER);
PROCEDURE TRUTH_NOT  (XC,YC:INTEGER);
PROCEDURE TRUTH_RS   (XC,YC:INTEGER);
PROCEDURE TRUTH_JK   (XC,YC:INTEGER);
PROCEDURE TRUTH_D    (XC,YC:INTEGER);

PROCEDURE TEST_PIC   (XC,YC:INTEGER);

{END OF INTERFACE SECTION}
IMPLEMENTATION

          {****************************}
          {*       NAND GATE          *}
          {****************************}
```

```
PROCEDURE NAND_GATE(XC,YC:INTEGER);
BEGIN
 ELLIPSE (XC,YC,250,109,33,12);
 LINE (XC-12,YC+10,XC-12,YC-10);
 CIRCLE (XC+38,YC,5);
END;
          {*****************************}
          {*         AND GATE          *}
          {*****************************}

 PROCEDURE AND_GATE(XC,YC:INTEGER);
 BEGIN
  ELLIPSE (XC,YC,250,109,33,12);
  LINE (XC-12,YC+10,XC-12,YC-10);
 END;
          {*****************************}
          {*         NOR GATE          *}
          {*****************************}

 PROCEDURE NOR_GATE(XC,YC:INTEGER);
 BEGIN
  ELLIPSE (XC-10,YC,270,90,40,12);
  ELLIPSE (XC-40,YC,333,28,33,26);
  CIRCLE (XC+36,YC,5);
 END;
          {*****************************}
          {*    OR GATE  (STYLE 1)     *}
          {*****************************}
 PROCEDURE OR_GATE(XC,YC:INTEGER);
 BEGIN
  ELLIPSE (XC-10,YC,270,90,40,12);
  ELLIPSE (XC-40,YC,333,28,33,26);
 END;
          {*****************************}
          {*    NOR GATE (STYLE 2)     *}
          {*****************************}
 PROCEDURE NOR_GATE2(XC,YC:INTEGER);
 BEGIN
  ELLIPSE (XC+29,YC,87,273,40,12);
  ELLIPSE (XC+62,YC,152,207,33,26);
  CIRCLE (XC-17,YC,5);
 END;
          {*****************************}
          {*       EX_OR GATE          *}
          {*****************************}
 PROCEDURE EXOR_GATE(XC,YC:INTEGER);
 BEGIN
```

```
 ELLIPSE (XC-10,YC,270,90,40,12);
 ELLIPSE (XC-40,YC,333,28,33,26);
 ELLIPSE (XC-44,YC,333,28,33,26);END;
         {*****************************}
         {*       EX_NOR GATE         *}
         {*****************************}
PROCEDURE EXNOR_GATE(XC,YC:INTEGER);
BEGIN
 ELLIPSE (XC-10,YC,270,90,40,12);
 ELLIPSE (XC-40,YC,333,28,33,26);
 ELLIPSE (XC-44,YC,333,28,33,26);
 CIRCLE (XC+36,YC,5);END;
         {*****************************}
         {*          NOT GATE         *}
         {*****************************}
PROCEDURE NOT_GATE(XC,YC:INTEGER);
BEGIN
 LINE (XC-10,YC-10,XC-10,YC+10);
 LINE (XC-10,YC-10,XC+10,YC);
 LINE (XC-10,YC+10,XC+10,YC);
 CIRCLE (XC+14,YC,5);END;
         {*****************************}
         {*       BLOCK_IC(14PIN)     *}
         {*****************************}
 PROCEDURE BLOCK_14(XC,YC:INTEGER);
 BEGIN
  RECTANGLE (XC-210,YC-60,XC+210,YC+60);
  PIESLICE (XC-210,YC,90,0,20);
  PIESLICE (XC-210,YC,270,360,20);
  RECTANGLE (XC-13,YC+60,XC+13,YC+70);
  RECTANGLE (XC-43,YC+60,XC-69,YC+70);
  RECTANGLE (XC-99,YC+60,XC-125,YC+70);
  RECTANGLE (XC-155,YC-60,XC-181,YC+70);
  RECTANGLE (XC+43,YC+60,XC+69,YC+70);
  RECTANGLE (XC+99,YC+60,XC+125,YC+70);
  RECTANGLE (XC+155,YC+60,XC+181,YC+70);
  RECTANGLE (XC-13,YC-60,XC+13,YC-70);
  RECTANGLE (XC-43,YC-60,XC-69,YC-70);
  RECTANGLE (XC-99,YC-60,XC-125,YC-70);
  RECTANGLE (XC-155,YC-60,XC-181,YC-70);
  RECTANGLE (XC+43,YC-60,XC+69,YC-70);
  RECTANGLE (XC+99,YC-60,XC+125,YC-70);
  RECTANGLE (XC+155,YC-60,XC+181,YC-70);
  SETTEXTSTYLE(2,0,4);
  OUTTEXTXY (XC-168,YC+60,'1');
  OUTTEXTXY (XC-112,YC+60,'2');
  OUTTEXTXY (XC-56,YC+60,'3');
  OUTTEXTXY (XC,YC+60,'4');
```

```
    OUTTEXTXY (XC+56,YC+60,'5');
    OUTTEXTXY (XC+112,YC+60,'6');
    OUTTEXTXY (XC+168,YC+60,'7');
    OUTTEXTXY (XC+168,YC-70,'8');
    OUTTEXTXY (XC+112,YC-70,'9');
    OUTTEXTXY (XC+50,YC-70,'10');
    OUTTEXTXY (XC-5,YC-70,'11');
    OUTTEXTXY (XC-59,YC-70,'12');
    OUTTEXTXY (XC-115,YC-70,'13');
    OUTTEXTXY (XC-171,YC-70,'14');END;
     {*****************************}
     {*      BLOCK_IC(16PIN)      *}
     {*****************************}
   PROCEDURE BLOCK_16(XC,YC:INTEGER);
   BEGIN
    XC:=XC-30;
    RECTANGLE (XC-210,YC-60,XC+265,YC+60);
    PIESLICE (XC-210,YC,90,0,20);
    PIESLICE (XC-210,YC,270,360,20);
    RECTANGLE (XC-13,YC+60,XC+13,YC+70);
    RECTANGLE (XC-43,YC+60,XC-69,YC+70);
    RECTANGLE (XC-99,YC+60,XC-125,YC+70);
    RECTANGLE (XC-155,YC+60,XC-181,YC+70);
    RECTANGLE (XC+43,YC+60,XC+69,YC+70);
RECTANGLE (XC+155,YC+60,XC+181,YC+70);
    RECTANGLE (XC+211,YC+60,XC+237,YC+70);
    RECTANGLE (XC-13,YC-60,XC+13,YC-70);
    RECTANGLE (XC-43,YC-60,XC-69,YC-70);
    RECTANGLE (XC-99,YC-60,XC-125,YC-70);
    RECTANGLE (XC-155,YC-60,XC-181,YC-70);
    RECTANGLE (XC+43,YC-60,XC+69,YC-70);
    RECTANGLE (XC+99,YC-60,XC+125,YC-70);
    RECTANGLE (XC+155,YC-60,XC+181,YC-70);
    RECTANGLE (XC+211,YC-60,XC+237,YC-70);
    SETTEXTSTYLE(2,0,4);
    OUTTEXTXY (XC-168,YC+60,'1');
    OUTTEXTXY (XC-112,YC+60,'2');
    OUTTEXTXY (XC-56,YC+60,'3');
    OUTTEXTXY (XC,YC+60,'4');
    OUTTEXTXY (XC+56,YC+60,'5');
    OUTTEXTXY (XC+112,YC+60,'6');
    OUTTEXTXY (XC+168,YC+60,'7');
    OUTTEXTXY (XC+221,YC+60,'8');
    OUTTEXTXY (XC+168,YC-70,'10');
    OUTTEXTXY (XC+112,YC-70,'11');
    OUTTEXTXY (XC+50,YC-70,'12');
    OUTTEXTXY (XC-5,YC-70,'13');
    OUTTEXTXY (XC-59,YC-70,'14');
    OUTTEXTXY (XC-115,YC-70,'15');
```

```
    OUTTEXTXY (XC-171,YC-70,'16');
    END;
     {*****************************}
     {*       BLOCK RS_FF         *}
     {*****************************}
PROCEDURE RS_FF(XC,YC:INTEGER);
BEGIN
  RECTANGLE (XC-30,YC-30,XC+30,YC+30);
  SETTEXTSTYLE (2,0,6);
  OUTTEXTXY (XC-26,YC+10,'S');
  OUTTEXTXY (XC-26,YC-25,'R');
  OUTTEXTXY (XC+20,YC+10,'Q');
  OUTTEXTXY (XC+20,YC+3,'-');
  OUTTEXTXY (XC+20,YC-25,'Q');
END;
     {*****************************}
     {*       BLOCK JK_FF         *}
     {*****************************}
PROCEDURE JK_FF(XC,YC:INTEGER
BEGIN
  RECTANGLE (XC-30,YC-30,XC+30,YC+30);
  SETTEXTSTYLE (2,0,6);
  OUTTEXTXY (XC-26,YC+10,'K');
  OUTTEXTXY (XC-26,YC-25,'J');
  OUTTEXTXY (XC+20,YC+10,'Q');
  OUTTEXTXY (XC+20,YC+3,'-');
  OUTTEXTXY (XC+20,YC-25,'Q');
END;
     {*****************************}
     {*       BLOCK D_FF          *}
     {*****************************}
PROCEDURE D_FF(XC,YC:INTEGER);
BEGIN
  RECTANGLE (XC-30,YC-30,XC+30,YC+30);
  SETTEXTSTYLE (2,0,6);
  OUTTEXTXY (XC-26,YC+10,'CLK');
  OUTTEXTXY (XC-26,YC-25,'D');
  OUTTEXTXY (XC+20,YC+10,'Q');
  OUTTEXTXY (XC+20,YC+3,'-');
  OUTTEXTXY (XC+20,YC-25,'Q');
END;
     {*****************************}
     {* BLOCK  D1_FF (CLK,CLR,PR)*}
     {*****************************}
PROCEDURE D1_FF(XC,YC:INTEGER);
BEGIN
  RECTANGLE (XC-40,YC-35,XC+40,YC+35);
  SETTEXTSTYLE (2,0,6);
  OUTTEXTXY (XC-31,YC+4,'D');
```

```
  OUTTEXTXY (XC+26,YC+10,'Q');
  OUTTEXTXY (XC+26,YC-32,'-');
  OUTTEXTXY (XC+26,YC-25,'Q');
  SETTEXTSTYLE (2,0,5);
  OUTTEXTXY (XC-28,YC-20,'CLK');
  OUTTEXTXY (XC-6,YC+18,'PR');
  OUTTEXTXY (XC-8,YC-33,'CLR');
  YC:=YC-24;
  LINE (XC-31,YC+10,XC-39,YC+14);
  LINE (XC-31,YC+10,XC-39,YC+6);
  YC:=YC+24;
  CIRCLE (XC,YC+39,5);
  CIRCLE (XC,YC-39,5);END;
    {****************************}
    {* BLOCK  D2_FF (CLK,CLR,PR)*}
    {****************************}
PROCEDURE D2_FF(XC,YC:INTEGER);
BEGIN
  RECTANGLE (XC-40,YC-35,XC+40,YC+35);
  SETTEXTSTYLE (2,0,6);
  OUTTEXTXY (XC-31,YC-25,'D');
  OUTTEXTXY (XC+26,YC+10,'Q');
  OUTTEXTXY (XC+26,YC+3,'-');
  OUTTEXTXY (XC+26,YC-2'Q');
  SETTEXTSTYLE (2,0,5);
  OUTTEXTXY (XC-28,YC+4,'CLK');
  OUTTEXTXY (XC-10,YC+20,'CLR');
  OUTTEXTXY (XC-8,YC-33,'PR');
  LINE (XC-31,YC+10,XC-39,YC+14);
  LINE (XC-31,YC+10,XC-39,YC+6);
  CIRCLE (XC,YC+39,5);
  CIRCLE (XC,YC-39,5);
  END;
    {****************************}
    {* BLOCK JK1_FF (CLK,CLR,PR)*}
    {****************************}
PROCEDURE JK1_FF(XC,YC:INTEGER);
BEGIN
  RECTANGLE (XC-40,YC-35,XC+40,YC+35);
  SETTEXTSTYLE (2,0,6);
  OUTTEXTXY (XC-31,YC+10,'K');
  OUTTEXTXY (XC-31,YC-25,'J');
  OUTTEXTXY (XC+26,YC+10,'Q');
  OUTTEXTXY (XC+26,YC+3,'-');
  OUTTEXTXY (XC+26,YC-25,'Q');
  SETTEXTSTYLE (2,0,5);
  OUTTEXTXY (XC-28,YC-6,'CLK');
  OUTTEXTXY (XC-9,YC+20,'CLR');
  OUTTEXTXY (XC-5,YC-33,'PR ');
```

```
  YC:=YC-10;
  LINE (XC-31,YC+10,XC-39,YC+14);
  LINE (XC-31,YC+10,XC-39,YC+6);
  YC:=YC+10;
  CIRCLE (XC,YC+39,5);
  CIRCLE (XC,YC-39,5);
  CIRCLE (XC-46,YC,5);
  END;
    {*****************************}
    {* BLOCK JK2_FF (CLK,CLR,PR)*}
    {*****************************}
PROCEDURE JK2_FF(XC,YC:INTEGER);
BEGIN
  RECTANGLE (XC-40,YC-35,XC+40,YC+35);
  SETTEXTSTYLE (2,0,6);
  OUTTEXTXY (XC-31,YC+10,'J');
  OUTTEXTXY (XC-31,YC-25,'K');
  OUTTEXTXY (XC+26,YC+10,'Q');
  OUTTEXTXY (XC+26,YC-32,'-');
  OUTTEXTXY (XC+26,YC-25,'Q');
  SETTEXTSTYLE (2,0,5);
  OUTTEXTXY (XC-28,YC-6,'CLK');
  OUTTEXTXY (XC-6 ,YC+20,'PR');
  OUTTEXTXY (XC-10,YC-33,'CLR');
  YC:=YC-10;
  LINE (XC-31,YC+10,XC-39,YC+14);
  LINE (XC-31,YC+10,XC-39,YC+6);
  YC:=YC+10;
  CIRCLE (XC,YC+39,5);
 CIRCLE (XC,YC-39,5);
  CIRCLE (XC-46,YC,5);
  END;

    {*****************************}
    {*   TRUTH_TABLE  OR-GATE    *}
    {*****************************}

  PROCEDURE TRUTH_OR(XC,YC: INTEGER);
  BEGIN
  RECTANGLE (XC-80,YC-85,XC+40,YC+50);
  RECTANGLE (XC-40,YC-85,XC+40,YC+50);
  RECTANGLE (XC,YC-85,XC+40,YC+50);
  RECTANGLE (XC-80,YC-50,XC+40,YC+50);
  RECTANGLE (XC-80,YC-25,XC+40,YC+50);
  RECTANGLE (XC-80,YC,XC+40,YC+50);
  RECTANGLE (XC-80,YC+25,XC+40,YC+50);

  SETTEXTSTYLE (0,0,2);
```

```
OUTTEXTXY (XC-67,YC-76,'A');
OUTTEXTXY (XC-27,YC-76,'B');
OUTTEXTXY (XC+12,YC-76,'Y');
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC+12,YC-47,'0');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'1');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'1');
OUTTEXTXY (XC-67,YC+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+12,YC+28,'1');
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-90,YC+55,'TRUTH  TABLE');
 END;
  {***************************}
  {*   TRUTH_TABLE AND-GATE   *}
  {***************************}
PROCEDURE TRUTH_AND(XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-80,YC-85,XC+40,YC+50);
RECTANGLE (XC-40,YC-85,XC+40,YC+50);
RECTANGLE (XC,YC-85,XC+40,YC+50);
RECTANGLE (XC-80,YC-50,XC+40,YC+50);
RECTANGLE (XC-80,YC-25,XC+40,YC+50);
RECTANGLE (XC-80,YC,XC+40,YC+50);
RECTANGLE (XC-80,YC+25,XC+40,YC+50);
SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-76,'A');
OUTTEXTXY (XC-27,YC-76,'B');
OUTTEXTXY (XC+12,YC-76,'Y');
SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC+12,YC-47,'0');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'0');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'0');
OUTTEXTXY (XC-67,YC+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+12,YC+28,'1');
SETTEXTSTYLE(1,0,2);
```

```
OUTTEXTXY(XC-90,YC+55,'TRUTH  TABLE');
 END;
  {*****************************}
  {*   TRUTH_TABLE NAND-GATE  *}
  {*****************************}
PROCEDURE TRUTH_NAND(XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-80,YC-85,XC+40,YC+50);
RECTANGLE (XC-40,YC-85,XC+40,YC+50);
RECTANGLE (XC,YC-85,XC+40,YC+50);
RECTANGLE (XC-80,YC-50,XC+40,YC+50);
RECTANGLE (XC-80,YC-25,XC+40,YC+50);
RECTANGLE (XC-80,YC,XC+40,YC+50);
RECTANGLE (XC-80,YC+25,XC+40,YC+50);

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-76,'A');
OUTTEXTXY (XC-27,YC-76,'B');
OUTTEXTXY (XC+12,YC-76,'Y');

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC+12,YC-47,'1');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'1');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'1');
OUTTEXTXY (XC-67,+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+12,YC+28,'0');
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-90,YC+55,'TRUTH  TABLE');
 END;
  {*****************************}
  {*   TRUTH_TABLE  NOR-GATE  *}
  {*****************************}
PROCEDURE TRUTH_NOR(XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-80,YC-85,XC+40,YC+50);
RECTANGLE (XC-40,YC-85,XC+40,YC+50);
RECTANGLE (XC,YC-85,XC+40,YC+50);
RECTANGLE (XC-80,YC-50,XC+40,YC+50);
RECTANGLE (XC-80,YC-25,XC+40,YC+50);
RECTANGLE (XC-80,YC,XC+40,YC+50);
RECTANGLE (XC-80,YC+25,XC+40,YC+50);
```

```
SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-76,'A');
OUTTEXTXY (XC-27,YC-76,'B');
OUTTEXTXY (XC+12,YC-76,'Y');
SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC+12,YC-47,'1');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'0');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'0');
OUTTEXTXY (XC-67,YC+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+12,YC+28,'0');
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-90,YC+55,'TRUTH  TABLE');
 END;
  {*****************************}
  {*   TRUTH_TABLE EXOR-GATE   *}
  {*****************************}
PROCEDURE TRUTH_EXOR(XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-80,YC-85,XC+40,YC+50);
RENGLE (XC-40,YC-85,XC+40,YC+50);
RECTANGLE (XC,YC-85,XC+40,YC+50);
RECTANGLE (XC-80,YC-50,XC+40,YC+50);
RECTANGLE (XC-80,YC-25,XC+40,YC+50);
RECTANGLE (XC-80,YC,XC+40,YC+50);
RECTANGLE (XC-80,YC+25,XC+40,YC+50);

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-76,'A');
OUTTEXTXY (XC-27,YC-76,'B');
OUTTEXTXY (XC+12,YC-76,'Y');

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC+12,YC-47,'0');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'1');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'1');
```

```
OUTTEXTXY (XC-67,YC+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+12,YC+28,'0');
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-90,YC+55,'TRUTH  TABLE');
 END;
  {*****************************}
  {*   TRUTH_TABLE EXNOR-GATE *}
  {*****************************}
PROCEDURE TRUTH_EXNOR(XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-80,YC-85,XC+40,YC+50);
RECTANGLE (XC-40,YC-85,XC+40,YC+50);
RECTANGLE (XC,YC-85,XC+40,YC+50);
RECTANGLE (XC-80,YC-50,XC+40,YC+50);
RECTANGLE (XC-80,YC-25,XC+40,YC+50);
RECTANGLE (XC-80,YC,XC+40,YC+50);
RECTANGLE (XC-80,YC+25,XC+40,YC+50);

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-76,'A');
OUTTEXTXY (XC-27,YC-76,'B');
OUTTEXTXY (XC+12,YC-76,'Y');

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC+12,YC-47,'1');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'0');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'0');
OUTTEXTXY (XC-67,YC+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+12,YC+28,'1');
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-90,YC+55,'TRUTH  TABLE');
 END;

  {*****************************}
  {*   TRUTH_TABLE   NOT-GATE *}
  {*****************************}

PROCEDURE TRUTH_NOT(XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-40,YC-25,XC+40,YC+50);
```

```
OUTTEXTXY (XC-291,YC+155,' Note that the negative going edge of');
OUTTEXTXY (XC-8,YC+155, 'the CLK signal has no effect on FF - ');
OUTTEXTXY (XC-291,YC+170,'output.                                ');
        END;


        END.
```

```
RECTANGLE (XC,YC-25,XC+40,YC+50);
RECTANGLE (XC-40,YC-25,XC+40,YC+50);
RECTANGLE (XC-40,YC,XC+40,YC+50);
RECTANGLE (XC-40,YC+25,XC+40,YC+50);
SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-27,YC-22,'A');
OUTTEXTXY (XC+12,YC-22,'Y');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+12,YC+28,'0');
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-70,YC+55,'TRUTH  TABLE');
END;
  {*****************************}
  {*   TRUTH_TABLE RS_FIFLOP  *}
  {*****************************}

PROCEDURE TRUTH_RS  (XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-80,YC-85,XC+80,YC+50);
RECTANGLE (XC-40,YC-85,XC+80,YC+50);
RECTANGLE (XC,YC-85,XC+80,YC+50);
RECTANGLE (XC-80,YC-50,XC+80,YC+50);
RECTANGLE (XC-80,YC-25,XC+80,YC+50);
RECTANGLE (XC-80,YC,XC+80,YC+50);
RECTANGLE (XC-80,YC+25,XC+80,YC+50);
SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-76,'R');
OUTTEXTXY (XC-27,YC-76,'S');
OUTTEXTXY (XC+12,YC-76,'Q');
OUTTEXTXY (XC+52,YC-76,'Q');
OUTTEXTXY (XC+53,YC-90,'-');
SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'0');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'1');
OUTTEXTXY (XC-67,YC+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+52,YC-22,'1');
OUTTEXTXY (XC+52,YC+3 ,'0');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC+6 ,YC-43,'NO CHANGE ');
```

```
OUTTEXTXY (XC+17,YC+32  ,'AVOID ');
OUTTEXTXY (XC+52,YC+28,' ');
OUTTEXTXY (XC+52,YC-47,' ');

SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-77,YC+55,'TRUTH  TABLE');
END;
  {*****************************}
  {*   TRUTH_TABLE JK_FIFLOP  *}
  {*****************************}

PROCEDURE TRUTH_JK  (XC,YC: INTEGER);
BEGIN
RECTANGLE (XC-80,YC-85,XC+80,YC+50);
RECTANGLE (XC-40,YC-85,XC+80,YC+50);
RECTANGLE (XC,YC-85,XC+80,YC+50);
RECTANGLE (XC-80,YC-50,XC+80,YC+50);
RECTANGLE (XC-80,YC-25,XC+80,YC+50);
RECTANGLE (XC-80,YC,XC+80,YC+50);
RECTANGLE (XC-80,YC+25,XC+80,YC+50);

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-76,'J');
OUTTEXTXY (XC-27,YC-76,'K');
OUTTEXTXY (XC+12,YC-76,'Q');
OUTTEXTXY (XC+52,YC-76,'Q');
OUTTEXTXY (XC+53,YC-90,'-');

SETTEXTSTYLE (0,0,2);
OUTTEXTXY (XC-67,YC-47,'0');
OUTTEXTXY (XC-27,YC-47,'0');
OUTTEXTXY (XC-67,YC-22,'0');
OUTTEXTXY (XC-27,YC-22,'1');
OUTTEXTXY (XC+12,YC-22,'0');
OUTTEXTXY (XC-67,YC+3,'1');
OUTTEXTXY (XC-27,YC+3,'0');
OUTTEXTXY (XC+12,YC+3,'1');
OUTTEXTXY (XC-67,YC+28,'1');
OUTTEXTXY (XC-27,YC+28,'1');
OUTTEXTXY (XC+52,YC-22,'1');
OUTTEXTXY (XC+52,YC+3 ,'0');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC+6 ,YC-43,'NO CHANGE ');
OUTTEXTXY (XC+17,YC+32  ,'TOGGLE');
OUTTEXTXY (XC+52,YC+28,' ');
OUTTEXTXY (XC+52,YC-47,' ');
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(XC-77,YC+55,'TRUTH  TABLE');
```

```
  END;

    {*****************************}
    {*   TRUTH_TABLE  D_FIFLOP   *}
    {*****************************}

  PROCEDURE TRUTH_D   (XC,YC: INTEGER);
  BEGIN
  RECTANGLE (XC-80,YC-85,XC+80,YC   );
  RECTANGLE (XC-40,YC-85,XC+80,YC   );
  RECTANGLE (XC,YC-85,XC+80,YC   );
  RECTANGLE (XC-80,YC-50,XC+80,YC   );
  RECTANGLE (XC-80,YC-25,XC+80,YC  );
  RECTANGLE (XC-80,YC,XC+80,YC       );

  SETTEXTSTYLE (0,0,2);
  OUTTEXTXY (XC-72,YC-76,'CK');
  OUTTEXTXY (XC-27,YC-76,'D');
  OUTTEXTXY (XC+12,YC-76,'Q');
  OUTTEXTXY (XC+52,YC-76,'Q');
  OUTTEXTXY (XC+53,YC-90,'-');

  SETTEXTSTYLE (0,0,2);
  OUTTEXTXY (XC-27,YC-47,'0');
  OUTTEXTXY (XC-27,YC-22,'1');
  OUTTEXTXY (XC+12,YC-22,'1');
  OUTTEXTXY (XC+52,YC-22,'0');
  OUTTEXTXY (XC+12,YC-47,'0 ');
  OUTTEXTXY (XC+52,YC-47,'1');
  LINE(XC-63,YC-32,XC-63,YC-42);
  LINE(XC-63,YC-32,XC-73,YC-32);
  LINE(XC-63,YC-42,XC-48,YC-42);
  LINE(XC-63,YC-8,XC-63,YC-18 );
  LINE(XC-63,YC-8,XC-73,YC-8);
  LINE(XC-63,YC-18,XC-48,YC-18 );
  SETTEXTSTYLE(1,0,2);
  OUTTEXTXY(XC-77,YC+10,'TRUTH  TABLE');
   END;

  PROCEDURE TEST_PIC(XC,YC:INTEGER);
  BEGIN
  TRUTH_NOR(XC,YC);
  END;
END.
```

```
{*****************************************************}
{*                                                   *}
{*          THIS PROGRAM IS DATA OF MENU_1           *}
{*                                                   *}
{*****************************************************}

UNIT  INTRO ;
INTERFACE
USES CRT,PICTURE,GRAPH ;
VAR I,J,XC,YC :INTEGER;
PROCEDURE INTRO_OR   (XC,YC :INTEGER);
PROCEDURE INTRO_AND  (XC,YC :INTEGER);
PROCEDURE INTRO_NAND (XC,YC :INTEGER);
PROCEDURE INTRO_NOR  (XC,YC :INTEGER);
PROCEDURE INTRO_EXOR (XC,YC :INTEGER);
PROCEDURE INTRO_EXNOR(XC,YC :INTEGER);
PROCEDURE INTRO_NOT  (XC,YC :INTEGER);
PROCEDURE INTRO_RS   (XC,YC :INTEGER);
PROCEDURE INTRO_JK   (XC,YC :INTEGER);
PROCEDURE INTRO_D    (XC,YC :INTEGER);

IMPLEMENTATION


   {*************************************}
   {*      INTRODUCTION TO OR_GATE      *}
   {*************************************}

 PROCEDURE INTRO_OR(XC,YC :INTEGER);
  BEGIN
  YC:=YC-30;
  SETTEXTSTYLE (1,0,4);
  OUTTEXTXY (XC-130,YC-150,'*** OR GATE ***');
  OUTTEXTXY (XC-125,YC-137,'  --------  ');
  OR_GATE(XC-150,YC-30);
  LINE(XC-158,YC-23,XC-196,YC-23);
  LINE(XC-158,YC-37,XC-196,YC-37);
  LINE(XC-119,YC-30,XC-87,YC-30);
  CIRCLE(XC-201,YC-23,5);
  CIRCLE(XC-201,YC-37,5);
  CIRCLE(XC-82,YC-30,5);
  SETTEXTSTYLE(1,0,1);
  OUTTEXTXY (XC-220,YC-26,'A');
  OUTTEXTXY (XC-222,YC-53,'B');
  OUTTEXTXY (XC-63,YC-40,'Y');
  TRUTH_OR(XC+190,YC);
  OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');
  SETTEXTSTYLE(2,0,5);
```

```
 OUTTEXTXY (XC-291,YC+95, ' The OR gate produces a one-output wh');
 OUTTEXTXY (XC,YC+95,     'en any input is at level . its mathe-');
 OUTTEXTXY (XC-291,YC+110,' matical expression is Y=A+B ,where t');
 OUTTEXTXY (XC-6,YC+110,  'he + stands for the OR operation and ');
 OUTTEXTXY (XC-291,YC+125,' not normal addition. for a three-inp');
 OUTTEXTXY (XC-13,YC+125, 'ut OR gate, it would be Y=A+B+C; and ');
 OUTTEXTXY (XC-291,YC+140,' so on.                              ');
 END;

  {*************************************}
  {*       INTRODUCTION TO AND_GATE      *}
  {*************************************}

PROCEDURE INTRO_AND(XC,YC :INTEGER);
 BEGIN
 YC:=YC-30;
 SETTEXTSTYLE (1,0,4);
 OUTTEXTXY (XC-130,YC-150,'*** AND GATE ***');
 OUTTEXTXY (XC-125,YC-137,'  ---------  ');
 AND_GATE(XC-150,YC-30);
 LINE(XC-162,YC-23,XC-196,YC-23);
 LINE(XC-162,YC-37,XC-196,YC-37);
 LINE(XC-117,YC-30,XC-87,YC-30);
 CIRCLE(XC-201,YC-23,5);
 CIRCLE(XC-201,YC-37,5);
 CIRCLE(XC-82,YC-30,5);
 SETTEXTSTYLE(1,0,1);
 OUTTEXTXY (XC-220,YC-26,'A');
 OUTTEXTXY (XC-222,YC-53,'B');
 OUTTEXTXY (XC-63,YC-40,'Y');
 TRUTH_AND(XC+190,YC);
 OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');

         SETTEXTSTYLE(2,0,5);
 OUTTEXTXY (XC-291,YC+95, 'The AND gate operates such that the o');
 OUTTEXTXY (XC-4,YC+95    ,'utput will be at the 1 level only    ');
 OUTTEXTXY (XC-291,YC+110,'when all inputs are 1. The mathematic');
 OUTTEXTXY (XC-13,YC+110, 'al expression for the two-input AND  ');
 OUTTEXTXY (XC-291,YC+125,'gate is written as Y=AB ,the same as ');
 OUTTEXTXY (XC-10,YC+125,  'ordinary multipication . For a three-');
 OUTTEXTXY (XC-291,YC+140,'input gate ,it would be Y=ABC ; and s');
 OUTTEXTXY (XC-10,YC+140, 'o on for more inputs                 ');
 END;

  {*************************************}
  {*       INTRODUCTION TO NAND_GATE     *}
  {*************************************}
PROCEDURE INTRO_NAND(XC,YC :INTEGER);
 BEGIN
```

```
          YC:=YC-30;
          SETTEXTSTYLE (1,0,4);
          OUTTEXTXY (XC-130,YC-150,'*** NAND GATE ***');
          OUTTEXTXY (XC-125,YC-137,'  ----------  ');
          NAND_GATE(XC-150,YC-30);
          LINE(XC-162,YC-23,XC-196,YC-23);
          LINE(XC-162,YC-37,XC-196,YC-37);
          LINE(XC-108,YC-30,XC-87,YC-30);
          CIRCLE(XC-201,YC-23,5);
          CIRCLE(XC-201,YC-37,5);
          CIRCLE(XC-80,YC-30,5);
          SETTEXTSTYLE(1,0,1);
          OUTTEXTXY (XC-220,YC-26,'A');
          OUTTEXTXY (XC-222,YC-53,'B');
          OUTTEXTXY (XC-63,YC-40,'Y');
          TRUTH_NAND(XC+190,YC);
          OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');
          SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+95, 'The NAND gate combines the AND and NO');
OUTTEXTXY (XC  ,YC+95,   'T operation,such that the output will');
OUTTEXTXY (XC-291,YC+110,'be 0 only when all inputs are 1. It l');
OUTTEXTXY (XC-15,YC+110, 'ogic expression is Y= AB , which indi');
OUTTEXTXY (XC-291,YC+125,'cates that inputs A and B are first  ');
OUTTEXTXY (XC-5,YC+125,  'ANDed and the result inverted. Thus, ');
OUTTEXTXY (XC-291,YC+140,'a NAND gate always produces an output');
OUTTEXTXY (XC-5 ,YC+140,'  that is the inverse (opposite) of ,');
OUTTEXTXY (XC-291,YC+155,'an AND gate.                        ');
          LINE (XC+155,YC+109,XC+168,YC+109);
        END;

          {************************************}
          {*      INTRODUCTION TO NOR_GATE    *}
          {************************************}
         PROCEDURE INTRO_NOR (XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          SETTEXTSTYLE (1,0,4);
          OUTTEXTXY (XC-130,YC-150,'*** NOR  GATE ***');
          OUTTEXTXY (XC-125,YC-137,'  ----------  ');
          NOR_GATE(XC-150,YC-30);
          LINE(XC-160,YC-23,XC-196,YC-23);
          LINE(XC-160,YC-37,XC-196,YC-37);
          LINE(XC-108,YC-30,XC-87,YC-30);
          CIRCLE(XC-201,YC-23,5);
          CIRCLE(XC-201,YC-37,5);
          CIRCLE(XC-80,YC-30,5);
          SETTEXTSTYLE(1,0,1);
          OUTTEXTXY (XC-220,YC-26,'A');
```

```
            OUTTEXTXY (XC-222,YC-53,'B');
            OUTTEXTXY (XC-63,YC-40,'Y');
            TRUTH_NOR (XC+190,YC);
            OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');

            SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+95, 'The NOR gate combines the OR and NOT ');
OUTTEXTXY (XC-5,YC+95,   'operation such that the output will  ');
OUTTEXTXY (XC-291,YC+110,'be 0 when any input is 1 . Its logic ');
OUTTEXTXY (XC-10,YC+110, 'expression is Y=A+B , which indicates');
OUTTEXTXY (XC-291,YC+125,'that A and B are first  ORed and then');
OUTTEXTXY (XC   ,YC+125, 'the result inverted.A NOR gate always');
OUTTEXTXY (XC-291,YC+140,'give an output an output logic level ');
OUTTEXTXY (XC-15,YC+140, 'that is the inverse  gate.   ');
            LINE (XC+115,YC+109,XC+133,YC+109 );
          END;

            {************************************}
            {*      INTRODUCTION TO EXOR_GATE     *}
            {************************************}

          PROCEDURE INTRO_EXOR (XC,YC :INTEGER);
           BEGIN
           YC:=YC-30;
           SETTEXTSTYLE (1,0,4);
           OUTTEXTXY (XC-130,YC-150,'*** EXOR GATE ***');
           OUTTEXTXY (XC-125,YC-137,'  ----------  ');
           EXOR_GATE(XC-150,YC-30);
           LINE(XC-160,YC-23,XC-196,YC-23);
           LINE(XC-160,YC-37,XC-196,YC-37);
           LINE(XC-119,YC-30,XC-87,YC-30);
           CIRCLE(XC-201,YC-23,5);
           CIRCLE(XC-201,YC-37,5);
           CIRCLE(XC-80,YC-30,5);
           SETTEXTSTYLE(1,0,1);
           OUTTEXTXY (XC-220,YC-26,'A');
           OUTTEXTXY (XC-222,YC-53,'B');
           OUTTEXTXY (XC-63,YC-40,'Y');
           TRUTH_EXOR (XC+190,YC);
           OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');
           SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+110,'The EX-OR gate ( EXCLUSIVE-OR gate ) ');
OUTTEXTXY (XC-5,YC+110,  'produces a 1 output only when the two');
OUTTEXTXY (XC-291,YC+125,'input are at different logic levels .');
OUTTEXTXY (XC-10,YC+125, 'an EX-OR gate always has two input , ')
OUTTEXTXY (XC-291,YC+140,'and its output expression is Y=A + B ');
          CIRCLE (XC-37,YC+146,6);
         END;
```

```
          {************************************}
          {*     INTRODUCTION TO EXNOR_GATE     *}
          {************************************}

        PROCEDURE INTRO_EXNOR (XC,YC :INTEGER);
         BEGIN
         YC:=YC-30;
         SETTEXTSTYLE (1,0,4);
         OUTTEXTXY (XC-130,YC-150,'*** EXNOR GATE ***');
         OUTTEXTXY (XC-125,YC-137,'  ----------- ');
         EXNOR_GATE(XC-150,YC-30);
         LINE(XC-160,YC-23,XC-196,YC-23);
         LINE(XC-160,YC-37,XC-196,YC-37);
         LINE(XC-109,YC-30,XC-87,YC-30);
         CIRCLE(XC-201,YC-23,5);
         CIRCLE(XC-201,YC-37,5);
         CIRCLE(XC-80,YC-30,5);
         SETTEXTSTYLE(1,0,1);
         OUTTEXTXY (XC-220,YC-26,'A');
         OUTTEXTXY (XC-222,YC-53,'B');
       OUTTEXTXY (XC-63,YC-40,'Y');
         TRUTH_EXNOR (XC+190,YC);
         OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');

         SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+110,'The EX-NOR gate (EXCULSIVE-NOR gate) ');
OUTTEXTXY (XC-5,YC+110,  'is the inverse of EX-OR gate .       ');
OUTTEXTXY (XC-291,YC+125,'It produces a 1 output are at the sam');
OUTTEXTXY (XC-8 ,YC+125, 'e logic levele . The EX-NOR output   ');
OUTTEXTXY (XC-291,YC+140,'expression is Y= A + B               ');
         LINE ( XC-160,YC+140,XC-119,YC+140);
         CIRCLE (XC-141,YC+146,6);
       END;

          {************************************}
          {*     INTRODUCTION TO NOT_GATE       *}
          {************************************}

        PROCEDURE INTRO_NOT   (XC,YC :INTEGER);
         BEGIN
         YC:=YC-30;
         SETTEXTSTYLE (1,0,4);
         OUTTEXTXY (XC-127,YC-150,'*** NOT  GATE ***');
         OUTTEXTXY (XC-125,YC-137,'  ----------  ');
         NOT_GATE(XC-140,YC-30);
         LINE(XC-151,YC-30,XC-196,YC-30);
         LINE(XC-120,YC-30,XC-87,YC-30);
         CIRCLE(XC-201,YC-30,5);
```

```
            CIRCLE(XC-80,YC-30,5);
            SETTEXTSTYLE(1,0,1);
            OUTTEXTXY (XC-225,YC-40,'A');
            OUTTEXTXY (XC-63,YC-40,'Y');
            TRUTH_NOT (XC+190,YC-35);
            OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');

            SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+95, 'This is actually not a gate since it ');
OUTTEXTXY (XC-15,YC+95, 'can only have one input.It is commonly');
OUTTEXTXY (XC-291,YC+110,'called an inverter and it produces an');
OUTTEXTXY (XC+5,YC+110,  'output whose logic level is always   ');
OUTTEXTXY (XC-291,YC+125,'opposite of the input logic level.The');
OUTTEXTXY (XC-8,YC+125,'mathematical statement of its operation');
OUTTEXTXY (XC-291,YC+140,'is Y= A .The overbar always indicates');
OUTTEXTXY (XC+5 ,YC+140, 'the inversion (NOT) operation        ');
            LINE (XC-247,YC+140,XC-239,YC+140);
          END;

            {************************************}
            {*     INTRODUCTION TO RS_FIFLOP      *}
            {************************************}

          PROCEDURE INTRO_RS  (XC,YC :INTEGER);
           BEGIN
           YC:=YC-30;
           SETTEXTSTYLE (1,0,4);
           OUTTEXTXY (XC-130,YC-150,'*** RS_FIFLOP ***');
           OUTTEXTXY (XC-125,YC-137,'  ---------   ');
           RS_FF     (XC-140,YC-30);
           LINE(XC-170,YC-14,XC-196,YC-14);
           LINE(XC-170,YC-46,XC-196,YC-46);
           LINE(XC-109,YC-14,XC-87,YC-14);
           LINE(XC-109,YC-46,XC-87,YC-46);
           CIRCLE(XC-201,YC-14,5);
           CIRCLE(XC-201,YC-46,5);
           CIRCLE(XC-80,YC-14,5);
           CIRCLE(XC-80,YC-46,5);
           SETTEXTSTYLE(1,0,1);
           OUTTEXTXY (XC-220,YC-21,' ');
           OUTTEXTXY (XC-222,YC-58,' ');
           OUTTEXTXY (XC-63,YC-21,' ');
           OUTTEXTXY (XC-63,YC-58,' ');
           TRUTH_RS  (XC+190,YC);
           OUTTEXTXY (XC-190,YC-20,' SYMBOL  ');
           SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+95, ' The basic RS-FF has two input , SET  ')
OUTTEXTXY (XC-15,YC+95,'  and RESET . These input are normally');
```

```
OUTTEXTXY (XC-291,YC+110,'both 1.If the SET(S) input is brought ');
OUTTEXTXY (XC-15,YC+110 ,'to 0, the Q output goes to the 1 state');
OUTTEXTXY (XC-291,YC+125, '(and Q =0) .Even if set return to 1 ,');
OUTTEXTXY (XC-15,YC+125,  'owing to the internal feedback. This ');
OUTTEXTXY (XC-291,YC+140, 'is called setting the FF. similarly ,');
OUTTEXTXY (XC-15 ,YC+140, 'when the RESET(R) input is brought to')
OUTTEXTXY (XC-291,YC+155, '0, Q will goto the 0 state and stay  ');
OUTTEXTXY (XC-15 ,YC+155, 'thre. This is called clearing the FF ');
OUTTEXTXY (XC-291,YC+170, 'Note that SET and RESET should not go');
OUTTEXTXY (XC+5,YC+170,   'high simultaneously or an ambiguous  ');
OUTTEXTXY (XC-291,YC+185, 'output state will result.            ');
          LINE (XC-255,YC+123,XC-245,YC+123);
          END;

           {************************************}
           {*     INTRODUCTION TO JK_FIFLOP      *}
           {************************************}
         PROCEDURE INTRO_JK (XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          SETTEXTSTE (1,0,4);
          OUTTEXTXY (XC-130,YC-150,'*** JK_FIFLOP ***');
          OUTTEXTXY (XC-125,YC-137,'  ---------    ');
          JK_FF     (XC-140,YC-30);
          SETTEXTSTYLE(2,0,5);
          OUTTEXTXY (XC-166,YC-36,'CLK');
          LINE(XC-170,YC-14,XC-196,YC-14);
          LINE(XC-170,YC-46,XC-196,YC-46);
          LINE(XC-109,YC-14,XC-87,YC-14);
          LINE(XC-109,YC-46,XC-87,YC-46);
          LINE(XC-170,YC-30,XC-196,YC-30);
          CIRCLE(XC-201,YC-30,5);
          CIRCLE(XC-201,YC-14,5);
          CIRCLE(XC-201,YC-46,5);
          CIRCLE(XC-80,YC-14,5);
          CIRCLE(XC-80,YC-46,5);
          SETTEXTSTYLE(1,0,1);
          OUTTEXTXY (XC-220,YC-21,' ');
          OUTTEXTXY (XC-222,YC-58,' ');
          OUTTEXTXY (XC-63,YC-21,' ');
          OUTTEXTXY (XC-63,YC-58,' ');
          TRUTH_JK  (XC+190,YC);
          OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');
          SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+80, 'The JK-FF has two control inputs, J a');
OUTTEXTXY (XC-3  ,YC+80,   'nd K. when the positive edge of the');
OUTTEXTXY (XC-291,YC+95, 'CLK signal occurs,according to accomp');
OUTTEXTXY (XC-3  ,YC+95,   'anying truth table');
```

```
OUTTEXTXY (XC-291,YC+110,' J=K=0 :If this codition is present (');
OUTTEXTXY (XC-5,YC+110,  'positive edge) there will be no chang');
OUTTEXTXY (XC-291,YC+125,' in the state of FF.;   J=1,K=1 :    ');
OUTTEXTXY (XC-45,YC+125, 'This codition always produces Q=1 on ');
OUTTEXTXY (XC-291,YC+140,' the occurrence of the positive edge ');
OUTTEXTXY (XC-5 ,YC+140,'of CLK signal.;   J=0,K=1 :This always');
OUTTEXTXY (XC-291,YC+155,' produces Q=0 on the occurrence of   ');
OUTTEXTXY (XC-10,YC+155,'the posivetive edge of the CLK signal.')
OUTTEXTXY (XC-291,YC+170,' J=1,K=1 :If this condition is present');
OUTTEXTXY (XC-7 ,YC+170, 'when the positive edge of the CLK   ');
OUTTEXTXY (XC-291,YC+185,'  signal occures,the Q output will swit');
OUTTEXTXY (XC-5 ,YC+185, 'ch to its opposite state (TOGGLE) .  ');
        END;
           {************************************}
           {*      INTRODUCTION TO D_FIFLOP      *}
           {************************************}
         PROCEDURE INTRO_D (XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          SETTEXTSTYLE (1,0,4);
          OUTTEXTXY (XC-130,YC-150,'*** D  FIFLOP ***');
          OUTTEXTXY (XC-125,YC-137,'  ---------   ');
          D_FF     (XC-140,YC-30);
          LINE(XC-170,YC-14,XC-196,YC-14);
          LINE(XC-170,YC-46,XC-196,YC-46);
          LINE(XC-109,YC-14,XC-87,YC-14);
          LINE(XC-109,YC-46,XC-87,YC-46);
          CIRCLE(XC-201,YC-14,5);
          CIRCLE(XC-201,YC-46,5);
          CIRCLE(XC-80,YC-14,5);
          CIRCLE(XC-80,YC-46,5);
          SETTEXTSTYLE(1,0,1);
          OUTTEXTXY (XC-220,YC-21,' ');
          OUTTEXTXY (XC-222,YC-58,' ');
          OUTTEXTXY (XC-63,YC-21,' ');
          OUTTEXTXY (XC-63,YC-58,' ');
          TRUTH_D   (XC+190,YC+13);
          OUTTEXTXY (XC-190,YC+20,' SYMBOL  ');
          SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-291,YC+80, 'Figure shows the symbol and truth tab');
OUTTEXTXY (XC-8,YC+80,  'le for an edge-triggered D_FF. It is ');
OUTTEXTXY (XC-291,YC+95, 'so called because it has a single con');
OUTTEXTXY (XC-8,YC+95,   'trol input, D. This FF operates such ');
OUTTEXTXY (XC-291,YC+110,'that the logic level present at the D');
OUTTEXTXY (XC-10,YC+110,  'input is transferred to the Q output ');
OUTTEXTXY (XC-291,YC+125,'only on the positive going edge of   ');
OUTTEXTXY (XC-30,YC+125, 'the CLK input signal. The D input has');
OUTTEXTXY (XC-291,YC+140,'no effect on Q at any other time.    ');
OUTTEXTXY (XC-10,YC+140, '                                     ');
```

```
{*****************************************************}
{*                                                   *}
{*  THIS PROGRAM IS CATALOG DATA_PIN_IC OF MENU_2    *}
{*                                                   *}
{*****************************************************}
UNIT  DATA_PIN;
INTERFACE
USES CRT,PICTURE,GRAPH ;
VAR I,J,XC,YC :INTEGER;
PROCEDURE PIN_7400 (XC,YC :INTEGER);
PROCEDURE PIN_7408 (XC,YC :INTEGER);
PROCEDURE PIN_7432 (XC,YC :INTEGER);
PROCEDURE PIN_7486 (XC,YC :INTEGER);
PROCEDURE PIN_7402 (XC,YC :INTEGER);
PROCEDURE PIN_7404 (XC,YC :INTEGER);
PROCEDURE PIN_74152(XC,YC :INTEGER);
PROCEDURE PIN_74164(XC,YC :INTEGER);
PROCEDURE PIN_7474 (XC,YC :INTEGER);
PROCEDURE PIN_74393(XC,YC :INTEGER);
PROCEDURE PIN_74390(XC,YC :INTEGER);
PROCEDURE PIN_74112(XC,YC :INTEGER);
PROCEDURE PIN_74283(XC,YC :INTEGER);
PROCEDURE PIN_7485 (XC,YC :INTEGER);
PROCEDURE PIN_74165(XC,YC :INTEGER);
PROCEDURE PIN_74195(XC,YC :INTEGER);
PROCEDURE PIN_74169(XC,YC :INTEGER);
PROCEDURE PIN_74131(XC,YC :INTEGER);
PROCEDURE PIN_74147(XC,YC :INTEGER);
PROCEDURE PIN_74447(XC,YC :INTEGER);
IMPLEMENTATION

   {*************************************}
   {*        DATA OF PIN_IC_7400        *}
   {*************************************}

PROCEDURE PIN_7400(XC,YC :INTEGER);
  BEGIN
  YC:= YC-30;
  BLOCK_14(XC,YC+25);
  NAND_GATE(XC+115,YC);
  NAND_GATE(XC+60,YC+50);
  NAND_GATE(XC-54,YC);
  NAND_GATE(XC-109,YC+50);
  LINE(XC+159,YC,XC+168,YC);
  LINE(XC+168,YC,XC+168,YC-34);
  LINE(XC+101,YC-7,XC+90,YC-7);
  LINE(XC+101,YC+7,XC+56,YC+7);
  LINE(XC+90,YC-7,XC+90,YC-19);
```

```
            LINE(XC+90,YC-19,XC+110,YC-19);
            LINE(XC+110,YC-19,XC+110,YC-34);
            LINE(XC+56,YC+7,XC+56,YC-34);
            LINE(XC-9,YC,XC,YC);
            LINE(XC,YC,XC,YC-34);
            LINE(XC-67,YC-7,XC-78,YC-7);
            LINE(XC-67,YC+7,XC-112,YC+7);
            LINE(XC-78,YC-7,XC-78,YC-19);
            LINE(XC-78,YC-19,XC-58,YC-19);
            LINE(XC-58,YC-19,XC-58,YC-34);
            LINE(XC-112,YC+7,XC-112,YC-34);
            LINE(XC-66,YC+50,XC-57,YC+50);
            LINE(XC-57,YC+50,XC-57,YC+85);
            LINE(XC-122,YC+43,XC-170,YC+43);
            LINE(XC-122,YC+57,XC-135,YC+57);
            LINE(XC-170,YC+43,XC-170,YC+85);
            LINE(XC-135,YC+57,XC-135,YC+70);
            LINE(XC-135,YC+70,XC-114,YC+70);
            LINE(XC-114,YC+70,XC-114,YC+85);
            LINE(XC+104,YC+50,XC+113,YC+50);
            LINE(XC+113,YC+50,XC+113,YC+85);
            LINE(XC+48,YC+43,XC,YC+43);
            LINE(XC+48,YC+57,XC+35,YC+57);
            LINE(XC,YC+43,XC,YC+85);
            LINE(XC+35,YC+57,XC+35,YC+70 );
            LINE(XC+35,YC+70,XC+56,YC+70);
            LINE(XC+56,YC+70,XC+56,YC+85);
            SETTEXTSTYLE (2,0,5);
            OUTTEXTXY (XC+159,YC+100,'GND');
            OUTTEXTXY (XC-178,YC-63,'VCC');
            SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-280,YC-150,' IC 7400 quad 2-input  NAND GATE');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC-150,'The base connection show for The 7400');
OUTTEXTXY (XC,YC+150,'  apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC-165,'These include : 74ALS00,74H000,74F00');
OUTTEXTXY (XC,YC+165,' 74HC00,74HCT00,74LS00,74S00,74AL1000 .'); END;
              {************************************}
              {*         DATA OF PIN_IC_7408        *}
              {************************************}
            PROCEDURE PIN_7408(XC,YC :INTEGER);
BEGIN        YC:= YC-30;
             BLOCK_14(XC,YC+25);
             AND_GATE(XC+115,YC);
             AND_GATE(XC+60,YC+50);
             AND_GATE(XC-54,YC);
             AND_GATE(XC-109,YC+50);
             LINE(XC+150,YC,XC+168,YC);
             LINE(XC+168,YC,XC+168,YC-34);
```

```
        LINE(XC+101,YC-7,XC+90,YC-7);
        LINE(XC+101,YC+7,XC+56,YC+7);
        LINE(XC+90,YC-7,XC+90,YC-19);
        LINE(XC+90,YC-19,XC+110,YC-19);
        LINE(XC+110,YC-19,XC+110,YC-34);
        LINE(XC+56,YC+7,XC+56,YC-34);
        LINE(XC-19,YC,XC,YC);
        LINE(XC,YC,XC,YC-34);
        LINE(XC-67,YC-7,XC-78,YC-7);
        LINE(XC-67,YC+7,XC-112,YC+7);
        LINE(XC-78,YC-7,XC-78,YC-19);
        LINE(XC-78,YC-19,XC-58,YC-19);
        LINE(XC-58,YC-19,XC-58,YC-34);
        LINE(XC-112,YC+7,XC-112,YC-34);
        LINE(XC-74,YC+50,XC-57,YC+50);
        LINE(XC-57,YC+50,XC-57,YC+85);
        LINE(XC-122,YC+43,XC-170,YC-43);
        LINE(XC-122,YC+57,XC-135,YC+57);
        LINE(XC-170,YC+43,XC-170,YC+85);
        LINE(XC-135,YC+57,XC-135,YC+70);
        LINE(XC-135,YC+70,XC-114,YC+70);
        LINE(XC-114,YC+70,XC-114,YC-85);
        LINE(XC+95,YC+50,XC+113,YC+50);
        LINE(XC+113,YC+50,XC+113,YC+85);
        LINE(XC+48,YC+43,XC,YC+43);
        LINE(XC+48,YC+57,XC+35,YC-57);
        LINE(XC,YC+43,XC,YC-85);
        LINE(XC-35,YC+57,XC+35,YC+70 );
        LINE(XC-35,YC+70,XC-56,YC-70);
        LINE(XC-56,YC+70,XC+56,YC+85);
        SETTEXTSTYLE (2,0,5);
        OUTTEXTXY (XC+159,YC-100,'GND');
        OUTTEXTXY (XC-178,YC-63,'VCC');
        SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-280,YC-150,' IC 7408 quad 2-input  AND GATE'),
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+150,'The base connection show for The 7408');
OUTTEXTXY (XC,YC+150,'  apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74ALS08,74HC08,74F08');
OUTTEXTXY (XC,YC+165,' 74HC08,74HCT08,74LS08,74S08,74AL1008 .');
      END;
        {************************************}
        {*        DATA OF PIN_IC_7432       *}
        {************************************}
       PROCEDURE PIN_7432(XC,YC :INTEGER);
        BEGIN
        YC:= YC-30;
        BLOCK_14(XC,YC+25);
```

```
            OR_GATE(XC+111,YC);
            OR_GATE(XC+58,YC+50);
            OR_GATE(XC-57,YC);
            OR_GATE(XC-112,YC+50);
            LINE(XC+143,YC,XC+168,YC);
            LINE(XC+168,YC,XC+168,YC-34);
            LINE(XC+101,YC-7,XC+90,YC-7);
            LINE(XC+101,YC+7,XC+56,YC+7);
            LINE(XC+90,YC-7,XC+90,YC-19);
            LINE(XC+90,YC-19,XC+110,YC-19);
            LINE(XC+110,YC-19,XC+110,YC-34);
            LINE(XC+56,YC+7,XC+56,YC-34);
            LINE(XC-25,YC,XC,YC);
            LINE(XC,YC,XC,YC-34);
            LINE(XC-67,YC-7,XC-78,YC-7);
            LINE(XC-67,YC+7,XC-112,YC+7);
            LINE(XC-78,YC-7,XC-78,YC-19);
            LINE(XC-78,YC-19,XC-58,YC-19);
            LINE(XC-58,YC-19,XC-58,YC-34);
            LINE(XC-112,YC+7,XC-112,YC-34);
            LINE(XC-80,YC+50,XC-57,YC+50);
            LINE(XC-57,YC+50,XC-57,YC+85);
            LINE(XC-122,YC+43,XC-170,YC+43);
            LINE(XC-122,YC+57,XC-135,YC+57);
            LINE(XC-170,YC+43,XC-170,YC+85);
            LINE(XC-135,YC+57,XC-135,YC+70);
            LINE(XC-135,YC+70,XC-114,YC+70);
            LINE(XC-114,YC+70,XC-114,YC+85);
            LINE(XC+90,YC+50,XC+113,YC+50);
            LINE(XC+113,YC+50,XC+113,YC+85);
            LINE(XC+48,YC+43,XC,YC+43);
            LINE(XC+48,YC+57,XC+36,YC+57);
            LINE(XC,YC+43,XC,YC+85);
            LINE(XC+36,YC+57,XC+36,YC+70 );
            LINE(XC+36,YC+70,XC+56,YC+70);
            LINE(XC+56,YC+70,XC+56,YC+85);
            SETTEXTSTYLE (2,0,5);
            OUTTEXTXY (XC+159,YC+100,'GND');
            OUTTEXTXY (XC-178,YC-63,'VCC');
            SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-280,YC-150,' IC 7432  quad 2-input  OR GATE');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+150,'The base connection show for The 7432');
OUTTEXTXY (XC,YC+150,'  apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74ALS32,74HC032,74F32');
OUTTEXTXY (XC,YC+165,' 74HC32,74HCT32,74LS32,74S32,74AL1032 .'); END;
            {*************************************}
            {*          DATA OF PIN_IC_7486      *}
            {*************************************}
```

```
          PROCEDURE PIN_7486(XC,YC :INTEGER);
BEGIN      YC:=YC-30;
           BLOCK_14(XC,YC+25);
           EXOR_GATE(XC+111,YC);
           EXOR_GATE(XC+58,YC+50);
           EXOR_GATE(XC-57,YC);
           EXOR_GATE(XC-112,YC+50);
           LINE(XC+143,YC,XC+168,YC);
           LINE(XC+168,YC,XC+168,YC-34);
           LINE(XC+99,YC-7,XC+90,YC-7);
           LINE(XC+99,YC+7,XC+56,YC+7);
           LINE(XC+90,YC-7,XC+90,YC-19);
           LINE(XC+90,YC-19,XC+110,YC-19);
           LINE(XC+110,YC-19,XC+110,YC-34);
           LINE(XC+56,YC+7,XC+56,YC-34);
           LINE(XC-25,YC,XC,YC);
           LINE(XC,YC,XC,YC-34);
           LINE(XC-69,YC-7,XC-78,YC-7);
           LINE(XC-69,YC+7,XC-112,YC+7);
           LINE(XC-78,YC-7,XC-78,YC-19);
           LINE(XC-78,YC-19,XC-58,YC-19);
           LINE(XC-58,YC-19,XC-58,YC-34);
           LINE(XC-112,YC+7,XC-112,YC-34);
           LINE(XC-80,YC+50,XC-57,YC+50);
           LINE(XC-57,YC+50,XC-57,YC+85);
           LINE(XC-124,YC+43,XC-170,YC+43);
           LINE(XC-124,YC+57,XC-135,YC+57);
           LINE(XC-170,YC+43,XC-170,YC+85);
           LINE(XC-135,YC+57,XC-135,YC+70);
           LINE(XC-135,YC+70,XC-114,YC+70);
           LINE(XC-114,YC+70,XC-114,YC+85);
           LINE(XC+90,YC+50,XC+113,YC+50);
           LINE(XC+113,YC+50,XC+113,YC+85);
           LINE(XC+46,YC+43,XC,YC+43);
           LINE(XC+46,YC+57,XC+35,YC-57);
           LINE(XC,YC+43,XC,YC+85);
           LINE(XC+35,YC+57,XC+35,YC+70 );
           LINE(XC+35,YC+70,XC+56,YC+70);
           LINE(XC+56,YC+70,XC+56,YC+85);
           SETTEXTSTYLE (2,0,5);
           OUTTEXTXY (XC+161,YC+100,'GND');
           OUTTEXTXY (XC-175,YC-60,'VCC');
           SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-280,YC-150,'IC 7486  quad 2-input EX-OR GATE');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 7486');
OUTTEXTXY (XC,YC+150,'. apply to all of the family groups ');
OUTTEXTXY (XC-280,YC+165,'These include : 74ALS86,74HC86,74F86');
OUTTEXTXY (XC,YC+165,' 74HC86,74HCT86,74LS86,74S86,74AL1086 .'); END;
```

```
          {*************************************}
          {*        DATA OF PIN_IC_7402        *}
          {*************************************}
        PROCEDURE PIN_7402(XC,YC :INTEGER);
BEGIN    YC:=YC-30;
         BLOCK_14(XC,YC+25);
         NOR_GATE2(XC+111,YC);
         NOR_GATE2(XC+58,YC+50);
         NOR_GATE2(XC-57,YC);
         NOR_GATE2(XC-110,YC+50);
         LINE(XC+142,YC+7,XC+168,YC+7);
         LINE(XC+168,YC+7,XC+168,YC-34);
         LINE(XC+142,YC-7,XC+152,YC-7);
         LINE(XC+89,YC,XC+56,YC );
         LINE(XC+152,YC-7,XC+152,YC-19);
         LINE(XC+152,YC-19,XC+110,YC-19);
         LINE(XC+110,YC-19,XC+110,YC-34);
         LINE(XC+56,YC,XC+56,YC-34);
         XC:=XC-168;
         LINE(XC+142,YC+7,XC+168,YC+7);
         LINE(XC+168,YC+7,XC+168,YC-34);
         LINE(XC+142,YC-7,XC+152,YC-7);
         LINE(XC+89,YC,XC+56,YC );
         LINE(XC+152,YC-7,XC+152,YC-19);
         LINE(XC+152,YC-19,XC+110,YC-19);
         LINE(XC+110,YC-19,XC+110,YC-34);
         LINE(XC+56,YC,XC+56,YC-34);
         XC:=XC+168;
         LINE(XC+89,YC+43,XC+113,YC+43);
         LINE(XC+113,YC+43,XC+113,YC+85);
         LINE(XC-36,YC+50,XC,YC+50);
         LINE(XC+89,YC+57,XC+97,YC+57);
         LINE(XC,YC-50,XC,YC+85);
         LINE(XC+98,YC+57,XC+98,YC+70 );
         LINE(XC-98,YC+70,XC+56,YC+70);
         LINE(XC+56,YC+70,XC+56,YC+85);
         XC:=XC-168;
         LINE(XC+89,YC+43,XC+113,YC+43);
         LINE(XC+113,YC+43,XC+113,YC+85);
         LINE(XC-36,YC+50,XC,YC+50);
         LINE(XC+89,YC+57,XC+97,YC+57);
         LINE(XC,YC+50,XC,YC+85);
         LINE(XC-98,YC+57,XC+98,YC+70 );
         LINE(XC+98,YC+70,XC+56,YC+70);
         LINE(XC+56,YC+70,XC+56,YC+85);
         XC:=XC+168;
         SETTEXTSTYLE (2,0,5);
         OUTTEXTXY (XC+159,YC+100,'GND');
         OUTTEXTXY (XC-178,YC-63,'VCC'); SETTEXTSTYLE (1,0,4);
```

```
OUTTEXTXY (XC-280,YC-150,'IC 7402  quad 2-input   OR GATE');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 7402');
OUTTEXTXY (XC,YC+150,'  apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74ALS02,74HO02,74F02');
OUTTEXTXY (XC,YC+165,' 74HC02,74HCT02,74LS02,74S02,74AL1002 .');
        END;
           {*************************************}
           {*        DATA OF PIN_IC_7404         *}
           {*************************************}
         PROCEDURE PIN_7404(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_14(XC,YC+25);
          NOT_GATE(XC+135,YC);
          NOT_GATE(XC+79,YC+50);
          NOT_GATE(XC-88,YC);
          NOT_GATE(XC-145,YC+50);
          NOT_GATE(XC-30,YC+50);
          NOT_GATE(XC+23,YC );
          LINE(XC+168,YC,XC+168,YC-34);   {8}
          LINE(XC+110,YC,XC+110,YC-34);   {9}
          LINE(XC+56,YC,XC+56,YC-34);     {10}
          LINE(XC+168,YC,XC+154,YC);
          LINE(XC+110,YC,XC+124,YC);
          LINE(XC+56,YC,XC+42,YC);
          XC:=XC-168;
          LINE(XC+166,YC,XC+166,YC-34);   {11}
          LINE(XC+112,YC,XC+112,YC-34);   {12}
          LINE(XC+54,YC,XC+54,YC-34);     {13}
          LINE(XC+166,YC,XC+181,YC   );
          LINE(XC+112,YC,XC+98,YC    );
          LINE(XC-54,YC,XC+70,YC    );
          XC:=XC-168;
          LINE(XC+113,YC+50,XC+113,YC+85); {6}
          LINE(XC+2,YC+50,XC+2,YC-85);     {5}
          LINE(XC+54,YC+50,XC+54,YC+85);   {4}
          LINE(XC+113,YC+50,XC-99,YC+50);
          LINE(XC+2,YC+50,XC-12,YC+50);
          LINE(XC+54,YC+50,XC+68,YC+50);
          XC:=XC-168;
          LINE(XC+113,YC+50,XC+113,YC+85); {3}
          LINE(XC-2,YC+50,XC-2,YC+85);     {2}
          LINE(XC+56,YC+50,XC+56,YC+85);   {1}
          LINE(XC+113,YC+50,XC+127,YC+50);
          LINE(XC-2,YC+50,XC+12,YC+50);
          LINE(XC+56,YC+50,XC+42,YC+50);
          XC:=XC+168;
          SETTEXTSTYLE (2,0,5);
```

```
          OUTTEXTXY (XC+159,YC+100,'GND');
          OUTTEXTXY (XC-178,YC-63,'VCC');
          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-280,YC-150,' IC 7404 HEX INVERTER (NOT GATE)');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 7404');
OUTTEXTXY (XC,YC+150,'  apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74ALS04,74HO04,74F04');
OUTTEXTXY (XC,YC+165,' 74HC04,74HCT04,74LS04,74S04,74AL1004 .');
        END;
          {*************************************}
          {*        DATA OF PIN_IC_74152       *}
          {*************************************}
         PROCEDURE PIN_74152(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_14(XC,YC+25);
          RECTANGLE(XC-147,YC-11,XC-145,YC+61);
          LINE(XC+168,YC+20,XC+168,YC-34);    {8}
          LINE(XC+168,YC+20,XC+145,YC+20);
          LINE(XC+110,YC-11,XC+110,YC-34);    {9}
          LINE(XC+56,YC-11,XC+56,YC-34);      {10}
          LINE(XC-2,YC-11,XC-2,YC-34);        {11}
          LINE(XC-56,YC-11,XC-56,YC-34);      {12}
          LINE(XC-114,YC-11,XC-114,YC-34);    {13}
          LINE(XC+113,YC+68,XC+113,YC+85);    {6}
          CIRCLE (XC+113,YC+64,5);
          LINE(XC+2,YC+61,XC+2,YC+85);        {4}
          LINE(XC+56,YC+61,XC+56,YC+85);      {5}
          LINE(XC-55,YC+61,XC-55,YC+85);      {3}
          LINE(XC-170,YC+25,XC-170,YC+85);    {1}
          LINE(XC-170,YC+25,XC-147,YC+25);
          LINE(XC-112,YC+61,XC-112,YC+85);    {2}
          SETTEXTSTYLE (2,0,5);
          OUTTEXTXY (XC+159,YC+100,'GND');
          OUTTEXTXY (XC-178,YC-63,'VCC');
          OUTTEXTXY (XC+102,YC-5 ,' B ');
          OUTTEXTXY (XC+46 ,YC-5 ,' A ');
          OUTTEXTXY (XC-7  ,YC-5 ,'D7 ');
          OUTTEXTXY (XC-61 ,YC-5 ,'D6 ');
          OUTTEXTXY (XC-119,YC-5 ,'D5 ');
          OUTTEXTXY (XC+104,YC+45,' W ');
          OUTTEXTXY (XC-5,YC+43, 'D1 ');
          OUTTEXTXY (XC+51,YC+43,'D0 ');
          OUTTEXTXY (XC-61 ,YC+43,'D2 ');
          OUTTEXTXY (XC-140,YC+20,'D4 ');
          OUTTEXTXY (XC-117,YC+43,'D3 ');
          OUTTEXTXY (XC+133,YC+15,'C  ');
```

```
          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-320,YC-150,' IC 74152  8 TO 1 LINE DATA SELECTOR');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN1-PIN5,PIN11-PIN13 :DATA INPUT   ');
OUTTEXTXY (XC,YC+120,'* PIN6 :OUTPUT  * PIN 8-PIN10 :DATA SELECT ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74152');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC152,74LS152    ');
OUTTEXTXY (XC,YC+165,'                                        ');
        END;
          {*************************************}
          {*        DATA OF PIN_IC_74164        *}
          {*************************************}
         PROCEDURE PIN_74164(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_14(XC,YC+25);
          RECTANGLE(XC-147,YC-11,XC+145,YC+61);
          LINE(XC+168,YC+20,XC+168,YC-34);   {8}
          LINE(XC+168,YC+20,XC+145,YC+20);
          LINE(XC+135,YC+20,XC+145,YC+28);
          LINE(XC+135,YC+20,XC+145,YC+12);
          LINE(XC+110,YC-11,XC+110,YC-34);   {9}
          LINE(XC+56,YC-11,XC+56,YC-34);     {10}
          LINE(XC-2,YC-11,XC-2,YC-34);       {11}
          LINE(XC-56,YC-11,XC-56,YC-34);     {12}
          LINE(XC-114,YC-11,XC-114,YC-34);   {13}
          LINE(XC+113,YC+61,XC+113,YC+85);   {6}
          LINE(XC+2,YC+61,XC+2,YC+85);       {4}
          LINE(XC+56,YC+61,XC+56,YC+85);     {5}
          LINE(XC-55,YC+61,XC-55,YC+85);     {3}
          LINE(XC-170,YC+25,XC-170,YC+85);   {1}
          LINE(XC-170,YC+25,XC-147,YC+25);
          LINE(XC-112,YC+61,XC-112,YC+85);   {2}
          SETTEXTSTYLE (2,0,5);
          OUTTEXTXY (XC+159,YC+100,'GND');
          OUTTEXTXY (XC-178,YC-63,'VCC');
          OUTTEXTXY (XC+92,YC-5 ,'CLEAR');
          OUTTEXTXY (XC+46 ,YC-5 ,' QE');
          OUTTEXTXY (XC-7  ,YC-5 ,'QF ');
          OUTTEXTXY (XC-61 ,YC-5 ,'QG ');
          OUTTEXTXY (XC-119,YC-5 ,'QH ');
          OUTTEXTXY (XC+99 ,YC+45,' QD');
          OUTTEXTXY (XC-5,YC+43, 'QB ');
          OUTTEXTXY (XC+51,YC+43,'QC ');
          OUTTEXTXY (XC-61 ,YC+43,'QA ');
          OUTTEXTXY (XC-140,YC+20,'A  ');
          OUTTEXTXY (XC-117,YC+43,'B  ');
```

```
          OUTTEXTXY (XC+113,YC+15,'CK ');
          SETTEXTSTYLE (1,0,4)
OUTTEXTXY (XC-265,YC-150,' IC 74164  8 BIT SHIFT REGISTER     ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN3-PIN6,PIN10-PIN13 :DATA OUTPUT  ');
OUTTEXTXY (XC,YC+120,'* PIN1,PIN2 :SERIAL INPUT  * PIN8 :CLOCK    ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74164');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC164,74LS164,74H164');
OUTTEXTXY (XC,YC+165,'   ,74HCT164,74F164,74ALS164              ');
        END;
          {************************************}
          {*        DATA OF PIN_IC_7474       *}
          {************************************}
         PROCEDURE PIN_7474(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_14(XC,YC+25);
          D2_FF (XC+90,YC+25);
          D1_FF (XC-80,YC+25);
          LINE(XC+168,YC+43,XC+168,YC-34);     {8}
          LINE(XC+168,YC+43,XC+131,YC+43);
          LINE(XC+110,YC-18,XC+110,YC-34);     {9}
          LINE(XC+110,YC-18,XC+145,YC-18);
          LINE(XC+145,YC-18,XC+145,YC+9 );
          LINE(XC+145,YC+9,XC+130,YC+9 );
          LINE(XC-56,YC-26 ,XC+56,YC-34);      {10}
          LINE(XC+56,YC-26 ,XC+90,YC-26);
          LINE(XC+90,YC-26 ,XC+90,YC-17);
          LINE(XC-2,YC+35,XC-2,YC-34);         {11}
          LINE(XC-2,YC+35,XC+50,YC-35);
          LINE(XC-56,YC-18,XC-56,YC-34);       {12}
          LINE(XC-56,YC-18,XC+38 ,YC-18);
          LINE(XC+38,YC-18,XC+33 ,YC+7 );
          LINE(XC+38,YC+7,XC+50,YC+7 );
          LINE(XC-114,YC-28,XC-114,YC-34);     {13}
          LINE(XC-114,YC-28,XC+25,YC-28);
          LINE(XC+25,YC-28,XC+25,YC+70);
          LINE(XC+25,YC+70,XC+90,YC+70);
          LINE(XC+90,YC+70,XC+90,YC+68);
          LINE(XC+113,YC+75,XC+113,YC+85);     {6}
          LINE(XC+113,YC+75,XC+17 ,YC+75);
          LINE(XC+17 ,YC+75,XC+17 ,YC+10 );
          LINE(XC-17 ,YC+10 ,XC-40 ,YC+10);
          LINE(XC+2,YC+76,XC+2,YC+85);         {4}
          LINE(XC+2,YC+76,XC-80,YC+76);
          LINE(XC-80,YC+76,XC-80,YC+68);
          LINE(XC+56,YC+80,XC+56,YC+85);       {5}
          LINE(XC+56,YC+80,XC+8 ,YC+80);
```

```
          LINE(XC+8 ,YC+80,XC+8 ,YC+45);
          LINE(XC+8 ,YC+45,XC-40,YC+45);
          LINE(XC-55,YC+80,XC-55,YC+85);        {3}
          LINE(XC-55,YC+80,XC-152,YC+80);
          LINE(XC-152,YC+80,XC-152,YC+11);
          LINE(XC-152,YC+11,XC-120,YC+11);
          LINE(XC-170,YC-23,XC-170,YC+85);      {1}
          LINE(XC-170,YC-23,XC-80,YC-23);
          LINE(XC-80,YC-23,XC-80,YC-18);
          LINE(XC-112,YC+68,XC-112,YC+85);      {2}
          LINE(XC-112,YC+68,XC-140,YC+68);
          LINE(XC-140,YC+68,XC-140,YC+39);
          LINE(XC-140,YC+39,XC-120,YC+39);
          SETTEXTSTYLE (2,0,5);
       OUTTEXTXY (XC+159,YC+100,'GND');
          OUTTEXTXY (XC-178,YC-63,'VCC');

          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-265,YC-150,' IC 7474 DUAL D TYPE FLIP FLOP      ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN..1:CLR1, 2:D1, 3:CLK1, 4:PR1, ');
OUTTEXTXY (XC,YC+120,'5:Q1, 6:NOT Q1, 8:NOT Q2, 9:Q2, 10:PR2, ');
OUTTEXTXY (XC-280,YC+135,'11:CLK2, 12:D2, 13:CLR2 *          ')
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 7474 ');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC74 ,74LS74 ,74H74 ');
OUTTEXTXY (XC,YC+165,'   ,74HCT74 ,74F74 ,74ALS74              ');
         END;
          {*************************************}
          {*        DATA OF PIN_IC_74393       *}
          {*************************************}
        PROCEDURE PIN_74393(XC,YC :INTEGER);
         BEGIN
         YC:=YC-30;
         BLOCK_14(XC,YC+25);
         RECTANGLE(XC-30 ,YC-11,XC+190,YC+10);
         RECTANGLE(XC-80,YC+40,XC+140,YC+61);
         CIRCLE(XC+85 ,YC+14,5);
         CIRCLE(XC+30,YC+36,5);
         LINE(XC+168,YC-11,XC+168,YC-34);   {8}
         LINE(XC+110,YC-11,XC+110,YC-34);   {9}
         LINE(XC+56,YC-11,XC+56,YC-34);     {10}
         LINE(XC-2,YC-11,XC-2,YC-34);       {11}
         LINE(XC-56,YC-4 ,XC-56,YC-34);     {12}
         LINE(XC-56,YC-4 ,XC-30,YC-4);     {12}
         LINE(XC-114,YC+21 ,XC-114,YC-34);   {13}
         LINE(XC-114,YC+21,XC+85 ,YC+21);   {13}
         LINE(XC+85 ,YC+21,XC+85 ,YC+18);   {13}
```

```
          LINE(XC+113,YC+61,XC+113,YC+85);    {6}
          LINE(XC+2,YC+61,XC+2,YC+85);        {4}
          LINE(XC+56,YC+61,XC+56,YC+85);      {5}
          LINE(XC-55,YC+61,XC-55,YC+85);      {3}
          LINE(XC-170,YC+29,XC-170,YC+85);    {1}
          LINE(XC-170,YC+29,XC+30 ,YC+29);
          LINE(XC+30,YC+29,XC+30 ,YC+33);
          LINE(XC-112,YC+55,XC-112,YC+85);    {2}
          LINE(XC-112,YC+55,XC-80,YC+55);    {2}
          SETTEXTSTYLE (2,0,5);
          OUTTEXTXY (XC+160,YC+100,'GND');
          OUTTEXTXY (XC-171,YC-60,'VCC');
          OUTTEXTXY (XC-131,YC+100,'CLEAR');
          OUTTEXTXY (XC-73,YC-60,'CLEAR');
        OUTTEXTXY (XC+92,YC-8 ,' QC ');
          OUTTEXTXY (XC+46 ,YC-8 ,'QB');
          OUTTEXTXY (XC-7  ,YC-8 ,'QA ');
          OUTTEXTXY (XC+163,YC-8 ,'QD');
          OUTTEXTXY (XC+105 ,YC+46,'QD');
          OUTTEXTXY (XC-5,YC+46, 'QB ');
          OUTTEXTXY (XC+51,YC+46,'QC ');
          OUTTEXTXY (XC-61 ,YC+46,'QA ');
          SETTEXTSTYLE (2,0,4);
          OUTTEXTXY (XC+76 ,YC-2  ,' A ');
          OUTTEXTXY (XC+23 ,YC+42,' A ');

          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-330,YC-150,' IC 74393  DUAL 4 BIT BINARY COUNTER');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN3-PIN6,PIN8-PIN11 :DATA OUTPUT  ');
OUTTEXTXY (XC,YC+120,'* PIN1,PIN13 :CONTROL  * PIN2,12:CLEAR  ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74393');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC393,74LS393,74H393');
OUTTEXTXY (XC,YC+165,'   ,74HCT393,74F393,74ALS393             '); END;
          {**************************************}
          {*        DATA OF PIN_IC_74390        *}
          {**************************************}
        PROCEDURE PIN_74390(XC,YC :INTEGER);
         BEGIN
         YC:=YC-30;
         BLOCK_16(XC,YC+25);
         XC:=XC-27;
         RECTANGLE(XC-30 ,YC-11,XC+245,YC+10);
         CIRCLE(XC+85 ,YC+14,5);
         LINE(XC+223,YC-11,XC+223,YC-34);    {9}
         LINE(XC+164,YC-11,XC+164,YC-34);    {10}
         LINE(XC+107,YC-11,XC+107,YC-34);    {11}
         LINE(XC+54,YC-18,XC+54,YC-34);      {12}
```

```
            CIRCLE(XC+54,YC-15,5);
            LINE(XC-4,YC-11,XC-4,YC-34);        {13}
            LINE(XC-57,YC-4 ,XC-57,YC-34);      {14}
            LINE(XC-57,YC-4 ,XC-30,YC-4);
            LINE(XC-117,YC+21 ,XC-117,YC-34);   {15}
            LINE(XC-117,YC+21,XC+85 ,YC+21);
            LINE(XC+85 ,YC+21,XC+85 ,YC+18);
            XC:=XC+27;
            XC:=XC-30;
            RECTANGLE(XC-80,YC+40,XC+195,YC+61);
            CIRCLE(XC+30,YC+36,5);
            LINE(XC+168,YC+61,XC+168,YC+85);    {7}
            LINE(XC+113,YC+61,XC+113,YC+85);    {6}
            LINE(XC+2,YC+68,XC+2,YC+85);        {4}
            CIRCLE(XC+2,YC+65,5);
            LINE(XC+56,YC+61,XC+56,YC+85);      {5}
            LINE(XC-55,YC+61,XC-55,YC+85);      {3}
            LINE(XC-170,YC+29,XC-170,YC+85);    {1}
            LINE(XC-170,YC+29,XC+30 ,YC+29);
            LINE(XC+30,YC+29,XC+30 ,YC+33);
            LINE(XC-112,YC+55,XC-112,YC+85);    {2}
            LINE(XC-112,YC+55,XC-80,YC+55);
            XC:=XC+30;
            SETTEXTSTYLE (2,0,5);
            XC:=XC-25;
            OUTTEXTXY (XC+213,YC+100,'GND');
            OUTTEXTXY (XC-186,YC-60,'VCC');
            OUTTEXTXY (XC-134,YC+100,'CLEAR');
            OUTTEXTXY (XC-77,YC-60,'CLEAR');
            OUTTEXTXY (XC+92,YC-8 ,' QB ');
            OUTTEXTXY (XC+49 ,YC-8 ,'B ');
            OUTTEXTXY (XC-7  ,YC-8 ,'QA ');
            XC:=XC+25; XC:=XC-30;
            OUTTEXTXY (XC+163,YC-8 ,'QC');
            OUTTEXTXY (XC+105 ,YC-46,'QC');
            OUTTEXTXY (XC+220,YC-8 ,'QD');
            OUTTEXTXY (XC+162 ,YC+46,'QD');
            OUTTEXTXY (XC-2,YC+46, 'B ');
            OUTTEXTXY (XC+51,YC+46,'QB ');
            OUTTEXTXY (XC-61 ,YC+46,'QA ');
            SETTEXTSTYLE (2,0,4);
            OUTTEXTXY (XC+23 ,YC+42,' A ');
            OUTTEXTXY (XC+77 ,YC-2  ,' A '); XC:=XC+30;
            SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-280,YC-150,' IC 74390  DUAL DECADE COUNTER      ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN3,PIN5-PIN7,PIN13,PIN9-PIN11 :DA');
OUTTEXTXY (XC-3,YC+120,'TA OUTPUT  *PIN1,PIN4,PIN12,PIN15 :CONTROL');
OUTTEXTXY (XC-280,YC+135,'* PIN2,PIN14 :CLEAR                  ');
```

```
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74390');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC390,74LS390,74H390');
OUTTEXTXY (XC,YC+165,'   ,74HCT390,74F390,74ALS390              ');
        END;
          {************************************}
          {*        DATA OF PIN_IC_74112        *}
          {************************************}
         PROCEDURE PIN_74112(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_16(XC,YC+25);
          XC:=XC+28;
          JK1_FF (XC+90,YC+25);
          LINE(XC+168,YC+9 ,XC+168,YC-34);      {9}
          LINE(XC+168,YC+43,XC+131,YC+43);
         LINE(XC+110,YC-26,XC+110,YC-34);      {10}
          LINE(XC+131,YC+9,XC+168,YC+9 );
          LINE(XC+56,YC-26 ,XC+56,YC-34);       {11}
          LINE(XC+110,YC-26 ,XC+90,YC-26);
          LINE(XC+56,YC-26 ,XC+39,YC-26);
          LINE(XC+39,YC-26 ,XC+39,YC+9);
          LINE(XC+39,YC+9 ,XC+49,YC+9);
          LINE(XC+90,YC-26 ,XC+90,YC-17);
          LINE(XC-2,YC+35,XC-2,YC-34);          {12}
          LINE(XC-2,YC+35,XC-50,YC+35);
          LINE(XC-56,YC-18,XC-56,YC-34);        {13}
          LINE(XC-56,YC-18,XC-28 ,YC-18);
          LINE(XC-28,YC-18,XC-28 ,YC+25);
          LINE(XC-28,YC+25,XC+39,YC+25);
          LINE(XC-114,YC-28,XC-114,YC-34);      {14}
          LINE(XC-114,YC-28,XC+25,YC-28);
          LINE(XC+25,YC-28,XC+25,YC+70);
          LINE(XC+25,YC+70,XC+90,YC+70);
          LINE(XC+90,YC+70,XC+90,YC+68);
          XC:=XC-28;
          JK2_FF (XC-107,YC+25);
          XC:=XC-27;
          LINE(XC+166,YC+75,XC+166,YC+85);      {7}
          LINE(XC+166,YC+75,XC+223,YC+75);      {7}
          LINE(XC+223,YC+43,XC+223,YC+75);      {7}
          LINE(XC+113,YC+75,XC+113,YC+85);      {6}
          LINE(XC+113,YC+75,XC+17 ,YC+75);
          LINE(XC+17 ,YC+75,XC+17 ,YC+10 );
          LINE(XC+17 ,YC+10 ,XC-40 ,YC+10);
          LINE(XC+2,YC+76,XC+2,YC+85);          {4}
          LINE(XC+2,YC+76,XC-80,YC+76);
          LINE(XC-80,YC+76,XC-80,YC+68);
          LINE(XC+56,YC+80,XC+56,YC+85);        {5}
```

```
          LINE(XC+56,YC+80,XC+8 ,YC+80);
          LINE(XC+8 ,YC+80,XC+8 ,YC+45);
          LINE(XC+8 ,YC+45,XC-40,YC+45);
          LINE(XC-55,YC+80,XC-55,YC+85);        {3}
          LINE(XC-55,YC+80,XC-140,YC+80);
          LINE(XC-140,YC+80,XC-140,YC+65);
          LINE(XC-152,YC+68,XC-152,YC+11);
          LINE(XC-152,YC+68,XC-140,YC+68);
          LINE(XC-152,YC+11,XC-120,YC+11);
          LINE(XC-170,YC+25,XC-170,YC+85);      {1}
          LINE(XC-170,YC+25,XC-130,YC+25);      {1}
          LINE(XC-114,YC-23,XC-80,YC-23);
          LINE(XC-114,YC-23,XC-114,YC-34);
          LINE(XC-80,YC-23,XC-80,YC-18);
          LINE(XC-112,YC+68,XC-112,YC+85);      {2}
          LINE(XC-112,YC+68,XC-140,YC+68);
          LIN(XC-140,YC+68,XC-140,YC+39);
          LINE(XC-140,YC+39,XC-120,YC+39);
          XC:=XC+27;
          SETTEXTSTYLE (2,0,5);
          OUTTEXTXY (XC+184,YC+100,'GND');
          OUTTEXTXY (XC-210,YC-63,'VCC');

          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-300,YC-150,'   IC 74112   DUAL  JK  FLIP FLOP   ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN..1:CLK1, 2:K1, 3:J1 , 4:PR1, ');
OUTTEXTXY (XC-15,YC+120,'5:Q1, 6:NOT Q1, 7:NOT Q2, 9:Q2, 10:PR2, ');
OUTTEXTXY (XC-280,YC+135,'11:J2 , 12:K2, 13:CLK2, 14:CLR2, 15:CLR1');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74112 ');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC112,74LS112,74S112');
OUTTEXTXY (XC,YC+165,'   ,74HCT112,74F112,74ALS112              ');
        END,
          {*************************************}
          {*        DATA OF PIN_IC_74283         *}
          {*************************************}
         PROCEDURE PIN_74283(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_16(XC,YC+25);
          XC:=XC-27;
          LINE(XC+223,YC+25,XC+223,YC-34);   {9}
          LINE(XC+223,YC+25,XC+192,YC+25);   {9}

          LINE(XC+164,YC-11,XC+164,YC-34);   {10}
          LINE(XC+107,YC-11,XC+107,YC-34);   {11}
          LINE(XC+54,YC-11,XC+54,YC-34);     {12}
          LINE(XC-4,YC-11,XC-4,YC-34);       {13}
```

```
            LINE(XC-57,YC-11,XC-57,YC-34);     {14}
            LINE(XC-117,YC-11 ,XC-117,YC-34);  {15}
            XC:=XC+27;
            XC:=XC-30;
            RECTANGLE(XC-140,YC-10 ,XC+195,YC+61);
            LINE(XC+168,YC+61,XC+168,YC+85);   {7}
            LINE(XC+113,YC+61,XC+113,YC+85);   {6}
            LINE(XC+2,YC+61,XC+2,YC+85);       {4}
            LINE(XC+56,YC+61,XC+56,YC+85);     {5}
            LINE(XC-55,YC+61,XC-55,YC+85);     {3}
            LINE(XC-170,YC+25,XC-170,YC+85);   {1}
            LINE(XC-170,YC+25,XC-140,YC+25);   {1}
            LINE(XC-112,YC+61,XC-112,YC+85);   {2}
            XC:=XC+30;
            SETTEXTSTYLE (2,0,5);
            XC:=XC-25;
            OUTTEXTXY (XC+213,YC+100,'GND')
            OUTTEXTXY (XC-186,YC-60,'VCC');
            OUTTEXTXY (XC+158,YC+20,' C4 ');  {9}
            OUTTEXTXY (XC+92,YC-8 ,' B4 ');   {11}
            OUTTEXTXY (XC+49 ,YC-8 ,'A4');    {12}
            OUTTEXTXY (XC+157 ,YC-8 ,'Y4 ');  {10}
            OUTTEXTXY (XC-71,YC-8 ,' A3 ');   {14}
            OUTTEXTXY (XC-123,YC-8 ,'B3');    {15}
            OUTTEXTXY (XC-11 ,YC-8 ,'Y3 ');   {13}
            XC:=XC+25;
            XC:=XC-30;
            OUTTEXTXY (XC+108 ,YC+46,'B1');   {6}
            OUTTEXTXY (XC+162 ,YC+46,'C0');   {7}
            OUTTEXTXY (XC-6,YC+46, 'Y1');     {4}
            OUTTEXTXY (XC+51,YC+46,'A1 ');    {5}
            OUTTEXTXY (XC-61 ,YC+46,'A2');    {3}
            OUTTEXTXY (XC-117 ,YC+46,'B2 ');  {2}
            OUTTEXTXY (XC-130,YC+20,'Y2 ');   {1}
            XC:=XC+30;
            SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-300,YC-150,' IC 74283  4 BIT BINARY FULL ADDER  ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN1,PIN4,PIN7,PIN9,PIN10,PIN13 :DA');
OUTTEXTXY (XC-3,YC+120,'TA OUTPUT  *PIN2,PIN3,PIN5,PIN6,PIN11');
OUTTEXTXY (XC-280,YC+135,',PIN12,PIN14,PIN15 :DATA INPUT       ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74283');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC283,74LS283,74H283');
OUTTEXTXY (XC,YC+165,',74HCT283,74F283,74ALS283             ');
        END;
          {**************************************}
          {*        DATA OF PIN_IC_7485         *}
          {**************************************}
```

```
        PROCEDURE PIN_7485(XC,YC :INTEGER);
BEGIN    YC:=YC-30;
         BLOCK_16(XC,YC+25); XC:=XC-27;
         LINE(XC+223,YC+25,XC+223,YC-34);    {9}
         LINE(XC+223,YC+25,XC+192,YC+25);    {9}
         LINE(XC+164,YC-11,XC+164,YC-34);    {10}
         LINE(XC+107,YC-11,XC+107,YC-34);    {11}
         LINE(XC+54,YC-11,XC+54,YC-34);      {12}
         LINE(XC-4,YC-11,XC-4,YC-34);        {13}
         LINE(XC-57,YC-11,XC-57,YC-34);      {14}
         LINE(XC-117,YC-11 ,XC-117,YC-34);   {15}
         XC:=XC+27;
         XC:=XC-30;
         RECTANGLE(XC-140,YC-10 ,XC+195,YC+61);
         LINE(XC+168,YC+61,XC+168,YC+85);    {7}
         LINE(XC+113,YC+61,XC+113,YC+85);    {6}
         LINE(XC+2,YC+61,XC+2,YC+85);        {4}
         LINE(XC+56,YC+61,XC+56,YC+85);      {5}
         LINE(XC-55,YC+61,XC-55,YC+85);      {3}
         LINE(XC-170,YC+25,XC-170,YC+85);    {1}
         LINE(XC-170,YC+25,XC-140,YC+25);    {1}
         LINE(XC-112,YC+61,XC-112,YC+85);    {2}
         XC:=XC+30;
         SETTEXTSTYLE (2,0,5);
         XC:=XC-25;
         OUTTEXTXY (XC+210,YC+100,'GND');
         OUTTEXTXY (XC-186,YC-60,'VCC');
         OUTTEXTXY (XC+158,YC+20,' B0 ');    {9}
         OUTTEXTXY (XC+92,YC-8 ,' B1 ');     {11}
         OUTTEXTXY (XC+49 ,YC-8 ,'A1');      {12}
         OUTTEXTXY (XC+157 ,YC-8 ,'A0 ');    {10}
         OUTTEXTXY (XC-71,YC-8 ,' B2 ');     {14}
         OUTTEXTXY (XC-123,YC-8 ,'A3');      {15}
         OUTTEXTXY (XC-11 ,YC-8 ,'A2 ');     {13}
         XC:=XC+25;
         XC:=XC-30;
         OUTTEXTXY (XC+205 ,YC+46,'A=B');    {6}
         OUTTEXTXY (XC+101 ,YC+100,'OUT');   {6}
         OUTTEXTXY (XC+159 ,YC+46,'A<B ');   {7}
         OUTTEXTXY (XC+158 ,YC+100,'OUT ');  {7}
         OUTTEXTXY (XC-6,YC+46, 'A>B');      {4}
         OUTTEXTXY (XC-6,YC+100, 'IN ');     {4}
         OUTTEXTXY (XC+49,YC+46,'A>B ');     {5}
         OUTTEXTXY (XC+47,YC+100,'OUT ');    {5}
         OUTTEXTXY (XC-65 ,YC+46,'A=B');     {3}
         OUTTEXTXY (XC-61 ,YC+100,'IN ');    {3}
         OUTTEXTXY (XC-120 ,YC+46,'A<B');    {2}
         OUTTEXTXY (XC-117 ,YC+100,'IN ');   {2}
         OUTTEXTXY (XC-130,YC+20,'B3 ');     {1} XC:=XC+30;
```

```
          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-300,YC-150,'     IC 7485  4 BIT COMPARATOR    ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN2-PIN4 :CASCADE INPUT *PIN5-PIN7');
OUTTEXTXY (XC+5,YC+120,' :OUTPUT  *PIN1,PIN9-PIN15 :DATA OUT');
OUTTEXTXY (XC-280,YC+135,'PUT                                    ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 7485 ');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC85 ,74LS85 ,74H85 ');
OUTTEXTXY (XC,YC+165,'   ,74HCT85 ,74F85 ,74ALS85              ');
        END;
          {************************************}
         {*        DATA OF PIN_IC_74165        *}
          {************************************}
         PROCEDURE PIN_74165(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_16(XC,YC+25);
          XC:=XC-27;
          LINE(XC+223,YC+25,XC+223,YC-34);   {9}
          LINE(XC+223,YC+25,XC+192,YC+25);   {9}
          LINE(XC+164,YC-11,XC+164,YC-34);   {10}
          LINE(XC+107,YC-11,XC+107,YC-34);   {11}
          LINE(XC+54,YC-11,XC+54,YC-34);     {12}
          LINE(XC-4,YC-11,XC-4,YC-34);       {13}
          LINE(XC-57,YC-11,XC-57,YC-34);     {14}
          LINE(XC-117,YC-18 ,XC-117,YC-34);  {15}
          CIRCLE (XC-117,YC-14,5);
          XC:=XC+27;
          XC:=XC-30;
          RECTANGLE(XC-140,YC-10 ,XC+195,YC+61);
          LINE(XC+168,YC+61,XC+168,YC+85);   {7}
          LINE(XC+113,YC+61,XC+113,YC+85);   {6}
          LINE(XC+2,YC+61,XC+2,YC+85);       {4}
          LINE(XC+56,YC+61,XC+56,YC+85);     {5}
          LINE(XC-55,YC+61,XC-55,YC+85);     {3}
          LINE(XC-170,YC+25,XC-170,YC+85);   {1}
          LINE(XC-170,YC+25,XC-140,YC+25);   {1}
          LINE(XC-112,YC+61,XC-112,YC+85);   {2}
          LINE(XC-112,YC+55,XC-118,YC+61);   {2}
          LINE(XC-112,YC+55,XC-106,YC+61);   {2}
          XC:=XC+30;
          SETTEXTSTYLE (2,0,5);
          XC:=XC-25;
          OUTTEXTXY (XC+210,YC+100,'GND');
          OUTTEXTXY (XC-186,YC-60,'VCC');
          OUTTEXTXY (XC+161,YC+20,' QH');    {9}
          OUTTEXTXY (XC+92,YC-8 ,' A  ');    {11}
          OUTTEXTXY (XC+49 ,YC-8 ,'B ');     {12}
```

```
          OUTTEXTXY (XC+142 ,YC-8 ,'SERIAL');{10}
          OUTTEXTXY (XC+156,YC+5, 'IN ');    {4}
          OUTTEXTXY (XC-69,YC-8 ,' D  ');    {14}
          OUTTEXTXY (XC-133,YC-8 ,'CLOCK');  {15}
          OUTTEXTXY (XC-9  ,YC-8 ,'C  ');    {13}
          XC:=XC+25;
          XC:=XC-30;
          OUTTEXTXY (XC+109 ,YC+46,'H  ');   {6}
          OUTTEXTXY (XC+159 ,YC+46,'QH  ');  {7}
          OUTTEXTXY (XC+159 ,YC+40,'-  ');   {7}
          OUTTEXTXY (XC   ,YC+46, 'F  ');    {4}
          OUTTEXTXY (XC+54,YC+46,'G   ');    {5}
          OUTTEXTXY (XC-56 ,YC+46,'E  ');    {3}
          OUTTEXTXY (XC-118 ,YC+43,'CK ');   {2}
          OUTTEXTXY (XC-130,YC+20,'SHIFT/LOAD');
          XC:=XC+30;

          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-300,YC-150,'IC 74165  8 BIT SHIFT REGISTER  PISO');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN3-PIN6,PIN11-PIN14 :PARALLEL INP');
OUTTEXTXY (XC-1,YC+120,'UT, * PIN7,PIN9 :OUTPUT, PIN10 :SERIAL');
OUTTEXTXY (XC-280,YC+135,' INPUT, * PIN15 :CLOCK INHIBIT     ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74165');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC165,74LS165,74H165');
OUTTEXTXY (XC,YC+165,'    ,74HCT165,74F165,74ALS165              ');
        END;
          {************************************}
          {*        DATA OF PIN_IC_74195      *}
          {************************************}

        PROCEDURE PIN_74195(XC,YC :INTEGER);
         BEGIN
         YC:=YC-30;
         BLOCK_16(XC,YC+25);
         XC:=XC-27;
         LINE(XC+223,YC+25,XC+223,YC-34);   {9}
         LINE(XC+223,YC+25,XC+192,YC+25);   {9}
         LINE(XC+164,YC-11,XC+164,YC-34);   {10}
         LINE(XC+164,YC-2,XC+172,YC-10);    {10}
         LINE(XC+164,YC-2,XC+156,YC-10);    {10}
         LINE(XC+107,YC-11,XC+107,YC-34);   {11}
         LINE(XC+54,YC-11,XC+54,YC-34);     {12}
         LINE(XC-4,YC-11,XC-4,YC-34);       {13}
         LINE(XC-57,YC-11,XC-57,YC-34);     {14}
         LINE(XC-117,YC-11 ,XC-117,YC-34);  {15}
         XC:=XC+27;
```

```
        XC:=XC-30;
        RECTANGLE(XC-140,YC-10 ,XC+195,YC+61);
        LINE(XC+168,YC+61,XC+168,YC+85);    {7}
        LINE(XC+113,YC+61,XC+113,YC+85);    {6}
        LINE(XC+2,YC+61,XC+2,YC+85);        {4}
        LINE(XC+56,YC+61,XC+56,YC+85);      {5}
        LINE(XC-55,YC+61,XC-55,YC+85);      {3}
        LINE(XC-170,YC+25,XC-170,YC+85);    {1}
        LINE(XC-170,YC+25,XC-151,YC+25);    {1}
        CIRCLE (XC-146,YC+25,5);
        LINE(XC-112,YC+61,XC-112,YC+85);    {2}
        XC:=XC+30;
        SETTEXTSTYLE (2,0,5);
        XC:=XC-25;
        OUTTEXTXY (XC+210,YC+100,'GND');
        OUTTEXTXY (XC-186,YC-60,'VCC');
       OUTTEXTXY (XC+110,YC+18,'SHIFT/LOAD ');    {9}
        OUTTEXTXY (XC+92,YC-8 ,' QD ');     {11}
        OUTTEXTXY (XC+92,YC-14,' -  ');     {11}
        OUTTEXTXY (XC+52 ,YC-8 ,'QD');      {12}
        OUTTEXTXY (XC+156 ,YC+1 ,'CK ');{10}
        OUTTEXTXY (XC-73,YC-8 ,' QB ');     {14}
        OUTTEXTXY (XC-125,YC-8 ,'QA ');  {15}
        OUTTEXTXY (XC-13 ,YC-8 ,'QC ');     {13}
        XC:=XC+25;
        XC:=XC-30;
        OUTTEXTXY (XC+109 ,YC+46,'C  ');    {6}
        OUTTEXTXY (XC+159 ,YC+46,'D  ');    {7}
        OUTTEXTXY (XC-56 ,YC+40,'-  ');    {7}
        OUTTEXTXY (XC  ,YC+46, 'A  ');      {4}
        OUTTEXTXY (XC+54,YC+46,'B  ');      {5}
        OUTTEXTXY (XC-56 ,YC+46,'K  ');     {3}
        OUTTEXTXY (XC-116 ,YC+46,'J  ');    {2}
        OUTTEXTXY (XC-130,YC+20,'CLEAR');
        XC:=XC+30;
        SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-300,YC-150,'IC 74195  8 BIT SHIFT REGISTER  PIPO');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN4-PIN7 :PARALLEL INPUT, * PIN2,');
OUTTEXTXY (XC-1,YC+120,'PIN3 :SERIAL INPUT, * PIN11-PIN15 :OUT');
OUTTEXTXY (XC-280,YC+135,'PUT, * PIN1 :CLEAR, * PIN10 :CLOCK ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74195');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC195,74LS195,74H195');
OUTTEXTXY (XC,YC+165,'   ,74HCT195,74F195,74ALS195              ');
      END;
        {*************************************}
        {*        DATA OF PIN_IC_74169       *}
        {*************************************}
```

```
          PROCEDURE PIN_74169(XC,YC :INTEGER);
BEGIN      YC:=YC-30;
           BLOCK_16(XC,YC+25); XC:=XC-27;
           LINE(XC+223,YC+25,XC+223,YC-34);    {9}
           LINE(XC+223,YC+25,XC+204,YC+25);    {9}
           CIRCLE (XC+198,YC+25,5);
           LINE(XC+164,YC-18,XC+164,YC-34);    {10}
           CIRCLE (XC+164,YC-14,5);
           LINE(XC+107,YC-11,XC+107,YC-34);    {11}
           LINE(XC+54,YC-11,XC+54,YC-34);      {12}
           LINE(XC-4,YC-11,XC-4,YC-34);        {13}
           LINE(XC-57,YC-11,XC-57,YC-34);      {14}
           LINE(XC-117,YC-18 ,XC-117,YC-34);   {15}
           CIRCLE (XC-117,YC-14,5); XC:=XC+27; XC:=XC-30;
           RECTANGLE(XC-14YC-10 ,XC+195,YC+61);
           LINE(XC+168,YC+69,XC+168,YC+85);    {7}
           CIRCLE (XC+168,YC+65,5);
           LINE(XC+113,YC+61,XC+113,YC+85);    {6}
           LINE(XC+2,YC+61,XC+2,YC+85);        {4}
           LINE(XC+56,YC+61,XC+56,YC+85);      {5}
           LINE(XC-55,YC+61,XC-55,YC+85);      {3}
           LINE(XC-170,YC+25,XC-170,YC+85);    {1}
           LINE(XC-170,YC+25,XC-140,YC+25);    {1}
           LINE(XC-112,YC+61,XC-112,YC+85);    {2}
           LINE(XC-112,YC+55,XC-118,YC+61);    {2}
           LINE(XC-112,YC+55,XC-106,YC+61);    {2}
           XC:=XC+30;
           SETTEXTSTYLE (2,0,5);
           XC:=XC-25;
           OUTTEXTXY (XC+210,YC+100,'GND');
           OUTTEXTXY (XC-186,YC-60,'VCC');
           OUTTEXTXY (XC+148,YC+20,'LOAD');     {9}
           OUTTEXTXY (XC+92,YC-8 ,' QD ');      {11}
           OUTTEXTXY (XC+49 ,YC-8 ,'QC');       {12}
           OUTTEXTXY (XC+142 ,YC-8 ,'ENABLE');{10}
           OUTTEXTXY (XC+156,YC+5, 'T  ');      {4}
           OUTTEXTXY (XC+156,YC-1, '-  ');      {4}
           OUTTEXTXY (XC-69,YC-8 ,' QA ');      {14}
           OUTTEXTXY (XC-133,YC-8 ,'RIPPLE');   {15}
           OUTTEXTXY (XC-137,YC  ,'carry o/p'); {15}
           OUTTEXTXY (XC-9  ,YC-8 ,'QB ');      {13}
           XC:=XC+25;
           XC:=XC-30;
           OUTTEXTXY (XC+109 ,YC+46,'D  ');     {6}
           OUTTEXTXY (XC+142 ,YC+36,'ENABLE');  {7}
           OUTTEXTXY (XC+159 ,YC+49,'P');       {7}
           OUTTEXTXY (XC+159 ,YC+43,'-');       {7}
           OUTTEXTXY (XC  ,YC+46, 'B  ');       {4}
           OUTTEXTXY (XC+54,YC+46,'C   ');      {5}
```

```
          OUTTEXTXY (XC-56 ,YC+46,'A  ');     {3}
          OUTTEXTXY (XC-118 ,YC+43,'CK ');    {2}
          OUTTEXTXY (XC-130,YC+20,'UP/DOWN');
          XC:=XC+30;
          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-270,YC-150,'IC 74169  SYNCHRONOUS UP DOWN        ');
OUTTEXTXY (XC-270,YC-115,'          BIRARY COUNTER             ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN3-PIN6 :DATA INPUTS, * PIN11-PIN');
OUTTEXTXY (XC-1,YC+120,'15 :OUTPUTS, * PIN1 :UP/DOWN, * PIN2,');
OUTTEXTXY (XC-280,YC+135,'PIN10 :ENABLE, * PIN15 :CARRY OUTPUT');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74169;
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC169,74LS169,74H169');
OUTTEXTXY (XC,YC+165,'   ,74HCT169,74F169,74ALS169              ');
        END;
          {************************************}
          {*       DATA OF PIN_IC_74131       *}
          {************************************}
         PROCEDURE PIN_74131(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_16(XC,YC+25);
          XC:=XC-27;
          LINE(XC+223,YC+25,XC+223,YC-34);    {9}
          LINE(XC+223,YC+25,XC-204,YC+25);    {9}
          CIRCLE (XC-198,YC+25,5);
          LINE(XC+164,YC-18,XC+164,YC-34);    {10}
          CIRCLE (XC+164,YC-14,5);
          LINE(XC+107,YC-18,XC+107,YC-34);    {11}
          CIRCLE (XC+107,YC-14,5);
          LINE(XC+54,YC-18,XC+54,YC-34);      {12}
          CIRCLE (XC+54 ,YC-14,5);
          LINE(XC-4,YC-18,XC-4,YC-34);        {13}
          CIRCLE (XC-4  ,YC-14,5);
          LINE(XC-57,YC-18,XC-57,YC-34);      {14}
          CIRCLE (XC-57 ,YC-14,5);
          LINE(XC-117,YC-18 ,XC-117,YC-34);   {15}
          CIRCLE (XC-117,YC-14,5);
          XC:=XC-27;
          XC:=XC-30;
          RECTANGLE(XC-140,YC-10 ,XC+195,YC+61);
          LINE(XC+168,YC+69,XC+168,YC+85);    {7}
          CIRCLE (XC+168,YC+65,5);
          LINE(XC+113,YC+61,XC+113,YC+85);    {6}
          LINE(XC+2,YC+61,XC+2,YC+85);        {4}
          LINE(XC+2  ,YC+54,XC-6  ,YC+61);    {2}
          LINE(XC+2  ,YC+54,XC+10 ,YC+61);    {2}
```

```
            LINE(XC+56,YC+68,XC+56,YC+85);      {5}
            CIRCLE (XC+56 ,YC+65,5);
            LINE(XC-55,YC+61,XC-55,YC+85);      {3}
            LINE(XC-170,YC+25,XC-170,YC+85);    {1}
            LINE(XC-170,YC+25,XC-140,YC+25);    {1}
            LINE(XC-112,YC+61,XC-112,YC+85);    {2}
            XC:=XC+30;
            SETTEXTSTYLE (2,0,5);
            XC:=XC-25;
            OUTTEXTXY (XC+210,YC+100,'GND');
            OUTTEXTXY (XC-186,YC-60,'VCC');
            OUTTEXTXY (XC+158,YC+20,'Y6  ');     {9}
            OUTTEXTXY (XC+92,YC-8 ,' Y4 ');     {11}
            OUTTEXTXY (XC+49 ,YC-8 ,'Y3');      {12}
            OUTTEXTXY (XC+155 ,YC-8 ,'Y5');     {10}
XTXY (XC-69,YC-8 ,' Y1 ');      {14}
            OUTTEXTXY (XC-140,YC-8 ,'  Y0');    {15}
            OUTTEXTXY (XC-9  ,YC-8 ,'Y2 ');     {13}
            XC:=XC+25;
            XC:=XC-30;
            OUTTEXTXY (XC+109 ,YC+46,'G1 ');    {6}
            OUTTEXTXY (XC+162 ,YC+46,'Y7');     {7}
            OUTTEXTXY (XC-8 ,YC+37, 'CLK');     {4}
            OUTTEXTXY (XC+54,YC+46,'G2  ');     {5}
            OUTTEXTXY (XC+54,YC+40,'-   ');     {5}
            OUTTEXTXY (XC-56 ,YC+46,'C  ');     {3}
            OUTTEXTXY (XC-116 ,YC+46,'B  ');    {2}
            OUTTEXTXY (XC-130,YC+20,'A  ');
            XC:=XC+30;

            SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-252,YC-150,'IC 74131  3 TO 8 LINE DECODER        ');
OUTTEXTXY (XC-252,YC-115,'          WITH ADDRESS LATCH         ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN1-PIN3 :SELECT, *PIN7,PIN9-PIN15');
OUTTEXTXY (XC-1,YC+120,' :OUTPUTS, * PIN4 :CLOCK                ');
OUTTEXTXY (XC-280,YC+135,'PIN10 :ENABLE, * PIN15 :CARRY OUTPUT');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74131');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC131,74LS131,74H131');
OUTTEXTXY (XC,YC+165,'   ,74HCT131,74F131,74ALS131              ');
        END;

            {*************************************}
            {*        DATA OF PIN_IC_74147       *}
            {*************************************}

        PROCEDURE PIN_74147(XC,YC :INTEGER);
        BEGIN
```

```
YC:=YC-30;
BLOCK_16(XC,YC+25); XC:=XC-27;
LINE(XC+223,YC+25,XC+223,YC-34);    {9}
LINE(XC+223,YC+25,XC+204,YC+25);    {9}
CIRCLE (XC+198,YC+25,5);
LINE(XC+164,YC-18,XC+164,YC-34);    {10}
CIRCLE (XC+164,YC-14,5);
LINE(XC+107,YC-18,XC+107,YC-34);    {11}
CIRCLE (XC+107,YC-14,5);
LINE(XC+54,YC-18,XC+54,YC-34);      {12}
CIRCLE (XC+54 ,YC-14,5);
LINE(XC-4,YC-18,XC-4,YC-34);        {13}
CIRCLE (XC-4  ,YC-14,5);
LINE(XC-57,YC-18,XC-57,YC-34);      {14}
CIRCLE (XC-57 ,YC-14,5);
XC:=XC+27;
XC:=XC-30;
RECTANGLE(XC-140,YC-10 ,XC+195,YC+61);
LINE(XC+168,YC+69,XC+168,YC+85);    {7}
CIRCLE (XC+168,YC+65,5);
LINE(XC+113,YC+69,XC+113,YC+85);    {6}
CIRCLE (XC+113,YC+65,5);
LINE(XC+2,YC+69,XC-2,YC+85);        {4}
CIRCLE (XC+2,YC+65,5);
LINE(XC+56,YC+68,XC+56,YC+85);      {5}
CIRCLE (XC+56 ,YC+65,5);
LINE(XC-55,YC+69,XC-55,YC+85);      {3}
CIRCLE (XC-55 ,YC+65,5);
LINE(XC-170,YC+25,XC-170,YC+85);    {1}
LINE(XC-170,YC+25,XC-151,YC+25);    {1}
CIRCLE (XC-146,YC+25,5);
LINE(XC-112,YC+69,XC-112,YC+85);    {2}
CIRCLE (XC-112,YC+65,5); XC:=XC+30;
SETTEXTSTYLE (2,0,5); XC:=XC-25;
OUTTEXTXY (XC+210,YC+100,'GND');
OUTTEXTXY (XC-186,YC-60,'VCC');
OUTTEXTXY (XC+168,YC+20,'A   ');    {9}
OUTTEXTXY (XC-92,YC-8 ,' 1  ');     {11}
OUTTEXTXY (XC+49 ,YC-8 ,'2 ');      {12}
OUTTEXTXY (XC+155 ,YC-8 ,'9 ');     {10}
OUTTEXTXY (XC-69,YC-8 ,' D  ');     {14}
OUTTEXTXY (XC-9  ,YC-8 ,'3  ');     {13}
XC:=XC+25; XC:=XC-30;
OUTTEXTXY (XC+109 ,YC-46,'C  ');    {6}
OUTTEXTXY (XC+164 ,YC+46,'B ');     {7}
OUTTEXTXY (XC+2 ,YC+46, '7 ');      {4}
OUTTEXTXY (XC+54,YC+46,'8   ');     {5}
OUTTEXTXY (XC-56 ,YC+46,'6  ');     {3}
OUTTEXTXY (XC-116 ,YC+46,'5  ');    {2}
```

```
          OUTTEXTXY (XC-130,YC+20,'4  ');
          XC:=XC+30;

          SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-272,YC-150,'IC 74147  10 TO 4 LINE PRIORITY      ');
OUTTEXTXY (XC-252,YC-115,'          ENCODER                    ');
SETTEXTSTYLE(2,0,5);
OUTTEXTXY (XC-280,YC+120,'* PIN1-PIN5,PIN10-PIN13 :INPUT *PIN6 ');
OUTTEXTXY (XC-1,YC+120,'PIN7,PIN9,PIN14 :OUTPUT, * PIN15 :NC   ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74147');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups   ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC147,74LS147,74H147');
OUTTEXTXY (XC,YC+165,'   ,74HCT147,74F147,74ALS147              ');
        END;
          {************************************}
          {*        DATA OF PIN_IC_74447      *}
          {************************************}
         PROCEDURE PIN_74447(XC,YC :INTEGER);
          BEGIN
          YC:=YC-30;
          BLOCK_16(XC,YC+25);
          XC:=XC-27;
          LINE(XC+223,YC+25,XC+223,YC-34);   {9}
          LINE(XC+223,YC+25,XC+204,YC+25);   {9}
          CIRCLE (XC+198,YC+25,5);
          LINE(XC+164,YC-18,XC-164,YC-34);   {10}
          CIRCLE (XC+164,YC-14,5);
          LINE(XC+107,YC-18,XC+107,YC-34);   {11}
          CIRCLE (XC+107,YC-14,5);
          LINE(XC+54,YC-18,XC+54,YC-34);     {12}
          CIRCLE (XC+54 ,YC-14,5);
          LINE(XC-4,YC-18,XC-4,YC-34);       {13}
          CIRCLE (XC-4  ,YC-14,5);
          LINE(XC-57,YC-18,XC-57,YC-34);     {14}
          CIRCLE (XC-57 ,YC-14 5);
          LINE(XC-117,YC-18 ,XC-117,YC-34);  {15}
          CIRCLE (XC-117,YC-14,5);
          XC:=XC+27;
          XC:=XC-30;
          RECTANGLE(XC-140,YC-10 ,XC+195,YC+61);
          LINE(XC+168,YC+61,XC+168,YC+85);   {7}
          LINE(XC+113,YC+61,XC-113,YC+85);   {6}
          LINE(XC+2,YC+69,XC+2,YC+85);       {4}
          CIRCLE (XC+2,YC+65,5);
          LINE(XC+56,YC+68,XC+56,YC+85);     {5}
          CIRCLE (XC+56 ,YC+65,5);
          LINE(XC-55,YC+69,XC-55,YC+85);     {3}
          CIRCLE (XC-55,YC+65,5);
```

```
        LINE(XC-170,YC+25,XC-170,YC+85);    {1}
        LINE(XC-170,YC+25,XC-140,YC+25);    {1}
        LINE(XC-112,YC+61,XC-112,YC+85);    {2}
        XC:=XC+30;
        SETTEXTSTYLE (2,0,5);
        XC:=XC-25;
        OUTTEXTXY (XC+210,YC+100,'GND');
        OUTTEXTXY (XC-186,YC-60,'VCC');
        OUTTEXTXY (XC+168,YC+20,' e  ');     {9}
        OUTTEXTXY (XC+92,YC-8 ,' c  ');     {11}
        OUTTEXTXY (XC+49 ,YC-8 ,'b ');      {12}
        OUTTEXTXY (XC+155 ,YC-8 ,'d ');     {10}
        OUTTEXTXY (XC-69,YC-8 ,' g  ');     {14}
        OUTTEXTXY (XC-140,YC-8 ,' f ');     {15}
        OUTTEXTXY (XC-9  ,YC-8 ,'a  ');     {13}
        XC:=XC+25;
        XC:=XC-30;
        OUTTEXTXY (XC+109 ,YC-46,'D  ');    {6}
        OUTTEXTXY (XC+164 ,YC+46,'A ');     {7}
        OUTTEXTXY (XC-8 ,YC+46, 'RBO'); {4}
        OUTTEXTXY (XC+49,YC+46,'RBI ');     {5}
        OUTTEXTXY (XC-59 ,YC+46,'LT ');     {3}
        OUTTEXTXY (XC-116 ,YC+46,'C  ');    {2}
        OUTTEXTXY (XC-130,YC-20,'B  ');
        XC:=XC+30;

        SETTEXTSTYLE (1,0,4);
OUTTEXTXY (XC-315,YC-150,'IC 74447  BCD TO 7 SEGMENT DECODER   ');
SETTEXTSTYLE(2,0,5
OUTTEXTXY (XC-280,YC+120,'* PIN9-PIN15 :OUTPUT, * PIN1,PIN2,PIN');
OUTTEXTXY (XC-1,YC+120,'6,PIN7 :INPUT, * PIN4 :RB OUTPUT, * PIN');
OUTTEXTXY (XC-280,YC+135,'5 :RB INPUT                          ');
OUTTEXTXY (XC-280,YC+150,'The base connection show for the 74447');
OUTTEXTXY (XC,YC+150,'   apply to all of the family groups  ');
OUTTEXTXY (XC-280,YC+165,'These include : 74AHC447,74LS447,74H447');
OUTTEXTXY (XC,YC+165,'   ,74HC7447,74T447,74ALS447              ');
        END;

    END.
```

การสร้าง UNIX ในการเชื่อมโยงกับ DTH

ในภาษา PASCAL นั้นจะไม่มีคำสั่งในการติดต่อกับ PORT โดยตรงแต่เราสามารถที่ติดต่อ PORT โดย LINK ผ่านภาษา ASSEMBLY ได้โดยวิธีการดั่งที่กล่าวมาแล้วในบทที่ 1 เรื่อง การเชื่อมโยงภาษา PASCAL กับ ASSEMBLY สำหรับ UNIX ดังกล่าวจะมีชื่อว่า DTS_LINK.ASM

การสร้าง PORGRAM DTS LINK.ASM

```
;*********************************************
;*                                           *
;*      PROGRAM LINK DTS_MAIN TO DTH_ICBOX   *
;*                                           *
;*********************************************

      code      segment byte public
                assume  cs:code
                public control
   control      proc        near
                push        bp
                mov         bp,sp
                mov         ax,[bp+10] ;LOAD DATA OUTPUT
                mov         dh,02h     ;SET ADDRESS HIGH
                mov         dl,[bp+8]  ;LOAD ADDRESS LOW
                out         dx,ax      ;OUT DATA TO PORT
                stosw
                pop         bp
                ret         4
                control     endp
      code      ends
                end

        ;*-----END OF PROGRAM DTS_LINK-----*
```

DIGITAL TRAINER HARDWARE

บทที่ 3

การสร้างชุด
DIGITAL TRAINER HARDWARD

การ INTERFACE และ DECODER กับ MICRO COMPUTER (16 BIT)

การจัดแอดเดรสสำหรับ I/O ใน IBM/PC

ในหัวข้อนี้จะกล่าวถึงวิธีการอ้างและใช้งาน ADDRESS ต่าง ๆ ของ PORT I/O ที่ใช้งานอยู่ใน IBM/PC

การอ้าง ADDRESS ใน PORT I/O

ในการควบคุมและตรวจสอบสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิพซับพอร์ท หรือ CARD ต่าง ๆ ที่ใช้ระบบของ IBM/PC นั้นจะกระทำโดยผ่านทาง PORT I/O ของระบบดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกรณ์เหล่านี้จึงจำเป็นต้องศึกษาถึงวิธีการควบคุม PORT I/O ต่าง ๆ ของระบบด้วย และเนื่องจากการควบคุมหรือติดต่อ กับ PORT เหล่านี้ ต้องกระทำโดยอ้างถึง ADDRESS ของ port i/o เหล่านั้นโดยตรง เราจึงจำเป็นต้องศึกษา ถึงหลักการอ้าง ADDRESS ของ 8088 ใน IBM/PC ด้วย

สำหรับ ADDRESS ของ PORT I/O ต่าง ๆ นั้น จะเป็น ADDRESS ที่ถูกสร้างขึ้นโดย 8088 ซึ่ง ADDRESS เหล่านี้เป็น ADDRESS ที่จัดไว้สำหรับ PORT I/O โดยเฉพาะ คือแยกจาก ADDRESS ของหน่วยความจำโดยเด็ดขาด ส่วนการส่งข้อมูล ให้กับ PORT เหล่านี้จำทำได้โดยการใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลนั้นไปยัง ADDRESS ของ PORT ที่ต้องการ และ สำหรับการตรวจสอบหรือการอ่านข้อมูลจาก PORT ก็จะทำได้โดยการใช้คำสั่ง IN ของ 8088 อ่านข้อมูลจาก ADDRESS ของ PORT ที่ต้องการเช่นกัน

ภายในไมโครโปรเซสเซอร์ 8088 นี้จะมี ADDRESS สำหรับใช้กับ PORT I/O อยู่ทั้งสิ้น 65536 หรือ 64 K ADDRESS ซึ่งทำให้การอ้าง ADDRESS ของ PORT I/O ที่ทำงานร่วมกับ 8088 นั้นต้องใช้จำนวนเส้น ADDRESS ใน BUS ADDRESS ทั้งสิ้น 16 เส้น คือ A0-A15 แต่สำหรับใน IBM/PC นี้ถูกออกแบบมาให้ใช้เส้น ADDRESS เฉพาะ 10 เส้นล่างคือ A0-A9 เท่านั้นดังนั้นในการอ้างถึง ADDRESS ของ PORT ของอุปกรณ์หรือชิบซับพอร์ทใดๆ ที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้น

ADDRESS เพียง 10 เส้นด้วยโดยเส้น ADDRESS ที่เหลือคือ A10-A15 นั้นจะไม่ถูกนำไปใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้น ADDRESS A10-A15 นี้จะไม่ถูกนำไปใช้งาน แต่ค่า ADDRESS บนเส้น ADDRESS เหล่านี้ยังคงเปลี่ยนแปลงตามค่า ADDRESS ของ PORT ที่กำหนดไว้ในคำสั่ง OUT หรือ IN อยู่ด้วย เพียงแต่ไม่ได้ถูกนำมา DECODE ร่วมกับ ADDRESS A0-A9 เท่านั้น ตัวอย่างเช่นในการใช้คำสั่ง OUT ส่งข้อมูลไปยัง PORT ที่ตรงกับ ADDRESS 0410H , 0810 , 0C10H ทั้งนี้เนื่องจาก ADDRESS 6 BIT บนไม่ได้ถูกใช้งาน จึงทำให้การเปลี่ยนแปลงค่า ADDRESS บนเส้น ADDRESS A10-A15 นั้น ไม่ทำให้เกิดความแตกต่างใด ๆ เกิดขึ้น

เนื่องจากใน IBM/PC ได้ใช้งานเส้น ADDRESS เพียง 10 เส้น (คือ A9-A9) ตัวนั้น จึงสามารถที่จะอ้าง ADDRESS ของพอร์ท ได้สูงสุดเพียง 1024 PORT (จากจำนวน 64K PORT) เท่านั้นนอกจากนี้ในกรณีที่เป็นการอ่านข้อมูลในบิท A9 เป็น "0" แล้ว เราจะทำการอ่านข้อมูลจากพอร์ทของ IBM/PC ข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่ง PORT 1024 PORT ออกเป็น 2 ส่วน ๆ ละ 512 PORT อีกด้วย กล่าวคือถ้าข้อมูลในบิท A9 เป็น "0" แล้ว เราจะทำการอ่านข้อมูลได้เฉพาะจาก PORT ของอุปกรณ์ หรือชิพซัพพอร์ที่อยู่บนเมนบอร์ด (MAIN BOARD) ของ IBM/PC เช่น 8253-5 8237-5 หรือ 8259A เท่านั้น แต่ถ้าข้อมูลในบิท A9 นี้เป็น "1" ก็จะทำการอ่านข้อมูลได้เฉพาะจาก PORT ที่อยู่บน CARD ต่างๆ เท่านั้น

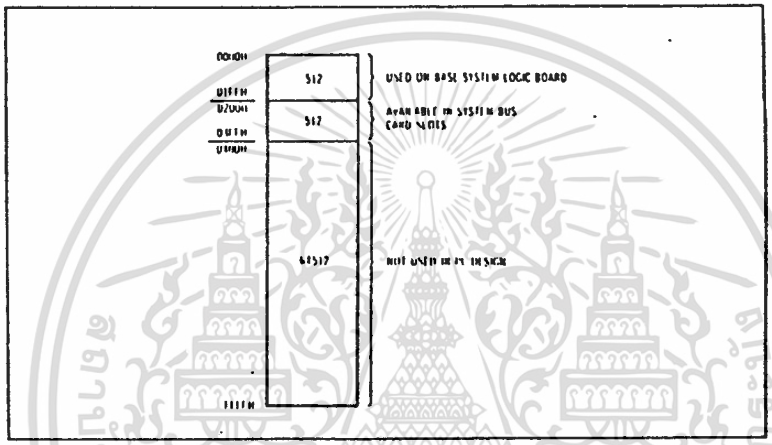
จากที่ได้กล่าวมานั้นจะสรุปได้ว่า PORT บน IBM/PC ทั้ง 1024 PORT ถูกแบ่งออกเป็น 2 กลุ่ม โดยที่กลุ่มแรกเป็นกลุ่มของ PORT ที่อยู่บน MAINBOARD และกลุ่มที่ 2 เป็นกลุ่มที่จัดเตรียมไว้สำหรับ PORT ที่อยู่บน CARD ต่าง ๆ

สำหรับในกรณีของการส่งข้อมูลให้กับ PORT ใด ๆ ใน IBM/PC ได้ดังนั้นการเลือก ADDRESS สำหรับ PORT ที่อยู่บน CARD ก็สามารถทำได้โดยสะดวก แต่อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึง ก็คือถ้า ADDRESS ที่เราเลือกให้กับ PORT นี้ตรงกับค่า ADDRESS เต็มที่มีอยู่บน MAIN BOARD แล้ว เมื่อเราทำากรส่งข้อมูลให้กับ PORT ที่อยู่ใน ADDRESS นี้ก็จะเท่ากับเป็นการส่งข้อมูลให้กับทั้ง PORT ที่อยู่บน MAIN BOARD และ PORT ที่อยู่บน CARD ซึ่งในกรณีเช่นนี้อาจจะก่อให้เกิดความผิดพลาดขึ้นได้เช่นกัน ดังนั้นในการกำหนดค่า ADDRESS ที่ ADDRESS BIT A0 มีค่าเป็น "1" คือ ADDRESS OFEOOH เท่านั้น

การใช้งานแอดเดรสสำหรับ PORT I/O ใน IBM/PC

จากที่ได้กล่าวไว้ในหัวข้อที่ผ่านมานั้น PORT I/O ทั้ง 1024 PORT ใน IBM/PC จะถูกแบ่งออกเป็น 2 กลุ่ม ๆ ละ 512 PORT สำหรับในหัวข้อนี้จะกล่าว

ถึงการใช้งาน PORT ต่างๆ เหล่านี้ โดยจะแบ่งออกเป็น 2 กลุ่มตามที่ได้อธิบายไว้ในหัวข้อที่ผ่านมาดังนี้

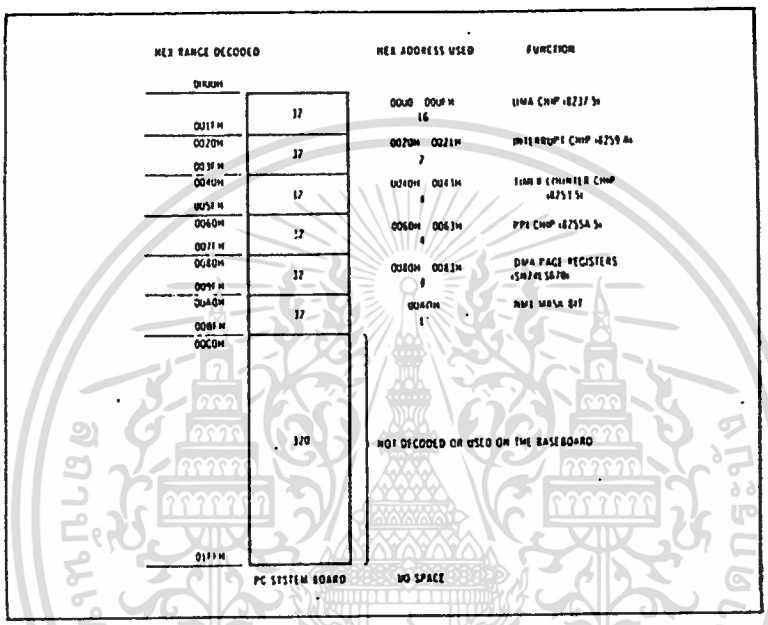


รูปที่ 3.1 การใช้งาน ADDRESS ของ PORT บน IBM/PC

1. ในกลุ่มแรกนี้เป็นกลุ่มของ PORT I/O ที่อยู่บน MAIN BCARD ของ IBM/PC ซึ่งจะมี ADDRESS อยู่ในตำแหน่ง 0000H จนถึง 01FFH หรือ ADDRESS ที่มีบิท A9 เป็น "0" นั่นเอง

สำหรับ ADDRDSS ของ PORT I/O ในกลุ่มนี้จะถูกใช้ในการอ้าง ADDRESS ของ ชิพซัพพอร์ท และ อุปกรณ์ที่เป็น I/O ต่าง ๆ บนเมนบอร์ดของ IBM/PC เช่น ADDRESS 0000H จนถึง 000FH จะถูกใช้เป็น ADDRESS สำหรับ 8237-5 DMA Controller เป็นต้น ในรูปที่ 3.2 จะแสดงถึงการใช้งาน ADDRESS ต่าง ๆ ตั้ง

แต่ OOOOH จนถึง O1FFH ในการอ้าง ADDRESS ของชิพพอร์ท และอุปกรณ์ต่างๆ ที่ทำหน้าที่เป็น I/O บน MAIN BOARD ของ IBM/PC

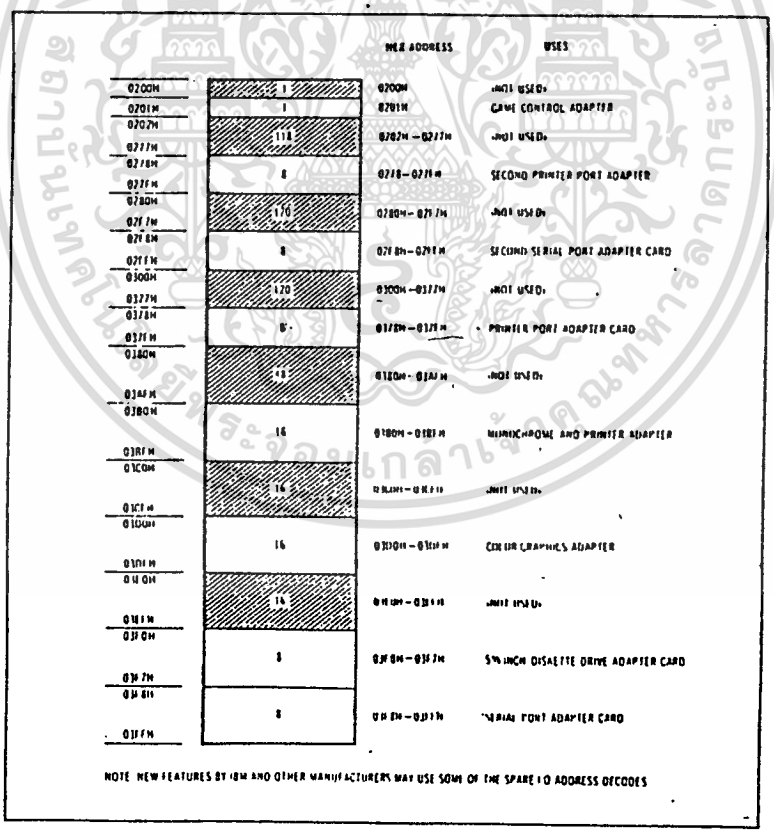


รูปที่ 3.2 การใช้งาน ADDRESS ต่าง ๆ สำหรับ PORT I/O ของ IBM/PC

จากรูปข้างต้นจะเห็นว่า ADDRESS OOCOH จนถึง ADDRESS O1FFH นั้น ไม่ได้ถูกใช้งานบนเมนบอร์ดของ IBM/PC ดังนั้นในกรณีนี้เราก็สามารถที่จะใช้งาน ADDRESS ต่าง ๆ เหล่านี้ได้ แต่อย่างไรก็ตาม ADDRESS เหล่านี้ยังคงถูกดีโคดให้เป็น ADDRESS ที่ใช้ในการอ่านข้อมูลจาก PORT I/O บนเมนบอร์ดเท่านั้นดังนั้น การใช้ค่า ADDRESS OOCOH-O1FFH กับ PORT I/O บน CARD หรือวงจรอินเทอเฟสที่เราสร้างขึ้นนั้น ต้องเป็น PORT OUTPUT เพียงชนิดเดียวเท่านั้น กล่าวคือ จะทำการอ่านข้อมูลจาก PORT I/O ที่มีค่า ADDRESS อยู่ในช่วง OOCOH-O1FFH ไม่ได้

2. ในกลุ่มที่สองนี้ จะเป็นกลุ่มของ PORT I/O ที่ถูกใช้งานอยู่บน CART ที่ใช้เสียบบน SLOT ต่าง ๆ ของ IBM/PC สำหรับ ADDRESS ของ PORT เหล่านี้จะเริ่มต้นจาก ADDRESS 0200H จนถึง 03FFH ซึ่งก็คือ ADDRESS ที่มีบิท A9 เป็น "1" นั่นเอง สำหรับการใช้งาน ADDRESS ของ PORT I/O ในกลุ่มนี้จะแสดงได้ดังรูป 3.3

อย่างไรก็ตามการใช้งาน ADDRESS ในกลุ่มนี้อาจจะเปลี่ยนแปลงไปได้ ทั้งนี้ขึ้นอยู่กับการใช้งาน CARD ต่าง ๆ ร่วมกับ IBM/PC โดยการ์ดที่ถูกออกแบบผลิตขึ้นใหม่นั้นอาจจะใช้ค่า ADDRESS ต่าง ๆ ที่เหลืออยู่นี้ได้ ดังนั้นก่อนที่จะทำการออกแบบวงจรอินเทอเฟสที่จำเป็นต้องใช้ค่า ADDRESS สำหรับ PORT I/O จึงควรจะตรวจสอบดูก่อนว่า CARD ต่าง ๆ ที่ใช้อยู่ในระบบของ IBM/PC ที่เราใช้งานอยู่นั้นมี CARD ใดบ้าง และ CARD เหล่านั้นใช้งาน ADDRESS จากนั้นจึงทำการออกแบบวงจรอินเทอเฟส โดยเลือกใช้เฉพาะ ADDRESS ที่ยังไม่ถูกใช้งาน

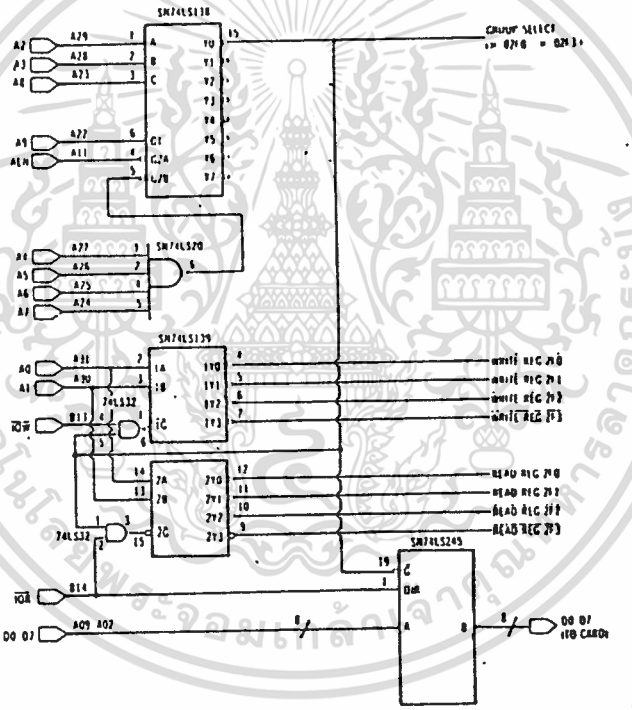


รูปที่ 3.3 การใช้งาน ADDRESS สำหรับ PORT I/O บน CARD ต่าง ๆ

เทคนิคในการดีโคดแอดเดรสสำหรับ I/O

ในหัวข้อต่าง ๆ ที่ผ่านมาข้างต้นนั้น ได้กล่าวถึงการอ้าง ADDRESS และการใช้งาน ADDRESS ต่างๆ ของ PORT I/O ใน IBM/PC สำหรับในหัวข้อนี้จะกล่าวถึง วิธีการต่าง ๆ ที่ใช้ในการดีโคดแอดเดรสต่าง ๆ ให้เป็นไปตามที่เราต้องการ การดีโคดแบบ Fixed

วิธีการดีโคดแบบนี้เป็นวิธีที่ง่าย และสดวกในการดีโคดแอดเดรสหรือกลุ่มของ ADDRESS ของ PORT I/O ซึ่งวิธีนี้เป็นการกำหนดจำนวน ADDRESS ที่เราต้องการใช้จากนั้น จึงทำการเลือก BLOCK ของ ADDRESS ที่ยังไม่ถูกใช้งาน โดย CARD หรือ วงจรอินเทอเฟสอื่น ๆ แล้วจึงออกแบบวงจรที่ทำการดีโคด ADDRESS ที่เราต้องการ สำหรับตัวอย่างวงจรที่ใช้ในการดีโคด ADDRESS ในแบบนี้จะแสดงได้ดังรูป 3.4

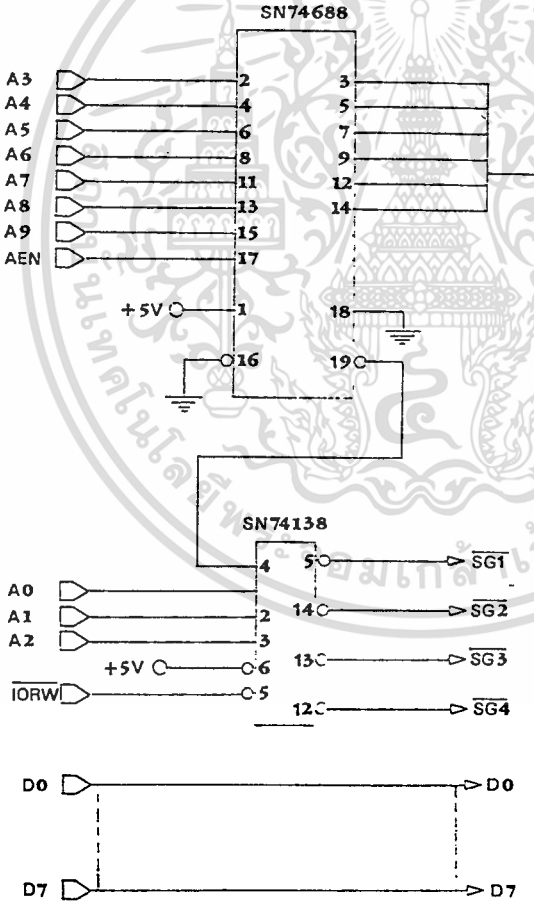


รูปที่ 3.4 ตัวอย่างวงจรดีโคดแอดเดรสแบบ Fixed

การสร้างชุดเชื่อมต่อกับ MICRO COMPUTER

สำหรับชุด DIGITAL TRAINER HARDWARE นี้จะใช้การ DECODER แบบ FIXED โดยจะใช้ค่า ADDRESS I/O PORT ตั้งแต่ OBFO-038F8 สัญญาณที่ส่งมาจาก MICRO COMPUTER จะประกอบด้วยสาย DATA D0-D7 และสาย ADDRESS A0-A7

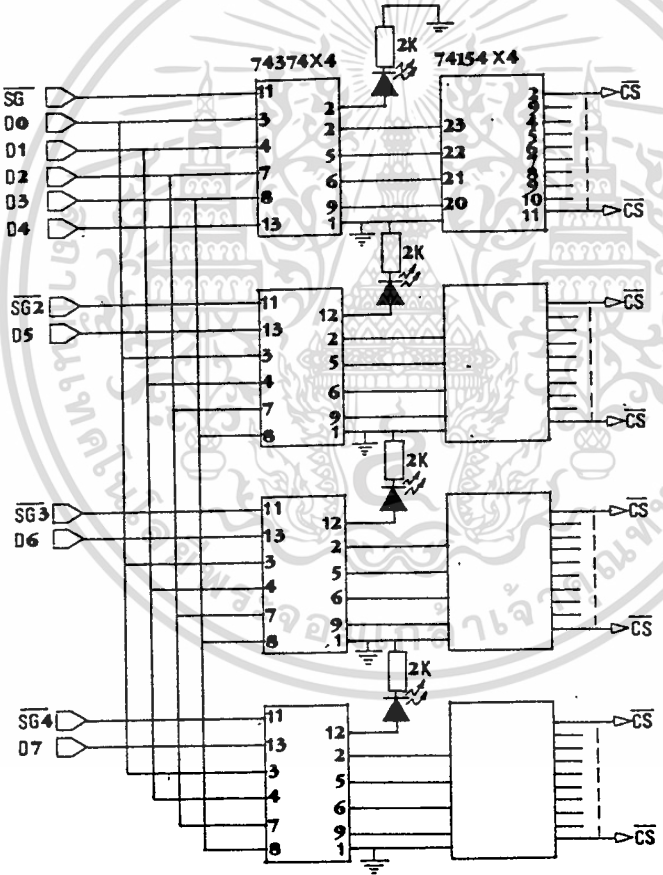
สายข้อมูล D0-D7 จะเป็นข้อมูลที่ใช้กำหนดเบอร์ IC ซึ่งเป็นรหัสแต่ละตัวของ IC ส่วนสาย A0-A9 นั้นเราจะใช้ A3-A9 ในการ DECODER หมายเลข PORT โดยใช้ IC 74688 เมื่อหมายเลข PORT ที่ตั้งไว้ตรงกับค่า PORT ที่ส่งมาจะทำให้ O/P ของ IC 74688 มีระดับแรงดัน "0" ส่วน A0-A2 นั้นจะใช้ ในการ DECODER ในส่วนของกลุ่ม IC ที่สร้างขึ้น โดยจะใช้ IC เบอร์ 74138 ในการ DECODER



รูปที่ 3.5 แสดงวงจร DECODER ของชุด DTH

การสร้างชุดสร้างสัญญาณในการเลือก IC

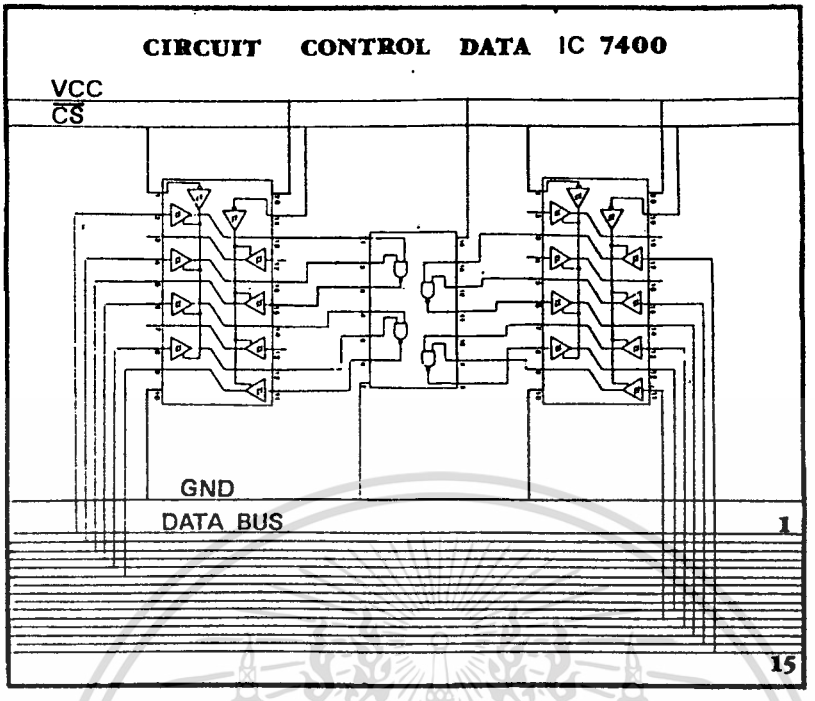
เมื่อเราทำการเลือก IC เบอร์ต่าง ๆ ตาม MENU 3 และเลือกเบอร์ SLOT ตาม MENU 4 จะทำให้ MICRO COMPUTER ส่งค่าต่าง ๆ ในการควบคุมชุด DIGITAL TRAINER HARDWARE ผ่านมาทาง SLOT ของ MICRO COMPUTER ค่าต่าง ๆ ที่ส่งมาจะประกอบด้วยสัญญาณในการเลือกเบอร์ IC และสัญญาณในการเลือก SLOT สำหรับสัญญานในการเลือกเบอร์ IC นั้นจะส่งมาทาง D0-D7 ของ SLOT MICRO COMPUTER ส่วนสัญญาณในการเลือก SLOT นั้นจะใช้ A0-A2 และเนื่องจากเราทำการแบ่ง IC ในแต่ละกลุ่มออกเป็นอีก 2 พวก คือ IC ที่มีขา 14 ขา และ IC ที่มี 16 ขา สาเหตุที่เราต้องทำการแบ่ง IC ออกเป็น 2 พวก ก็เพื่อความสดวกในการออกแบบ ส่วนของ HARD WARE



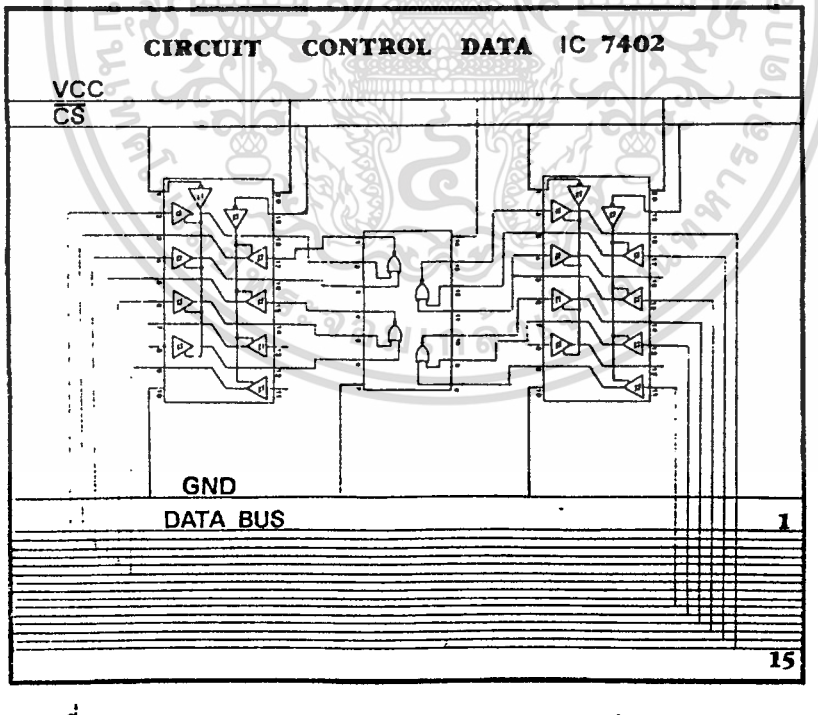
รูปที่ 3.6 แสดงวงจรในการสร้างสัญญาณเลือก IC

การสร้างชุดควบคุม DATA ของ IC

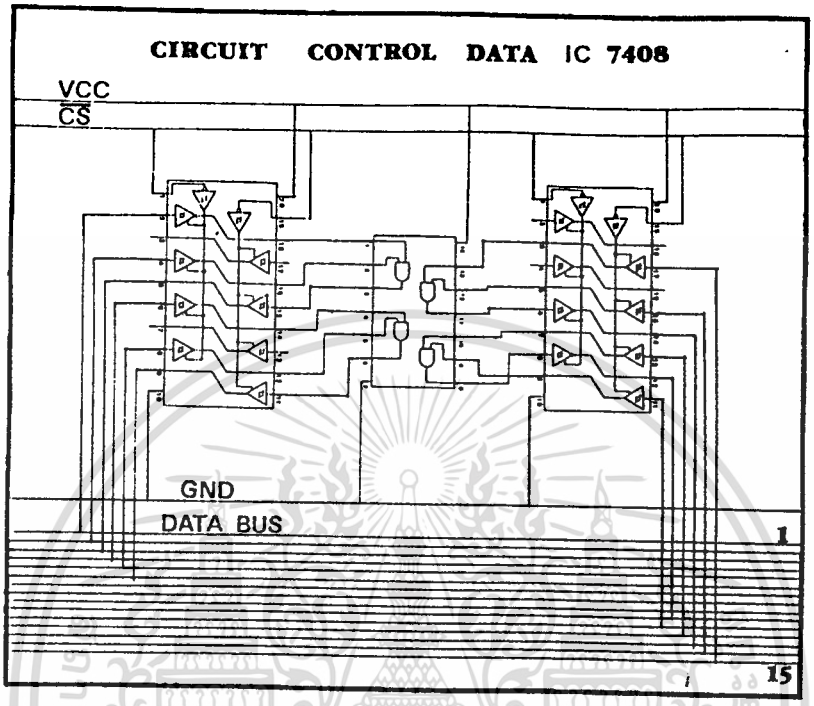
ในการควบคุม DATA ของ IC แต่ละตัวเราใช้ BUFFER ในการควบคุมข้อมูลของ IC แต่ละตัวโดย IC 1 ตัว จะใช้ BUFFER ประมาณ 14 ชุด (IC 1 ตัว มี BUFFER 8 ชุด) ในการควบคุม BUFFER แต่ละชุดเราจะมีสัญญาณ CS0-CS10 ที่ได้มาจากชุดสร้างสัญญาณในการเลือกเบอร์ IC เป็นตัวควบคุมขา CONTROL ของ BUFFER ในการที่จะให้ IC ตัวนั้นสามารถรับข้อมูลหรือจ่ายข้อมูลออกไปที่ DATA BUS ของชุด DTH



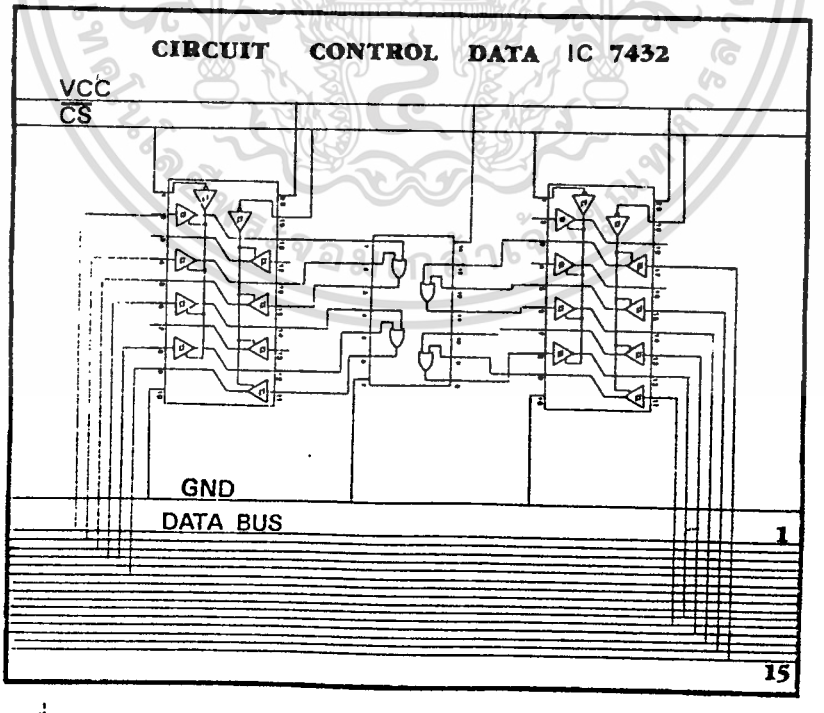
รูปที่ 3.7 แสดงการควบคุม DATA ของ IC เบอร์ 7400



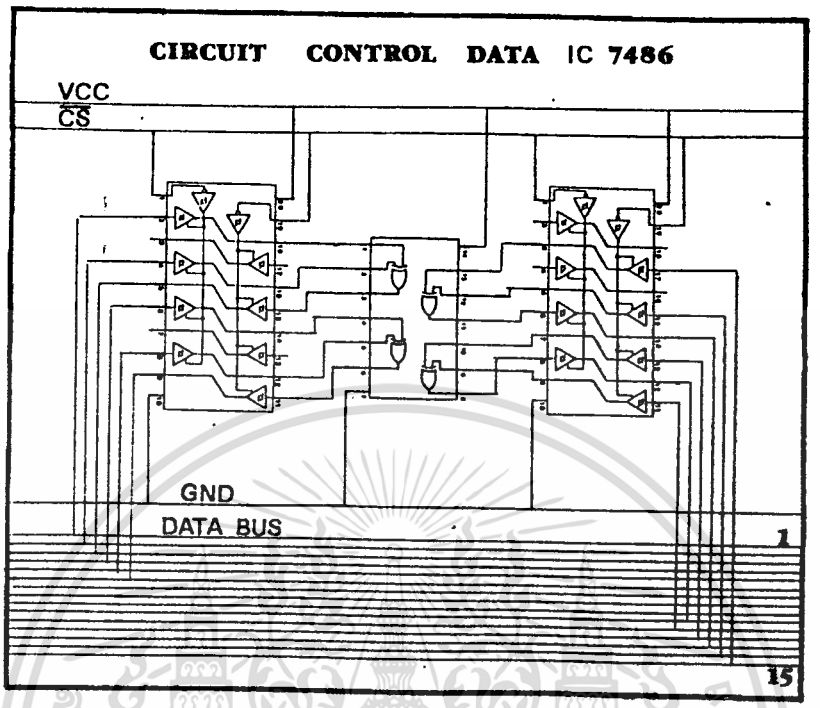
รูปที่ 3.8 แสดงการควบคุม DATA ของ IC เบอร์ 7402



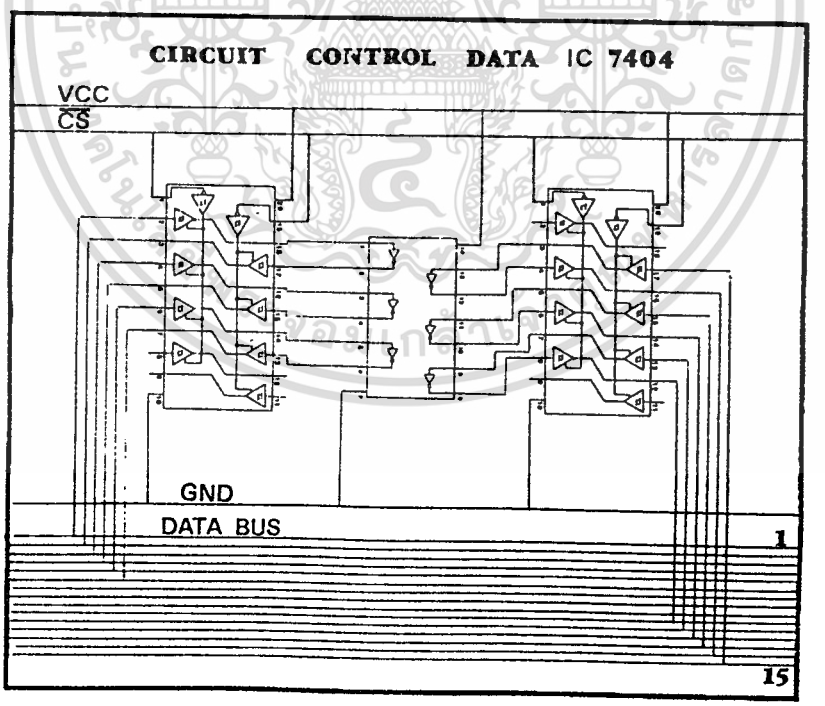
รูปที่ 3.9 แสดงการควบคุม DATA ของ IC เบอร์ 7408



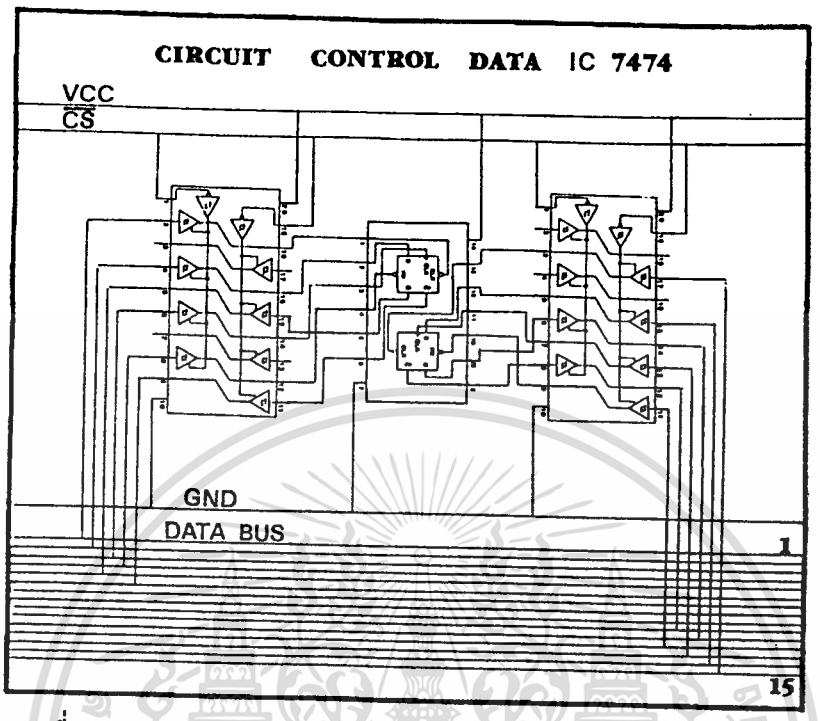
รูปที่ 3.10 แสดงการควบคุม DATA ของ IC เบอร์ 7432



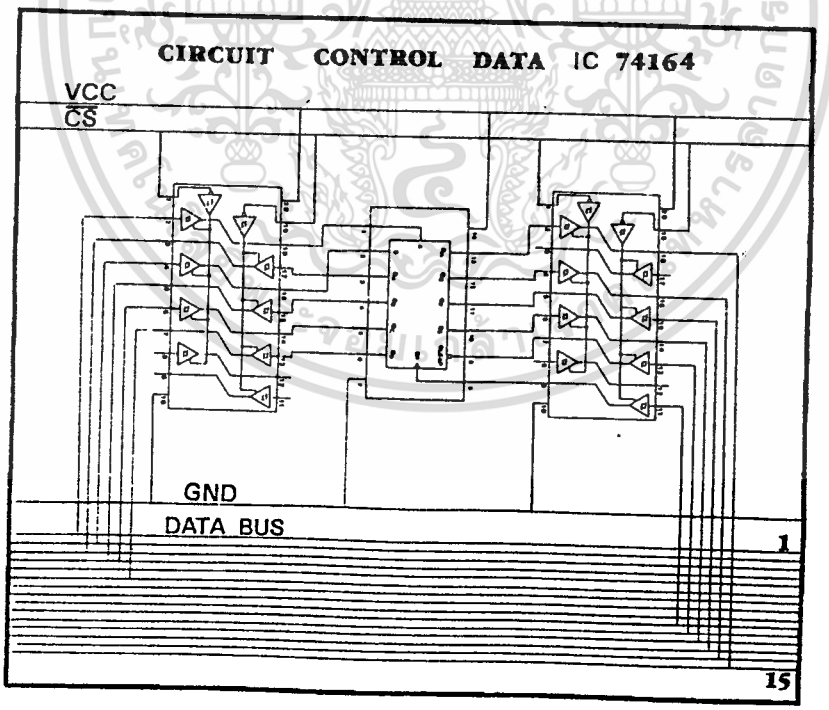
รูปที่ 3.11 แสดงการควบคุม DATA ของ IC เบอร์ 7486



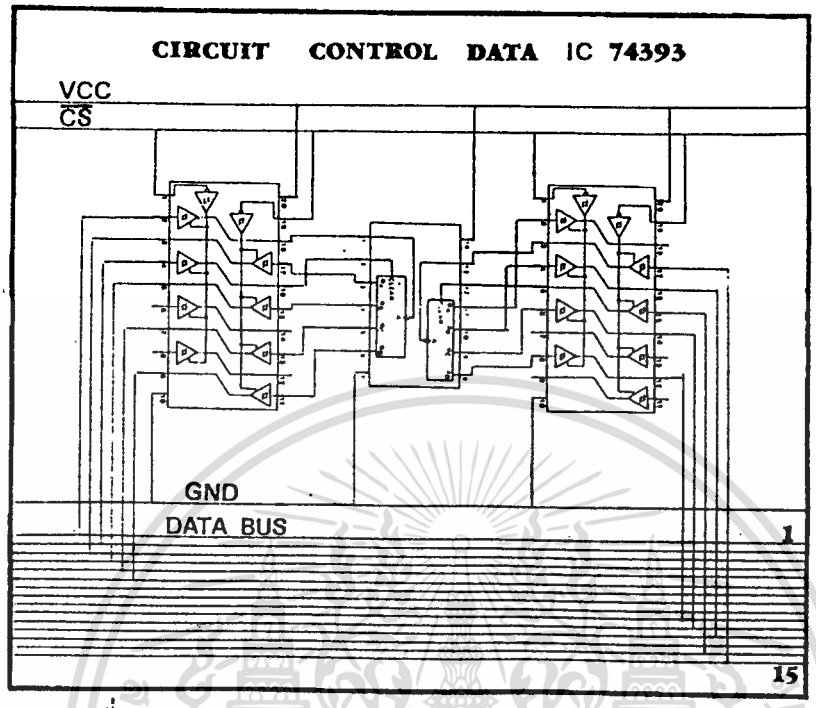
รูปที่ 3.12 แสดงการควบคุม DATA ของ IC เบอร์ 7404



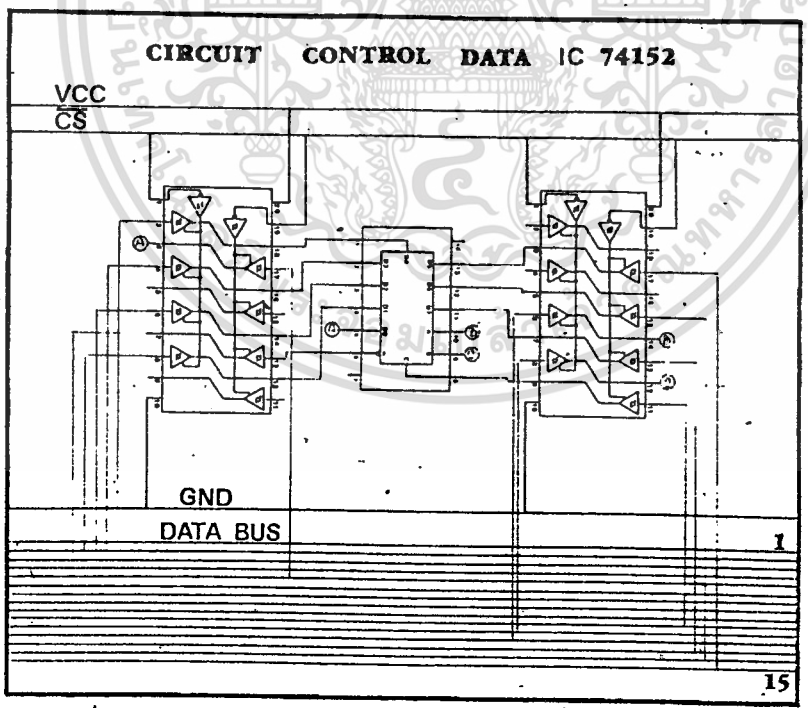
รูปที่ 3.13 แสดงการควบคุม DATA ของ IC เบอร์ 7474



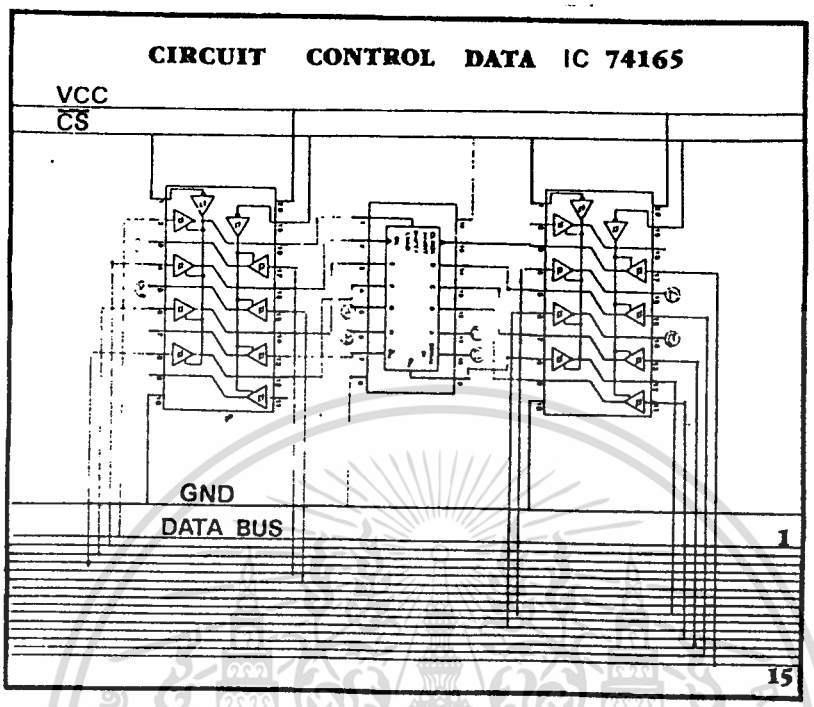
รูปที่ 3.14 แสดงการควบคุม DATA ของ IC เบอร์ 74164



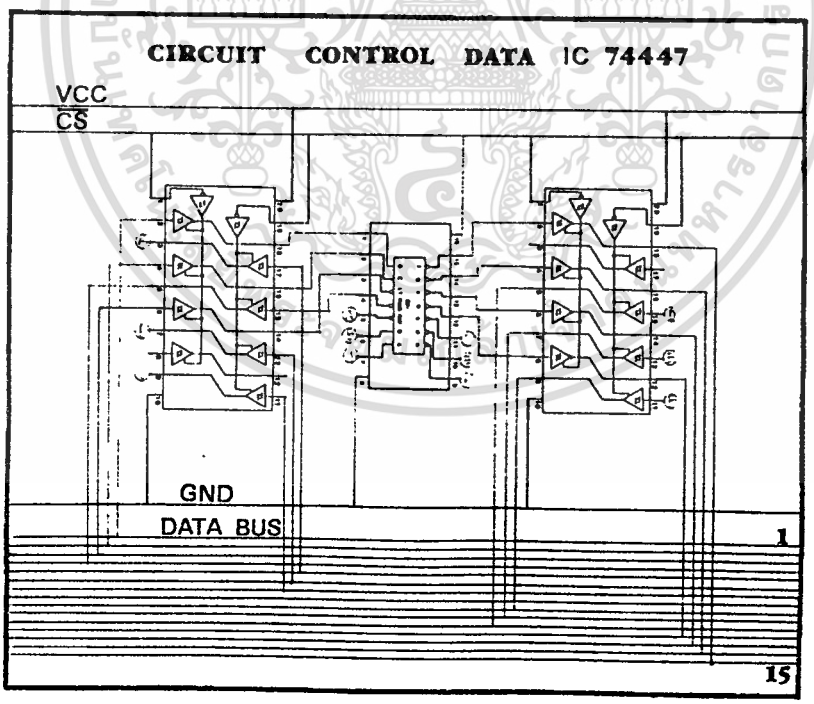
รูปที่ 3.15 แสดงการควบคุม DATA ของ IC เบอร์ 74393



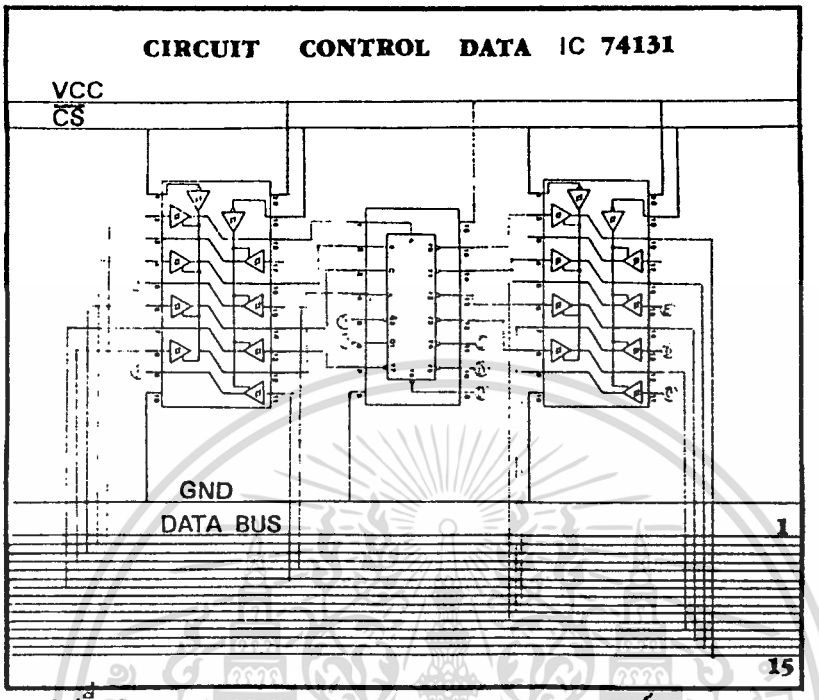
รูปที่ 3.16 แสดงการควบคุม DATA ของ IC เบอร์ 74152



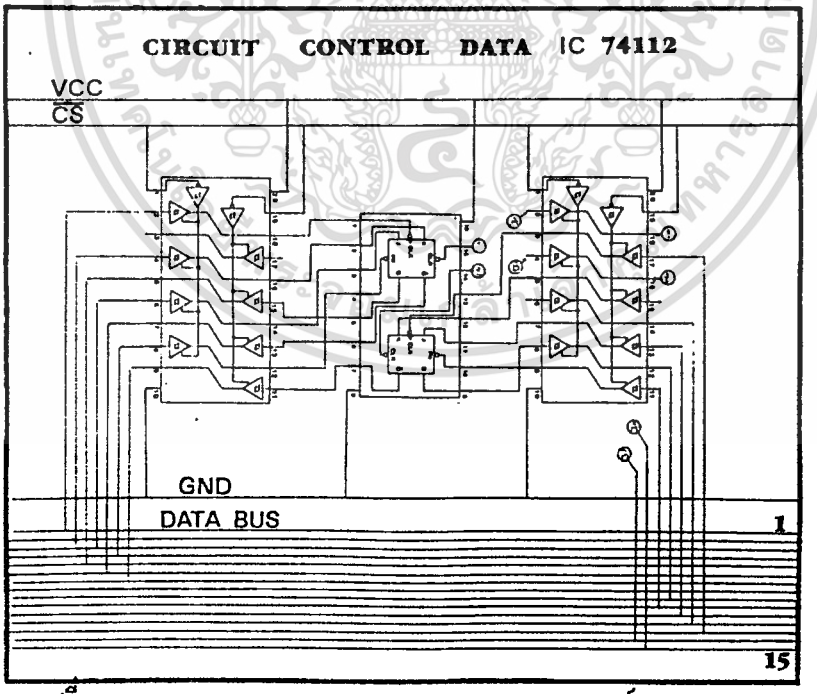
รูปที่ 3.17 แสดงการควบคุม DATA ของ IC เบอร์ 74165



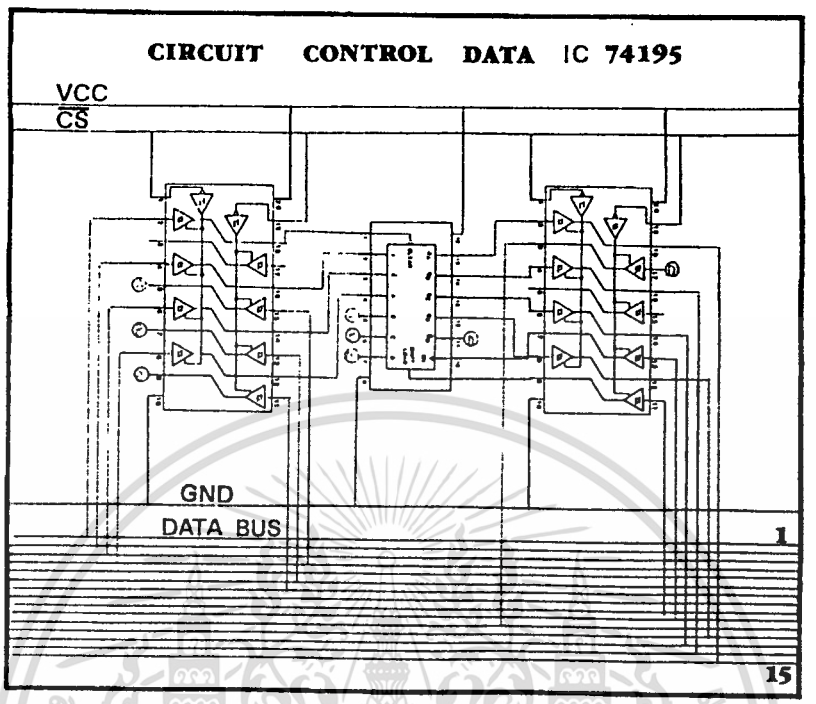
รูปที่ 3.18 แสดงการควบคุม DATA ของ IC เบอร์ 74447



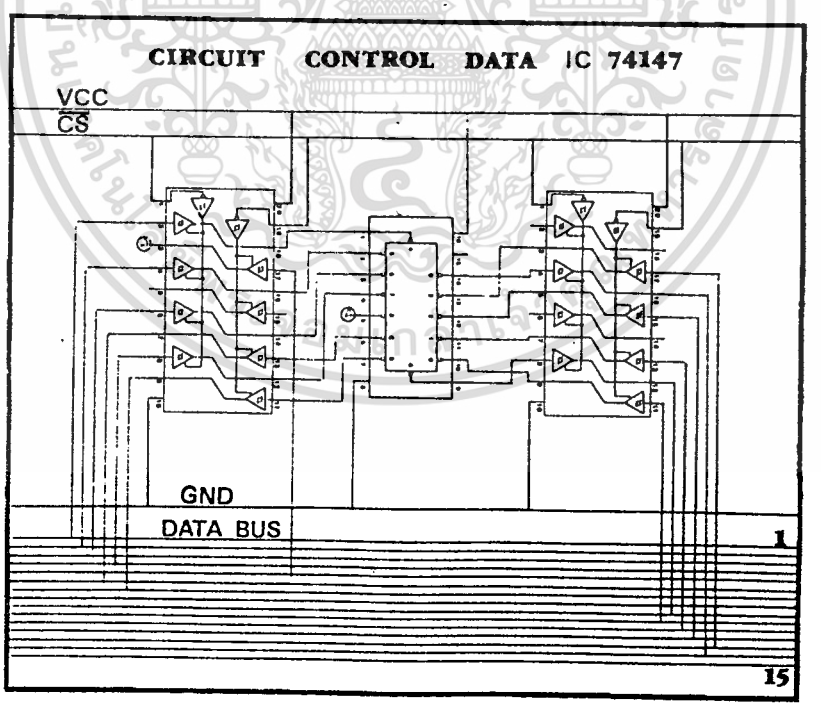
รูปที่ 3.19 แสดงการควบคุม DATA ของ IC เบอร์ 74131



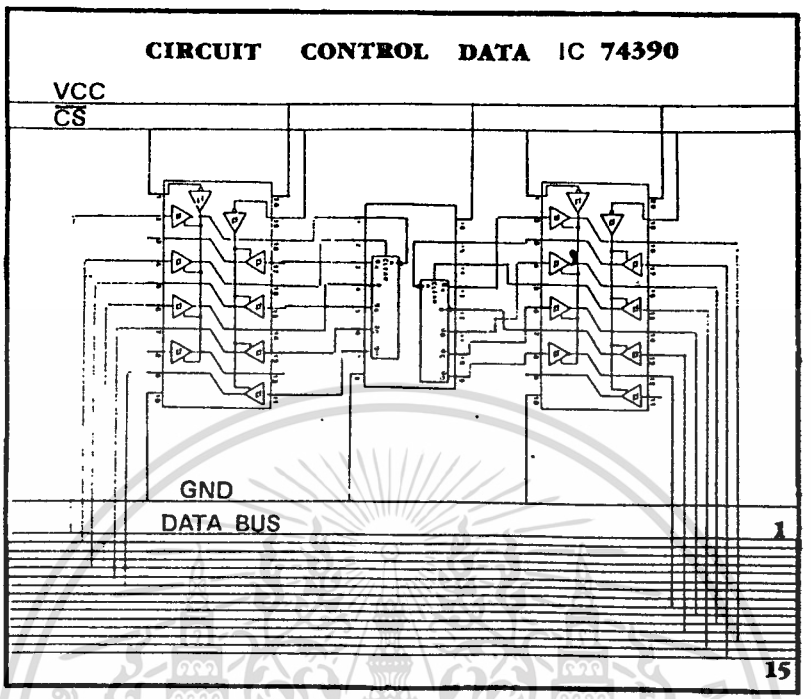
รูปที่ 3.20 แสดงการควบคุม DATA ของ IC เบอร์ 74112



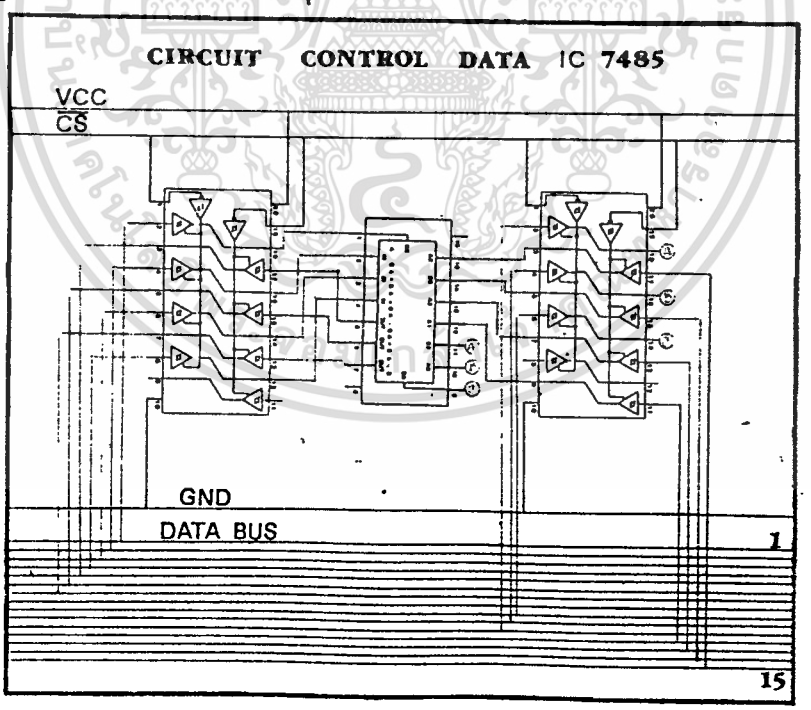
รูปที่ 3.21 แสดงการควบคุม DATA ของ IC เบอร์ 74195



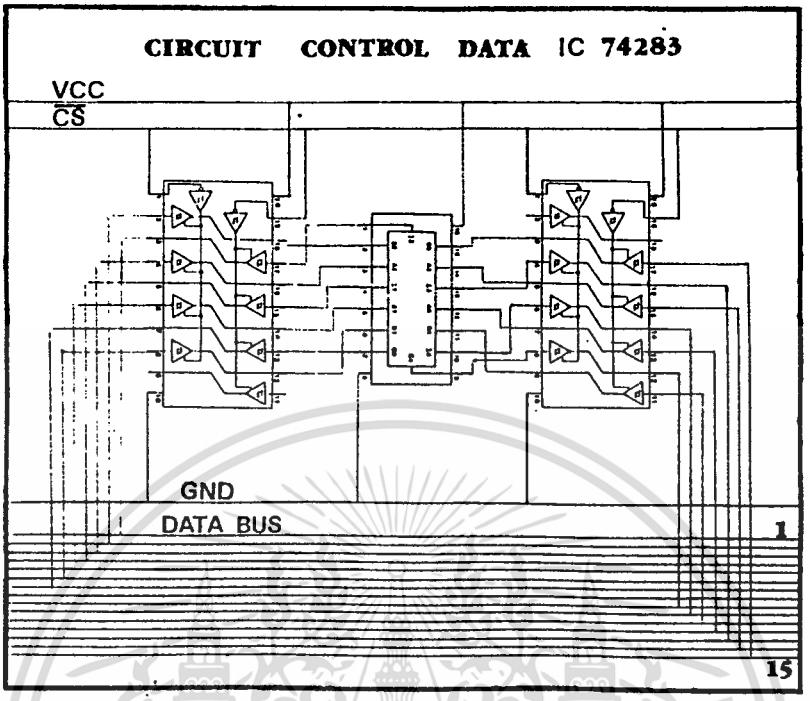
รูปที่ 3.22 แสดงการควบคุม DATA ของ IC เบอร์ 74147



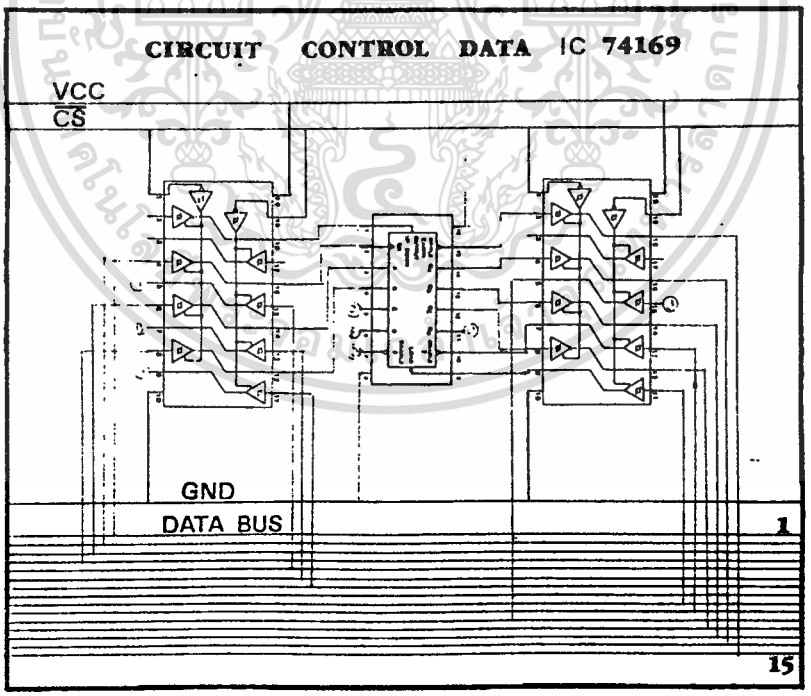
รูปที่ 3.23 แสดงการควบคุม DATA ของ IC เบอร์ 74390



รูปที่ 3.24 แสดงการควบคุม DATA ของ IC เบอร์ 7485



รูปที่ 3.25 แสดงการควบคุม DATA ของ IC เบอร์ 74283



รูปที่ 3.26 แสดงการควบคุม DATA ของ IC เบอร์ 74169

การสร้างชุด DIGITAL TRAINER TEST BOARD

ส่วนประกอบของชุด DTH-TEST BOARD มีดังนี้

- SUPPLY +5 , +12V
- ชุด SW และ LED MONITOR
- ชุด กำเนิดสัญญาณสี่เหลี่ยม (CLOCK)
- ส่วนของ SLOT ในการส่งค่า DATA ของ IC
- BOARD อเนกประสงค์

SUPPLY +5 , +12V

เป็นส่วนที่จ่ายแรงดันขนาด 5V และ 12V ที่สามารถนำไปใช้ในการทดลองวงจรทางด้าน DIGITAL แรงดันที่สร้างขึ้นจะส่งมาจากชุด DIGITAL TRAINER HARDWARE (IC DATA BOX) โดยการใช้ IC REGULATOR 7805 และ 7812 ในการควบคุมระดับแรงดันไฟ

ชุด SW และ LED MONITOR

ชุด SW บน TEST BOAD มี 2 แบบ คือสวิทซ์ LOGIC ธรรมดา 4 ตัว และ SW แบบป้อง การบาว์ อีก 2 ตัว โดย SW ทั้ง 2 แบบ จะเป็นตัวสร้างระดับแรงดัน "0" และ "1" ที่ใช้ในการทดลอง IC ทางด้าน DIGITAL โดยจะมี LED MONITOR ที่สามารถใช้ในการแสดงผลอีก 4 ตัว

ชุดกำเนิดสัญญาณสี่เหลี่ยม

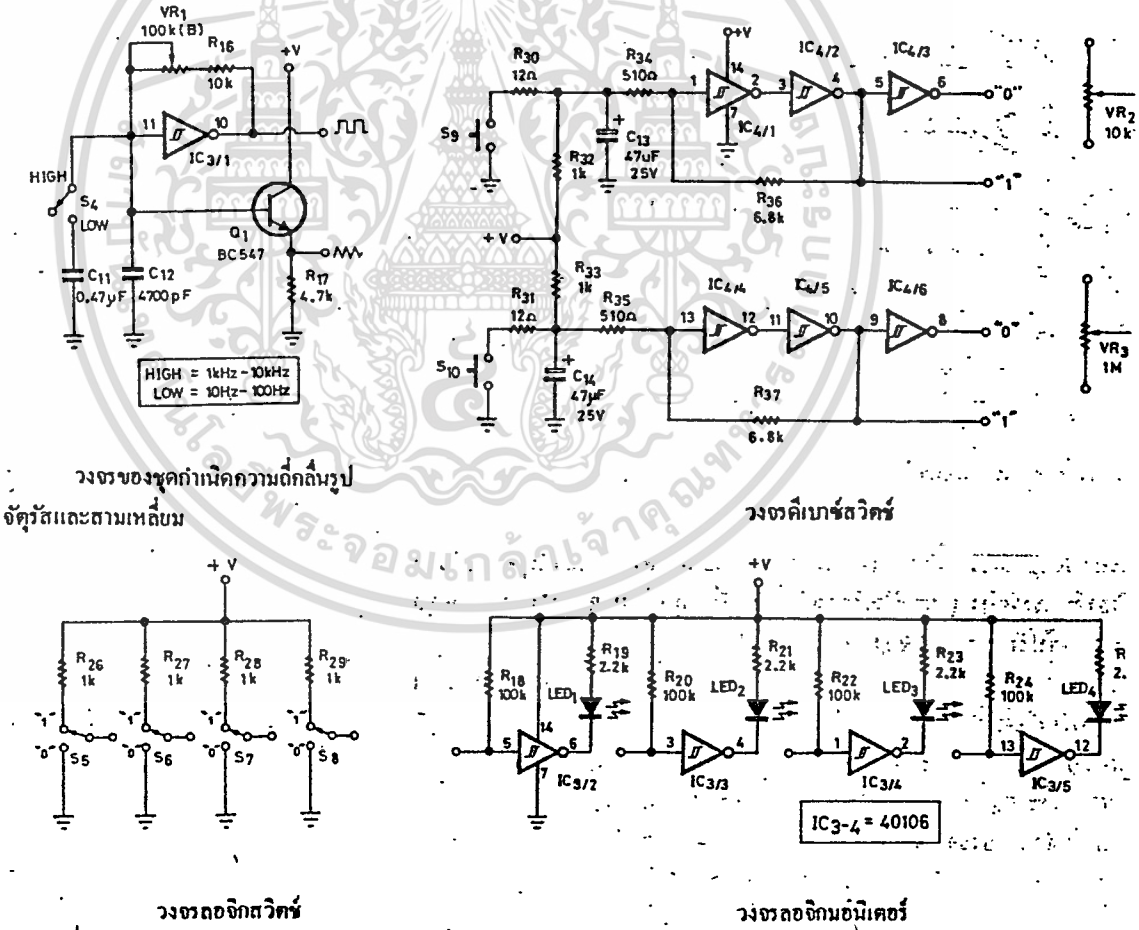
ชุดกำเนิดสัญญาณสี่เหลี่ยมนี้สามารถปรับความถี่ของ CLOCK ได้โดย VOLUME บน TEST BOARD ชุดกำเนิดสัญญาณสี่เหลี่ยมนี้เราสามารถจะนำไปใช้ในการทดลองวงจรประเภท COUNTER ได้หรือในการทดสอบวงจรประเภทอื่นๆ ก็ตาม

ส่วนของ SLOT ในการส่งค่า DATA ของ IC

SLOT ที่ส่งค่า DATA ตาม MENU จะมี 2 SLOT โดย SLOT ดังกล่าวจะเชื่อมต่อมาจากชุด IC DATA BLOCK เมื่อเราทำการเลือกเบอร์ IC เบอร์ใดตาม MENU เราสามารถใช้ DIP JUMPER ต่อที่ SLOT ดังกล่าวไบบน BOARD อเนกประสงค์เพื่อใช้ในการทดสอบได้เลยโดยอาศัยอุปกรณ์ ช่วยเหลือต่าง ๆ ที่มีอยู่บน DIGITAL TRAINER TEST BOARE

TEST BOARD

ในการทดลองวงจรเราสามารถอาศัย BOARD ดังกล่าวในการทดลองวงจรตามที่เราต้องการได้ โดย BOARD ดังกล่าวจะมีโครงสร้างตาม BOARD มาตราฐานทั่วไป



รูปที่ 3.27 แสดงวงจรต่าง ๆ ของ DIGITAL TRAINER TEST BOARD

บทที่ 4
การนำชุด DIGITAL TRAINER ไปใช้ในงาน

ในการใช้งานชุด DIGITAL TRAINER จะประกอบไปด้วย

1. แผน DISK ที่มี PROGRAM ชื่อ DTS_MAIN.EXE
2. ส่วนของ DIGITAL TRAINER HARDWAR
   - IC DATA BOX
   - TEST BOARD

ส่วนของ DTS_MAIN จะประกอบด้วย PROGRAM ต่าง ๆ ที่ได้กล่าวมาแล้วโดย PROGRAM ทั้งหมดจะรวมอยู่ใน PROGRAM หลักชื่อ DTS_MAIN ในส่วนของ DIGITAL TRAINER HARD WARE จะเป็นส่วนที่ใช้ในการศึกษาหรือทดลอง IC เบอร์ต่าง ๆ ตาม MENU ของ DTS_MAIN โดยสามารถทำการทดลองบนชุด DIGITAL TRAINER TEST BOARD

ขั้นตอนการใช้งาน DIGITAL TRAINER

1. ทำการเรียก PROGRAM ชื่อ DTS-MAIN

```
A> DTS_MAIN
```

เมื่อทำการเรียก PROGRAM ดังกล่าว บนจอภาพจะปรากฏคำว่า THE DIGITAL TRAINER

2. ทำการกด KEY ใด ๆ KEY หนึ่งเพื่อเข้าสู่ MENU ต่าง ๆ MENU ของชุด DTS_MAIN จะประกอบไปด้วย 4 MENU หลักโดย MENU 1 และ MENU 2 เป็นการใช้งานบน MICRO COMPUTER ส่วน MENU 3 และ MENU 4 เป็นส่วนที่เชื่อมโยงกับชุด DIGITAL TRAINER HARDWARE เพื่อใช้ในการทดลอง

2.1 INTRODUCTION MENU

เป็น MENU ที่ใช้ในการศึกษาค้นคว้าในการดูลายละเอียดเบื้องต้นของอุปกรณ์ทางด้าน DIGITAL พื้นฐานโดยจะประกอบไปด้วย การทำงาน TRUTH TABLE และสัญลักษ์โดย MENU 1 นี้จะมีตัวเลือกดังต่อไปนี้

```
'  INTRODUCTION MENU '
   ' OR_GATE     ',
   ' AND_GATE    ',
   ' NAND_GATE   ',
   ' NOR_GATE    ',
   ' NOT_GATE    ',
   ' EX-OR_GATE  ',
   ' EX-NOR_GATE',
   ' D_FIFLOP    ',
   ' SR_FIFLOP   ',
   ' JK_FIFLOP   ',
```

2.2 DATA PIN MENU

เป็น MENU ที่ใช้ในการศึกษาหรือดูลายละเอียดโครงสร้างภายในของ IC เบอร์ต่าง ๆ ประมาณ 20 เบอร์ ที่เป็น IC ที่ใช้ในการทดลองวงจรพื้นฐานทางด้าน DIGITAL โดยจะมีตัวเลือกดังต่อไปนี้

```
' #7400 { 2 INPUT NAND_GATE...} ',
' #7402 { 2 INPUT NOR_GATE....} ',
' #7404 { HEX INVERTER_GATE...} ',
' #7408 { 2 INPUT AND_GATE....} ',
' #7432 { 2 INPUT OR_GATE.....} ',
' #7474 { D_FLIP FLOP.........} ',
' #7486 { EX_OR_GATE..........} ',
' #74152{ 8 TO 1 LINE.........} ',
' #74164{ 8 BIT SHIFTREG......} ',
' #74393{ 4 BIT BINARY COUNT..} ',
' #7485 { 4 BIT COMPARATOR....} ',
' #74112{ JK_FLIP FLOP........} ',
' #74131{ 3TO8 DECODER........} ',
' #74147{ 10 TO 4 PRIORITY....} ',
' #74165{ 8 BIT SHIFT_PISO....} ',
```

```
' #74169{ UP DOWN DECADE COUNT} ',
' #74195{ 4 BIT SHIFT PIPO....} ',
' #74283{ 4 BIT FULL_ADDER....} ',
' #74390{ DECADE COUNTER......} ',
' #74447{ BCD TO 7 SEGMENT....} '
```

2.3 IC TESTER MENU

เป็น MENU ที่สัมพันธ์กับชุด DIGITAL TRAINER HARDWARE โดย MENU นี้จะประกอบด้วย IC ที่สามารถทำการทดสอบบน TEST BOARD ได้ 20 เบอร์ โดยสามารถเลือกซ้ำกันได้เบอร์ละ 2 ตัว

```
'   TEST IC MENU   '
' # 7400  ( 14_PIN ) ',
' # 7402  ( 14_PIN ) ',
' # 7404  ( 14_PIN ) ',
' # 7408  ( 14_PIN ) ',
' # 7432  ( 14_PIN ) ',
' # 7474  ( 14_PIN ) ',
' # 7486  ( 14_PIN ) ',
' # 74152 ( 14_PIN ) ',
' # 74164 ( 14_PIN ) ',
' # 74393 ( 14_PIN ) ',
' # 7485  ( 16_PIN ) ',
' # 74112 ( 16_PIN ) ',
' # 74131 ( 16_PIN ) ',
' # 74147 ( 16_PIN ) ',
' # 74165 ( 16_PIN ) ',
' # 74169 ( 16_PIN ) ',
' # 74195 ( 16_PIN ) ',
' # 74283 ( 16_PIN ) ',
' # 74390 ( 16_PIN ) ',
' # 74447 ( 16_PIN ) '
```

2.4 SELECT SLOT MENU

ในการเลือก IC จาก MENU 3 เราต้องทำการ SET MENU 4 เพื่อทำการเลือก SLOT ที่จะใช้งาน IC เบอร์ดังกล่าวด้วย ในการเลือกข้อมูลที่ถูกต้องจะได้ยินเสียงเพลงออกมา

```
' NUMBER SLOT# MENU '
' SELECT SLOT# NUMBER.. 1  ',
' SELECT SLOT# NUMBER.. 2  ',
```

การใช้งาน FUNCTION KEY ต่าง ๆ

FUNCTION KEY เป็น KEY พิเศษในการควบคุม PROGRAM โดยสามารถใช้ได้ถึง 10 FUNCTION คือ F1-F10 แต่ตอนนี้เราจะใช้เพียง 4 FUNCTION คือ

F2 - SET รูปแบบจอภาพแบบที่ 1
F3 - " " 2
F4 - " " 3
F10 - ใช้เพื่อออกจาก PROGRAM

การใช้ KEY CONTROL ต่าง ๆ

ชื่อ KEY	การใช้งาน
ลูกศรขึ้น	ใช้ในการเลื่อนแทบเลือกขึ้นข้างบน
" ลง	" " ลงข้างล่าง
" ซ้าย	ใช้ในการเลือกแทบ MENU ไปด้านซ้าย
" ขวา	" " ขวา
RETURN	ใช้สำหรับ RUN MENU และตัวเลือก

สรุปผลการทดลอง

สำหรับการทดลองชุด DIGITRAINER นั้นเราสามารถแบ่งการทดลองเป็น 2 ส่วนคือส่วนของ SOFT WARE และส่วนของ HARD WARE โดยในส่วนของ SOFT WARE นั้น เราสามารถทำการทดลองตามการใช้งานเพื่อการตรวจสอง FUNCTION การทำงานต่าง ๆ โดยเมื่อพบจุดบกพร่อง ก็ทำการกลับไปแก้ไขในส่วนของ PROGRAM และ COMPLIER ใหม่ โดยจะทำเช่นนี้ต่อไปจน PROGRAM สมบูรณ์ที่สุด

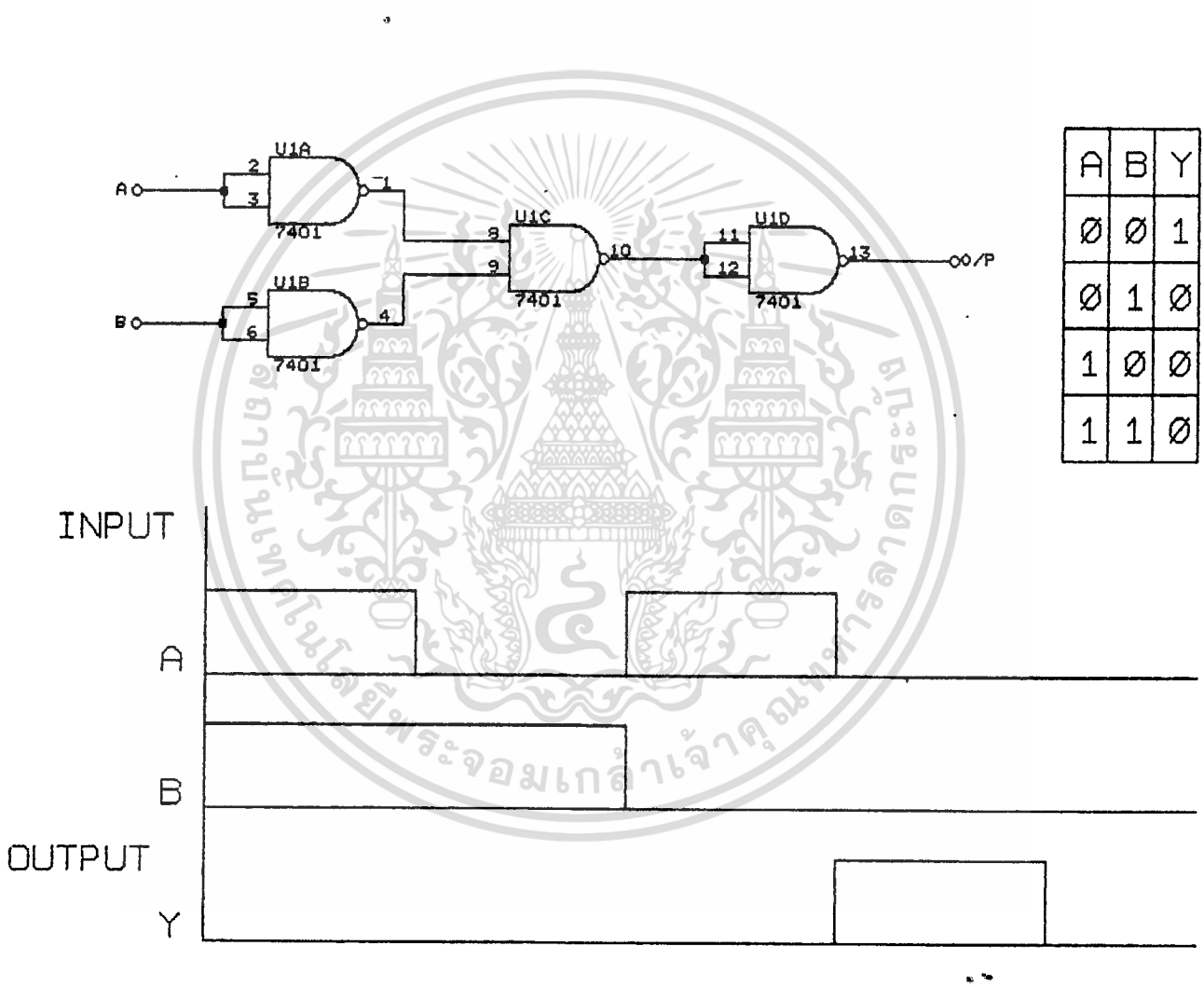
ในส่วนของ HARDWARD นั้นเราจะสามารถทำการทดลองได้หลายวิธี แต่วิธีที่สะดวกที่สุดและดีที่สุดโดยทดลอง IC แต่ละเบอร์ เช่น การทดลอง IC เบอร์ 7400 (NAND GATE) จะมีขั้นตอนดังนี้

1. เลื่อน MENU ไปที่ MENU 3 โดย KEY ลูกศรขวา
2. เลื่อนตัวเลือกใน MENU 3 โดย KEY ลูกศรลงหรือขึ้นตามตำแหน่งใน MENU
3. กด RETURN KEY หรือไป MENU 4 เลือก SLOT เช่น เลือก SLOT ที่ 1 จากนั้นการ RETURN KEY อีกทีจะได้ยินเสียงเพลงเบ็นอันว่า SLOT ที่ 1 จะเป็นเสมือน IC เบอร์ 7400 โดยขา 7 เป็น GND และขา 14 เป็น VCC
4. ต่อจากนั้นเขาก็ทำการต่อวงจรโดยใช้SWบน DIGITAL TRAINER TEST BOAD ในการทดสอบ เมื่อทำการทดสอบแล้วถ้าพบข้อผิดพลาดในส่วนของ HARDWARE เราจะทำการกลับไปตรวจสอบและแก้ไขใหม่จนสมบูรณ์

สำหรับ IC ที่เหลือเราก็ทำการทดสอบโดยวิธีดังกล่าวตามขั้นตอนที่กล่าวมาแล้ว

การทดลอง IC เบอร์ 7400 (NAND GATE)

เนื่องจาก IC เบอร์ 7400 1 ตัวประกอบด้วย IC NAND GATE 4 ตัว ดังนั้นในการทดลอง เพื่อตรวจสอบ สภาวะการทำงานของ IC ดังกล่าวจากชุด DIGITAL TRAINER ให้ได้ดีที่สุด โดยการนำ GATE ทั้ง 4 ตัวมาต่อเป็น IC NOR GATE และทำการตรวจสอบสภาวะการทำงานต่อไป



A	B	Y
Ø	Ø	1
Ø	1	Ø
1	Ø	Ø
1	1	Ø

รูปที่ 3.28 แสดงวงจรที่ใช้ในการทดลอง ตารางสภาวะและสัญญาณ

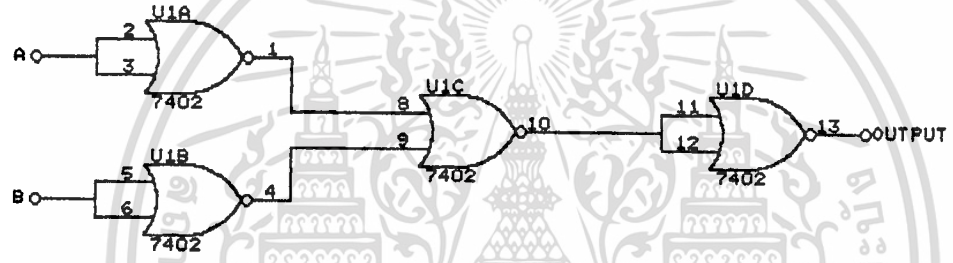
ขั้นตอนในการทดลอง

1. ทำการเลือกเบอร์ IC # 7400 และ SLOT # 1 จาก MENU 3,4 ตามลำดับ
2. ต่อวงจรตามวงจรที่ใช้ในการทดลอง
3. ป้อน IN PUT ที่ "A" "B" และตรวจสอบสภาวะ OUTPUT "Y" เทียบกับสภาวะมาตรฐาน

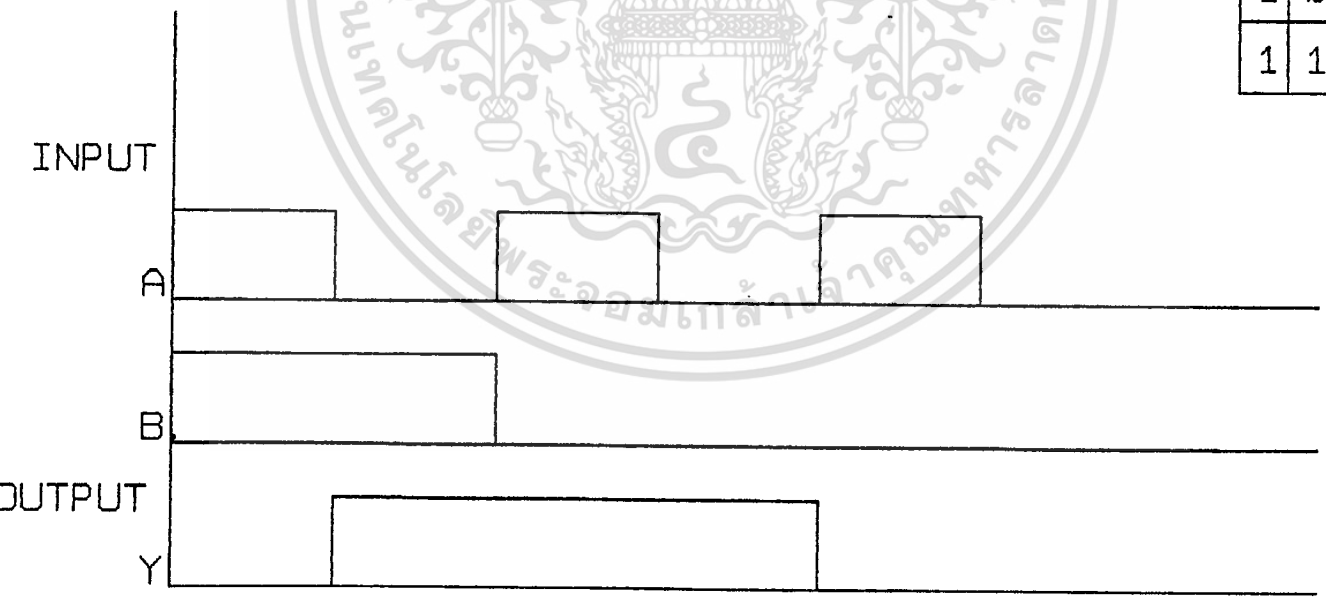
จากการทดลองวัดสภาวะ OUTPOT จากการป้อน INPUT A, B ปรากฏว่าได้สภาวะ OUTPUT "Y" ตรงตามสภาวะมาตรฐานทั่วไป

การทดลอง IC เบอร์ 7402 (NOR GATE)

การทดลอง IC เบอร์ 7402 โดยวงจรที่ใช้ในการทดลองเป็นการนำ IC NOR GATE 4 ตัว ของ IC เบอร์ 7402 มาต่อเป็น IC NAND GATE เพื่อใช้ในการตรวจสอบการทำงานของ IC ของชุด DIGITAL TRAINER



A	B	Y
Ø	Ø	1
Ø	1	1
1	Ø	1
1	1	Ø



รูปที่ 3.29 แสดงวงจรที่ใช้ในการทดลองตารางสภาวะและสัญญาณ

กิตติกรรมประกาศ

ในการสร้างชุด DIGITAL TRAINER นั้นสามารถทำได้สำเร็จตามขั้นตอนที่ได้วางไว้ทุกประการ โดยต้องขอขอบพระคุณ ผ.ศ. นิกร สุขุตมดันติ ที่ช่วยในการควบคุมและข้อเสนอแนะที่เป็นประโยชน์ อีกทั้งเพื่อน ๆ ทุกคนที่ให้การช่วยเหลือและหวังที่จะทำให้ชุด DIGITAL TRAINER มีประโยชน์สำหรับผู้ที่จะนำไปศึกษาต่อไป

หนังสืออ้างอิง

- เรียนรู้ภาษา PASCAL บุญเลิศ เอี่ยมทัศนา
- หลักการ เขียนโปรแกรมภาษาแอสแซมบัส 8088
  ดร. ศิริวรรณ ฉันทาดิศัย