การควบคุมโดยใช้มาตรฐาน (IEEE-488 GPIB)

THE CONTROLLING METHOD BY USING IEEE-488 (GPIB)

โดย

นายสมเกียรติ มายประเสริฐ

นายสาร อาชา

นายเสถียร ตรีทวีพรทรัพย์

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

การควบคุมโดยใช้มาตรฐาน (IEEE-488 GPIB)

THE CONTROLLING METHOD BY USING IEEE-488 (GPIB)

โดย

นายสมเกียรติ มายประเสริฐ รหัส 36013126

นายสาร อาชา รหัส 36023132

นายเสถียร ตรีทวีพรทรัพย์ รหัส 36013133

อาจารย์ที่ปรึกษา

รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมโดยใช้ IEEE-488

ผู้จัดทำ

1. นายสมเกียรติ มายประเสริฐ 36013126
2. นายสาร อาชา 36013132
3. นายเสถียร ตรีทวีพรทรัพย์ 36013133

.....

(รศ.ดร. มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

การควบคุมโดยใช้มาตรฐาน IEEE-488 (GPIB)

สมเกียตริ มายประเสริฐ
สาร อาชา
เสถียร ตรีทวีพรทรัพย์
รศ.ดร. มนัส สังวรศิลป์
ปีการศึกษา 2538

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ กล่าวถึงการควบคุมอุปกรณ์ต่างๆ โดยใช้มาตรฐาน IEEE 488 (GPIB) อันเป็นระบบมาตรฐานการส่งข้อมูลของ IEEE ซึ่งมีการใช้อย่างแพร่หลายในโรงงานอุตสาหกรรมในปัจจุบัน มาตรฐานการส่งข้อมูลแบบนี้ เป็นการรับหรือส่งข้อมูลระหว่าง ตัวควบคุม (Controller) กับอุปกรณ์ที่ทำหน้าที่เป็นตัวส่ง (Talker) หรือ เป็นตัวรับข้อมูลนั้น(Listener) โดยใช้การรับส่งข้อมูลแบบขนาน 8 bits โดยในปริญญานิพนธ์ฉบับนี้ ได้ทำการเขียนโปรแกรมภาษา C ติดต่อสื่อสารสารกับอุปกรณ์เครื่องมือวัด Function Generator, Power Supply, Multimetre และ Osilloscope รวมทั้งเสนอการประยุกต์ใช้งาน โดยการควบคุมให้ Function Generator และ Power Supply เพื่อป้อนสัญญาณและไฟเลี้ยงให้แก่วงจรที่ใช้ในการวัดหากราฟคุณลักษณะ(Charecteristic)ของ ทรานซิสเตอร์และมอสเฟส ที่ใช้ตรวจสอบ โดยกราฟที่ได้จะแสดงผลบนจอ Osilloscope และในเวลาเดียวกันสัญญาณที่ได้จะถูกสุ่มตัวอย่าง(Sampling)เพื่อนำไปแสดงผลบนจอคอมพิวเตอร์เช่นเดียวกัน

The Controlling Method by Using IEEE-488 (GPIB)

Mr. Somkiat Maiprasert
Mr. Sarn Archa
Mr. Sathien Treetaweepornsup
Assoc. Dr.Sungwarasin Manus
Adviser 1995

ABSTRACT

This thesis concerns about the controlling method by using IEEE-488 (GPIB). It is a standard system for transfering data. By sending or receiving information between Controller(PC) and Talker/Listener in parallel 8 bit of databus

In this thesis IEEE -488 is useto control Function generator, Power Supply and Osilloscope .In addition to present the application with IEEE-488 for finding Characteristic curve of transister and MOS transistor. By use funtion generator and power supply to control test signal, current and voltage of test circuit respectively. And use Osilloscope to show characteristic curve of test transistor and test MOSFET. In the same times, sampling signal are showed on PC monitor by controller(PC)

สารบัญ

สารบัญรูป

บทนำ

การควบคุมเครื่องมือวัดด้วย IEEE-488 (GPIB)

ระบบอุตสาหกรรมในอดีตมักใช้อุปกรณ์ไม่มากชิ้น หากมีการเชื่อมต่อก็สามารถทำได้โดยง่าย ต่อมาการอุตสาหกรรมได้เจริญรุดหน้าไปอย่างรวดเร็ว ดังนั้นระบบอุตสาหกรรมจึงมีขนาดใหญ่และ ซับซ้อนขึ้นด้วยการเชื่อมต่อระหว่างอุปกรณ์หลาย ๆ ชิ้นจะต้องเสียค่าใช้จ่ายจำนวนมากไปด้วย หากจะมีการเชื่อมต่อระหว่างอุปกรณ์หลายๆ ชิ้น จะต้องเสียค่าใช้จ่ายจำนวนมาก ดังนั้นจึงมีการคิดค้นระบบเชื่อมต่อ ซึ่งเป็นมาตรฐานขึ้นมานั้นคือ IEEE-488 นั่นเอง

ประวัติการพัฒนา IEEE-488

ดังที่ได้กล่าวมาข้างต้นว่า การเชื่อมต่อทางอุตสาหกรรมในอดีต เป็นไปด้วยความยากลำบากและเสียค่าใช้จ่ายมากดังนั้นบริษัทผู้ผลิตเครื่องมือวัดต่างๆในประเทศสหรัฐอเมริกาจึงร่วมกันจัดหาระบบเชื่อมต่อ มาตรฐานขึ้นมาซึ่งในประเทศเยอรมันก็มีการพัฒนาระบบเชื่อมต่อมาตรฐานเช่นกัน โดยความร่วมมือของ IEC (International Eletrotechnical Commission) จนในกระทั่งใน ปี 1972 สหรัฐอเมริกาโดยการนำของ IEEE (Instiute of electrical and Electronics Enginners IEEE) จึงได้มีการประชุมเพื่อวางแผนพิจารณาระบบเชื่อมต่อมาตรฐามร่วมกัน

บริษัทฮิวเล็ตแพกการ์ดผู้ผลิตเครื่องมือวัดรายใหญ่ในอเมริกาได้ทำการพัฒนาระบบเชื่อมต่อมาตรฐานอยู่ก่อนแล้วชื่อว่า HPIB (Hewwlett Packard Interface Bus) จึงได้เสนอโครงการให้ IEEE เพื่อพิจารณา และได้รับการยอมรับในปี 1975 โดย IEEE จัดให้เป็นมาตรฐานลำดับที่ 488 ดังนั้นจึงได้ชื่อว่า IEEE-Std 488-1975 ซึ่งต่อมาได้มีการปรับปรุงเป็น IEEE-Std 488-1978 หรือที่นิยมเรียกกันว่า GPIB (Generral Purpose Interface Bus)

บทที่ 1

IEEE-488 (GPIB)

1.1 โครงสร้างของ IEEE-488

ในระบบพื้นฐานของGPIB จะประกอบด้วยอุปกรณ์คือผู้ส่ง (Talker) ,ผู้รับ (Listener) และผู้ควบคุม (Controller)

- Talker ทำหน้าที่ส่งข้อมูลโดยในระบบสามารถมี Talker ได้หลายตัวแต่จะมีเพียงตัวเดียวเท่านั้นที่กำลังทำงานอยู่

- Listener ทำหน้าที่เป็นตัวรับข้อมูลโดยในระบบเดียวกันสามารถมี Listenerได้ หลายตัวเช่นกัน แต่ Listener สามารถทำได้ครั้งละหลายๆ ตัวได้

- Controller ทำหน้าที่ควบคุมอุปกรณ์ต่างๆ ในระบบโดยจะกำหนดให้ Talker ทำการส่งข้อมูลหรือกำหนดให้ Listener ทำการรับข้อมูล

อุปกรณ์ที่มี GPIB นั้นสามารถแบ่งตามหน้าที่ได้ดังนี้

1. ทำหน้าที่เป็น Talker เท่านั้นเช่น เครื่องมือวัด เป็นต้น

2. ทำหน้าที่เป็น Listener เท่านั้นเช่น เครื่องพิมพ์ (Printer), เครื่องบันทึก (Recorder) เป็นต้น

3. ทำหน้าที่เป็นทั้ง Talker และ Listener เช่น คอมพิวเตอร์ , เครื่องมือวัดที่สามารถควบคุมได้จากภายนอกเป็นต้น

4. ทำหน้าที่เป็น Talker Listener และ Controller ในตัวเดียวกันเช่น คอมพิวเตอร์ที่ทำหน้าที่ควบคุมระบบ

1.2 ขีดจำกัดของ IEEE-488

1. จำนวนอุปกรณ์ในระบบ (Talker, Listener, Controller) ที่ต่อกับสายสัญญาณ 1 เส้นจะต้องไม่เกิน 15 เครื่อง

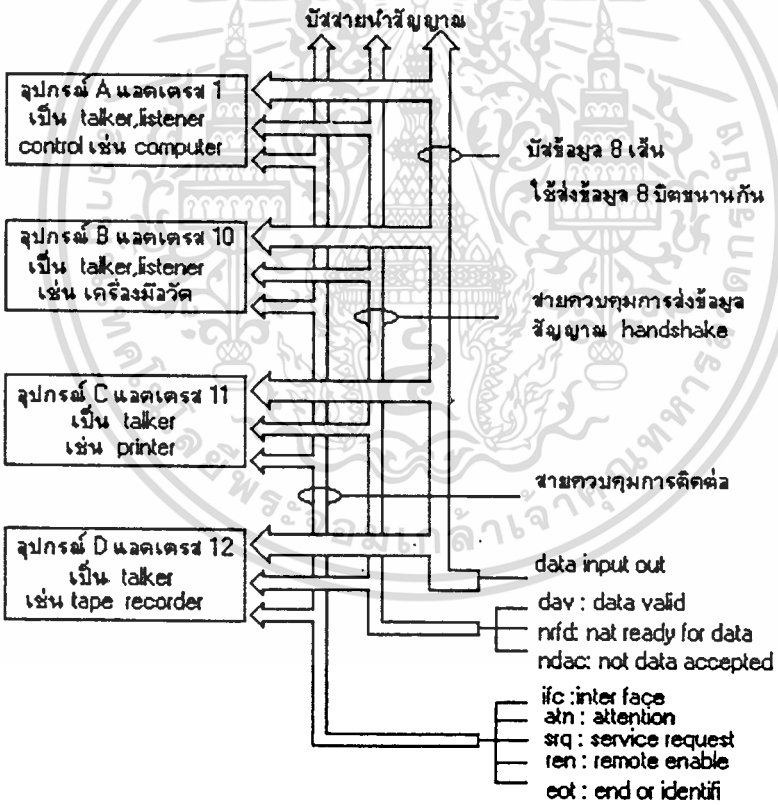
2. สายเคเบิ้ลที่ต่อระหว่างอุปกรณ์จะต้องยาวไม่เกิน 4 เมตร และความยาวรวมของสายเคเบิ้ลในระบบจะต้องไม่เกิน 20 เมตร

3. ความยาวในการส่งข้อมูลจะต้องไม่เกิน 1Mb/Sec (1 ล้านไบต์ต่อหนึ่งวินาที)

4. จะต้องมีการจ่ายไฟให้กับอุปกรณ์มากกว่าครึ่งหนึ่งของระบบ

1.3 รายละเอียดเกี่ยวกับ IEEE-488

ลักษณะทางกายภาพของ IEEE-488 นั้น คือเป็นสายสัญญาณแบบ 24 เส้น ขนานกันและมีขั้วต่ออยู่ทางปลายทั้งสองของสาย เพื่อต่อกับอุปกรณ์หรือต่อกันเพื่อให้สายสัญญาณมีความยาวเพิ่มขึ้นจำนวนสายสัญญาณ 24 เส้นนี้ มีเพียง 16 เส้นเท่านั้นที่ทำหน้าที่นำสัญญาณ ส่วนที่เหลืออีก 8 เส้นทำหน้าที่กราวด์ (ground) และชิลด์ (shield)



รูปที่ 1.1 แสดงการแบ่งเส้นสายนำสัญญาณ

โดยจำนวนสายที่ใช้นำสัญญาณ 16 เส้นนั้น ยังแบ่งได้เป็น 3 ประเภทตามรูปที่ 1.1 คือ

1. บัสข้อมูล (Data Bus) จำนวน 8 สาย คือ

   DI01-DI08

2. สายสัญญาณควบคุม (Control Line) จำนวน 5 สายคือ

   IFC (Interface Clear)

   ATN (Attention)

   SRQ (Service Request)

   REN (Remote Enable)

   EOI (End Or Identify)

3. สายแฮนด์เชค (HAND SHAKE) 3 สายคือ

   DAV (Data Valid)

   NRFD (Not Ready For Data)

   NDAC (Not Data Accepted)

1.4 ความหมายของสัญญาณต่างๆ ภายใน IEEE-488

ดังที่ได้กล่าวมาแล้วว่าสายสัญญาณ ต่างๆ ใน GPIB ได้แบ่งออกเป็น 3 กลุ่มในหัวข้อนี้จะอธิบายความหมายของสัญญาณต่างๆ ได้ดังนี้

กลุ่มสัญญาณข้อมูล

1. DI01-DI08 (Data Input/Output) สายสัญญาณทั้ง 8 เส้นนี้ทำหน้าที่เป็นทางผ่านของข้อมูลในระบบ

กลุ่มสัญญาณควบคุมการเชื่อมต่อ (Interface)

1. IFC (Interface clear) เป็นสัญญาณรีเซต หรือเคลียร์ระบบกำเนิดได้โดยตัวควบคุม หรือเคลียร์ระบบ กำเนิดได้โดยตัวควบคุม (Controller) เท่านั้นเมื่ออุปกรณ์ในบัสได้รับสัญญาณเคลียร์นี้จะกลับคืนสู่สภาวะเริ่มต้นใหม่ ซึ่งเป็นสภาวะเริ่มแรกก่อนการกำหนดฟังก์ชั่น หมือนแรกเปิดสวิตช์

2. ANT (Attention) เป็นสัญญาณที่ถูกส่งโดยอุปกรณ์ที่เป็นตัวควบคุมเช่นเดียวกันใช้ในการสั่งให้อุปกรณ์ ทุกตัวในระบบเตรียมพร้อมเพื่อรอรับคำสั่งต่อไป

3. SRQ (Service Request) เป็นสัญญาณที่ถูกส่งจากอุปกรณ์ต่างๆ เพื่อเป็นการบอกแก่ระบบว่าขณะนี้อุปกรณ์ดังกล่าวต้องการติดต่อจากตัวควบคุม

4. REN (Remote Enable) สัญญาณนี้เป็นสัญญาณที่ถูกส่งมาจากตัวควบคุมเพียงตัวเดียวเท่านั้นเพื่อใช้สั่ง ให้อุปกรณ์ต่างๆ เปลี่ยนจากโหมดที่ใช้งานปกติมาเป็นการควบคุม โดยตัวควบคุมแทน

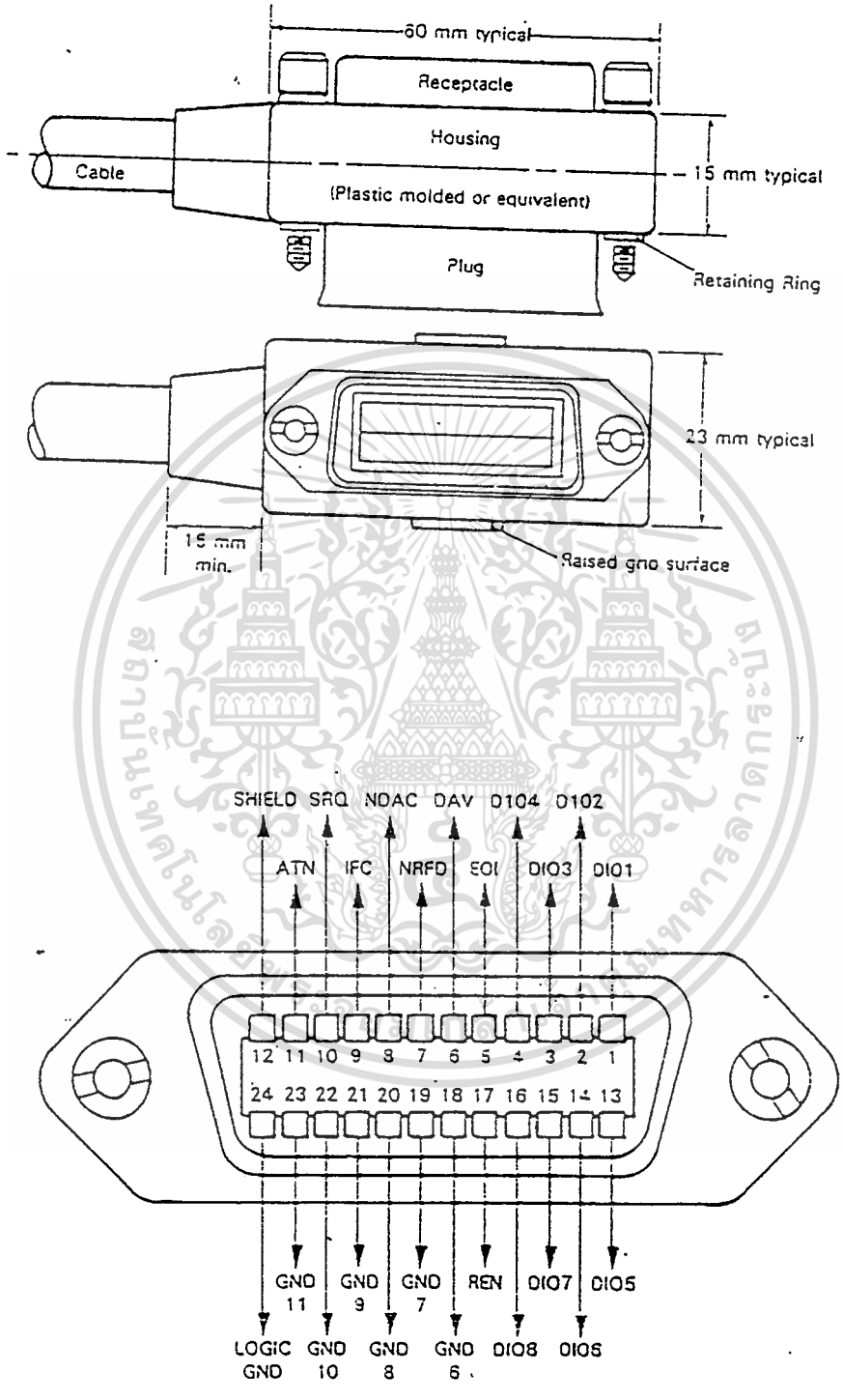
5. EOI (End Or Identify) เป็นสัญญาณที่ถูกส่งได้ โดยอุปกรณ์ที่เป็นตัวควบคุม หรืออุปกรณ์ที่เป็นตัวส่ง (Talker) ก็ได้ ใช้สำหรับแสดงว่าข่าวสารที่ส่งเป็นชุดนั้นได้เสร็จสิ้นลงแล้ว

กลุ่มสัญญาณควบคุมการรับ- ส่งข้อมูล

1. DAV (Data Valid) เมื่อสัญญาณนี้ถูกดึงเป็นลอจิก "Low" โดยอุปกรณ์ที่เป็นตัวส่งเป็นการแจ้งแก่ระบบบัสว่าขณะนี้ตัวส่งได้ทำการส่งข้อมูล ลงไปที่สายสัญญาณข้อมูลเรียบร้อยแล้ว

2. NRFD (Not Ready For Data) เมื่อสัญญาณนี้มีลอจิกเป็น "Low" จะเป็นการแสดงว่าในขณะนี้ระบบบัสยังไม่พร้อมที่จะรับข้อมูล เนื่องจากอุปกรณ์ในระบบยังพร้อมไม่หมดทุกตัว ซึ่งสัญญาณเส้นนี้จะไม่เป็น "Hi" จนอุปกรณ์ทุกตัวให้ลอจิกที่เป็น "Hi" ครบถ้วนแล้วสัญญาณนี้มีประโยชน์ในกรณีที่อุปกรณ์ในระบบมีความเร็วแตกต่างกัน

3. NDAC (Not Data Accepted) สัญญาณเส้นนี้เป็นสัญญาณที่ถูกควบคุมโดยอุปกรณ์ที่เป็นตัวรับ(Listener)โดยสัญญาณนี้จะมีลอจิกเป็น"Hi"ในขณะที่อุปกรณ์ที่เป็นตัวรับกำลังเก็บข้อมูลจากสายข้อมูล(Data Bus) และจะเป็น"Hi"เมื่ออุปกรณ์นั้นได้ทำการอ่านข้อมูลเสร็จเรียบร้อยแล้วโดยสัญญาณลอจิกที่ใช้ใน DATABUS (D1-D8) ของ IEEE-488 นี้มีลักษณะเป็นคอมพลีเมนต์ทั้งหมดคือ "1" เท่ากับ "Low" และ "0" เท่ากับ "Hi" ซึ่งตรงข้ามกับในวงจรที่เราคุ้นเคย

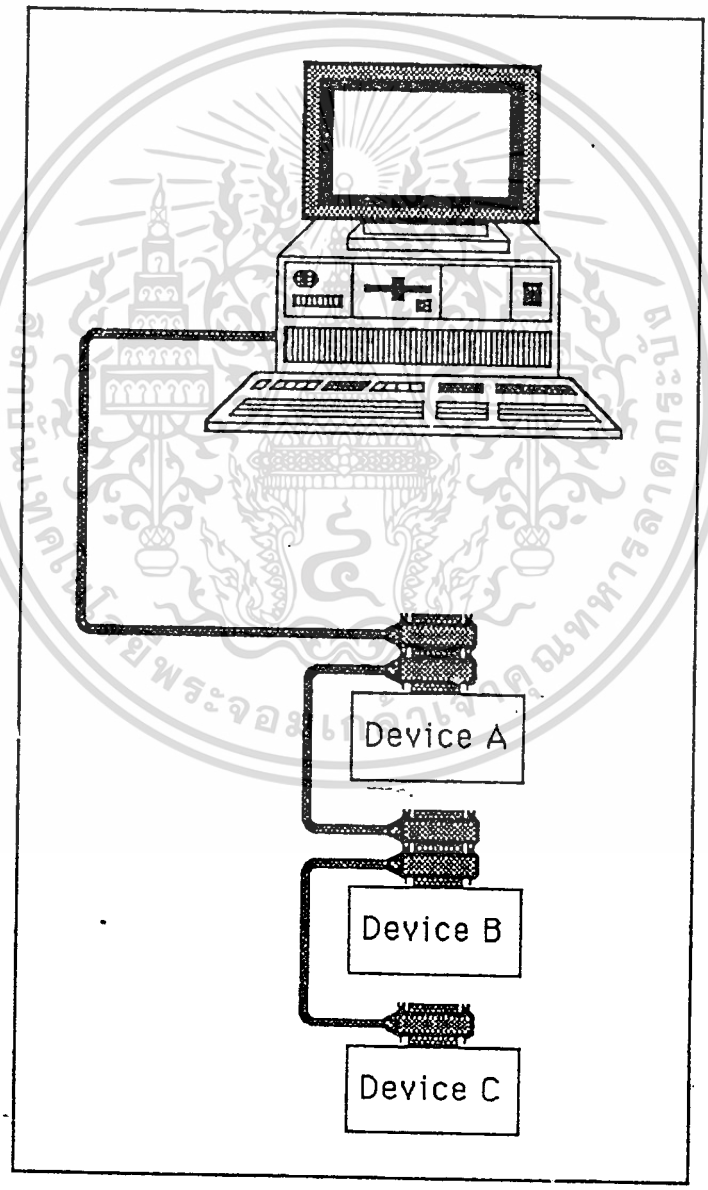


รูปที่ 1.2 ขั้วต่อของ GPIB และการจัดขาของสัญญาณต่างๆ

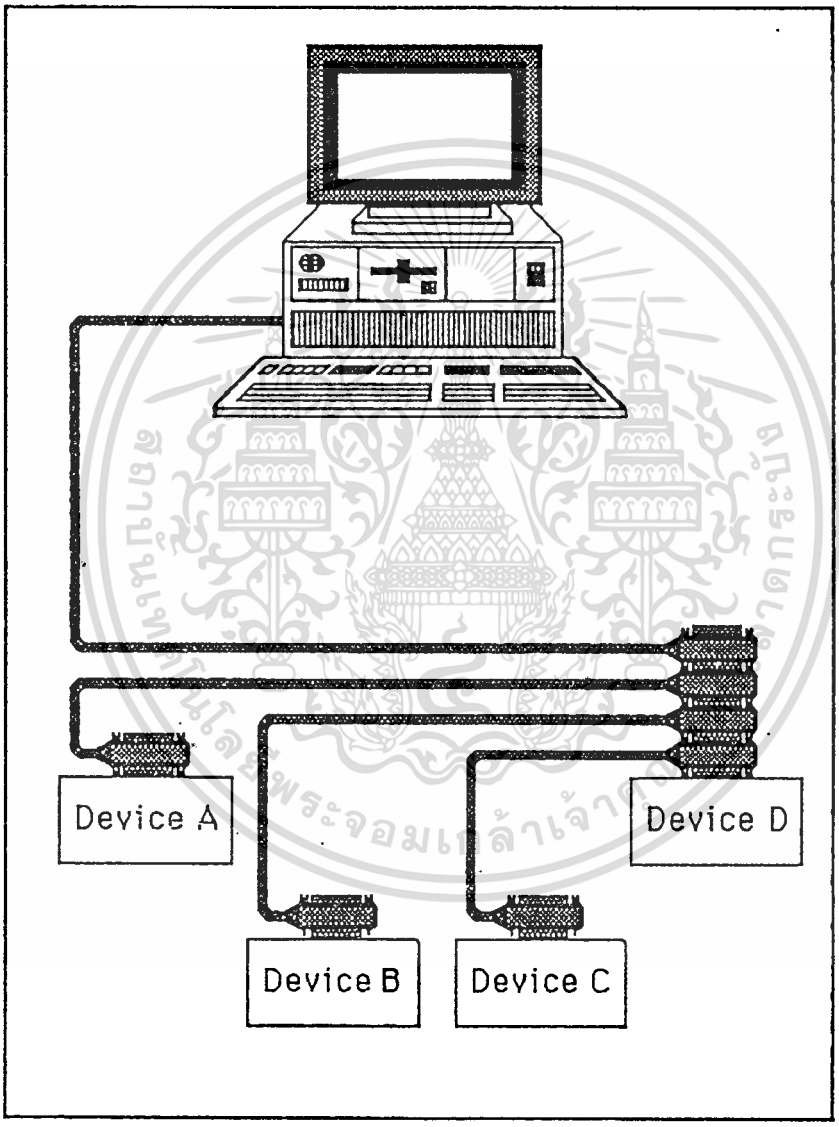
1.5 การเชื่อมต่ออุปกรณ์ต่าง ๆ ในระบบ IEEE-488

สำหรับการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ในระบบ IEEE-488 นั้นมีอยู่ 2 วิธีคือ

1. การเชื่อมต่อแบบเรียงต่อเนื่องกัน (Daisy Chain Configuration)
2. การเชื่อมต่อแบบกระจาย (Star Configuration)



รูปที่ 1.3 การเชื่อมต่อแบบเรียงต่อเนื่องกัน (Daisy Chain Configuration)



รูปที่ 1.4 การเชื่อมต่อแบบกระจาย (Star Configuration)

1.6 คำสั่งใช้งานของ GPIB

การสั่งการต่างๆเพื่อกำหนดหน้าที่การทำงานและกำหนดฟังก์ชั่น เช่นกำหนดช่วงการวัด โหมดการวัด หรืออื่น ๆ แก่เครื่องวัดที่ต่ออยู่ เหล่านี้นั้น ตัวควบคุมจะเป็นตัวกำหนดโดยการส่งรหัสคำสั่งไปที่อุปกรณ์โดยผ่าน DI1-DI6 รหัสคำสั่งนี้จะถูกส่งไปในช่วงที่สายสัญญาณ ATN เป็น LOW

คำสั่งสำหรับกำหนดหน้าที่การทำงานต่าง ๆ ตามมาตรฐานของ GPIB มีอยู่ด้วยกัน 128 คำสั่ง ดังแสดงใน ตารางที่ 1.1 ต่อไปนี้โดยแบ่งเป็น 5 กลุ่มคำสั่ง

รหัสที่ใช้ในระบบ GPIB บัสนั้นใช้ร่วมกันทั้งรหัสข้อมูล และรหัสคำสั่ง นั้นคือรหัสเดียวกันมีความหมายได้ 2 อย่างคือเมื่อ ATN เป็นLOW จะหมายถึงรหัสคำสั่ง แต่ถ้า ATN เป็น HIGH รหัสนี้จะแทนข้อมูลที่เป็น ASCII แทน ซึ่งในตารางก็ได้แบ่งความหมายออกเป็น 2 คอลัมน์

ASCII — IEEE 488 BUS MESSAGES (COMMANDS AND ADDRESSES) HEX CODES

MSD / LSD	0		1		2		3		4		5		6		7	
	ASCII	MSG	ASCII	MSG	ASCII	MSG¹	ASCII	MSG¹	ASCII	MSG¹	ASCII	MSG¹	ASCII	MSG	ASCII	MSG
0	NUL		DLE		SP	00	0	16	@	00	P	16	`		p	
1	SOH	GTL	DC1	LLO	!	01	1	17	A	01	Q	17	a		q	
2	STX		DC2		"	02	2	18	B	02	R	18	b		r	
3	ETX		DC3		#	03	3	19	C	03	S	19	c		s	
4	EOT	SDC	DC4	DCL	$	04	4	20	D	04	T	20	d		t	
5	ENQ	PPC	NAK	PPU	%	05	5	21	E	05	U	21	e		u	
6	ACK		SYN		&	06	6	22	F	06	V	22	f		v	
7	BEL		ETB		'	07	7	23	G	07	W	23	g		w	
8	BS	GET	CAN	SPE	(	08	8	24	H	08	X	24	h		x	
9	HT	TCT	EM	SPD	)	09	9	25	I	09	Y	25	i		y	
A	LF		SUB		*	10	:	26	J	10	Z	25	j		z	
B	VT		ESC		+	11	;	27	K	11	[	27	k		{	
C	FF		FS		,	12	<	28	L	12	\	28	l		|	
D	CR		GS		-	13	=	29	M	13	]	29	m		}	
E	SO		RS		.	14	>	30	N	14	^	30	n		~	
F	SI		US		/	15	?	UNL	O	15	_	UNT	o		DEL	

MEANING DEFINED BY PCG CODE

MEANING DEFINED BY PCG CODE

ADDRESSED COMMAND GROUP | UNIVERSAL COMMAND GROUP | LISTEN ADDRESS GROUP | TALK ADDRESS GROUP | SECONDARY COMMAND GROUP

PRIMARY COMMAND GROUP (PCG)

ตารางที่ 1.1 แสดงกลุ่มคำสั่ง

1. คำสั่งเจาะจงจุดหมาย (addressed command group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ที่เป็นตัวส่งหรือตัวรับ ที่กำหนดไว้ล่วงหน้าแล้วคำสั่งนี้ประกอบไปด้วย

- GTL (got to local) สั่งให้อุปกรณ์กลับคืนสู่สภาพการควบคุมปกติด้วยมือ
- SDC (selected device clear) สั่งให้อุปกรณ์เคลียร์ตัวเองสู่สภาพเริ่มต้นใหม่
- PPC (paralled poll configure) เป็นคำสั่งสำหรับทำการจัดสายสัญญาณ ของการทำการจัดสรรสายสัญญาณ ของการทำกระบวนการตรวจสอบสภาพอุปกรณ์ โดยวิธีขนานโดยใช้กำกับกลุ่มคำสั่งรอง
- GET (group excute trigger) ใช้สั่งเริ่มต้นการทำงานของอุปกรณ์ทีละหลายตัว
- TCT (take control) เป็นการกำหนดให้อุปกรณ์ตัวส่งทำหน้าที่เป็นตัวควบคุม

2. กลุ่มคำสั่งครอบคุม (universal command group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ทุกตัวที่อยู่ในบัส ประกอบด้วย

- LLO (local lockout) เป็นการสั่งให้อุปกรณ์ล็อกอยู่ที่สภาวะควบคุมโดยปุ่มปรับที่หน้าปัดตามปกติ
- DCL (devide clear) สั่งให้อุปกรณ์ทุกตัวกลับไปสู่สภาวะเริ่มต้น
- PPU (parallel poll unconfigure) ใช้ยกเลิกขบวนการตรวจ สอบสภาพแบบขนานทั้งหมด
- SPE (serial poll enable) เปลี่ยนโหมดของการตรวจสอบสภาพเป็นแบบอนุกรมในโหมดนี้จะเป็นการส่งสถานะของเครื่องแทนการส่งข้อมูล
- SPD (serial poll disable) ยกเลิกโหมดการตรวจสอบแบบอนุกรม

3. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (listener address group) เป็นคำสั่งสำหรับกำหนดให้อุปกรณ์เป็นตัวรับ ตามรหัสหมายเลขจาก 0-30 และมีคำสั่ง UNT (untalker) สำหรับยกเลิก

4. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวส่ง (talker address group) สำหรับกำหดให้อุปกรณ์เป็นตัวส่งตามรหัสหมายเลข 0-30 และมีคำสั่ง UNT (unlisten) สำหรับยกเลิกเช่นกัน

5. กลุ่มคำสั่งรอง (secondary command group) เป็นคำสั่งที่กำหนดรายละเอียดย่อยของอุปกรณ์แต่ละตัวที่ต่ออยู่ในระบบ ให้มีการทำงานอย่างไรตามจุดประสงค์ใช้งานของเครื่องมือนั้น

เช่นเดียวกับการปรับปุ่มต่างๆ ด้วยมือนั่นเอง คำสั่งรองนี้จะตามหลังคำสั่งหลักคือ จะใช้หลังจากอุปกรณ์ต่าง ๆ ถูกกำหนดวางตัวในระบบเรียบร้อยแล้ว

ค~

กำหนดไปนั้นเป็นอย่างไร และมีจุดประสงค์เพื่ออะไรดังต่อนี้

Device clear/Interface Clear

Device clear ใช้ในการทำให้อุปกรณ์ที่ต่ออยู่ในบัสกลับไปสู่สภาวะเริ่มต้น ยังไม่มีการกำหนดฟังก์ชั่นใดๆ สภาวะเริ่มต้นนี้ จะแตกต่างกันไปแล้วแต่ว่าอุปกรณ์นั้นออกแบบไว้อย่างไร Device clear มี 2 ลักษณะคือ เคลียร์หมดทุกตัวที่ต่ออยู่ (DCL) กับเคลียร์เจาะจงอุปกรณ์ตัวใดตัวหนึ่ง (SDC)

แต่ว่าในการเคลียร์อุปกรณ์ให้อยู่ในสภาวะเริ่มต้น นั้นไม่ได้หมายความว่า Interface function ของ GPIB จะถูกเคลียร์อุปกรณ์ให้ไปอยู่ในสภาวะเริ่มต้นด้วย แต่อย่างใด interface function คือสภาพการ interface ที่ได้กำหนดไว้ในระบบประกอบด้วยฟังก์ชั่นต่างๆ ดังแสดงในตารางที่ 2

ฟังชั่น	สัญญลักษณ์	การกลับสู่จุดเริ่มต้น IFC
source handshake	SH	ได้
acceptor handshake	AH	ได้
talker or enlarge talker	TO or TE	ได้
listener or enlarge listener	L or LT	ได้
service erquest	SR	ไม่ได้
remote/local	RL	ไม่ได้
parallel poll	PP	ไม่ได้
device clear	DC	ได้
device trigger	DT	ได้
controller	C	ได้

ตารางที่ 1.2 แสดงฟังก์ชั่นต่างๆในการ เคลียร์อุปกรณ์

Remote/Local

remote เป็นการกำหนดให้อุปกรณ์ที่อยู่ในระบบเช่นเครื่องมือวัดให้อยู่ในการ ควบคุมของอุปกรณ์ตัวอื่นแทนซึ่งปุ่มปรับต่างๆ บนหน้าปัดเครื่องจะไม่มีผลต่อการทำงาน ส่วน local เป็นการควบคุมการทำงานของเครื่องมือวัดด้วยปุ่มปรับบนหน้าปัดปกติ

การใช้ remote มีประโยชน์ในแง่ที่ขณะที่ตัวการควบคุมเช่น คอมพิวเตอร์กำลังติดต่ออุปกรณ์ตัวนั้นอยู่ หากไม่มีการตัดการควบคุมโดยปุ่มปรับบนหน้าปัดออก ถ้ามีใครมาปรับแต่งก็จะทำให้การทำงานผิดพลาดไปได้ การทำงานของ GPIB ใน remote and local มี 4 ลักษณะดังนี้

1. LOCS ก็คือ local นั้นเอง เป็นสภาพการควบคุมที่ปุ่มตามปกติ จะอยู่ในสภาพนี้ตอนเปิดเครื่องหรือ REN เป็น HIGH หรือเมื่อได้รับคำสั่ง GTL

2. REMS คือ remote หมายถึงการตัดการควบคุมโดยปุ่มหน้าปัดออก จะเกิดขึ้นเมื่อ REN เป็น LOW และจะถูกล๊อกไว้ เว้นแต่ว่าสวิตช์ local ที่ตัวอุปกรณ์จะถูกเปลี่ยนไปตำแหน่ง local

3. RWLS เป็นสภาพ remote ที่ถูกล๊อกเอาไว้เช่นกัน แต่การตัดการควบคุมตรงสวิตช์ local ที่ตัวอุปกรณ์ออกไปสภาพ remote โดย RWLS จึงมีความสำคัญสูงกว่า REMS อย่างไรก็ตามยังถูกยกเลิกได้ด้วยคำสั่ง LLO

4. LWLS มีสภาพเช่นเดียวกับ local แต่จะแตกต่างกันตรงที่สภาพ local โดย LWLS นี้ เมื่อได้รับคำสั่งกำหนดอุปกรณ์ตัวรับจะเปลี่ยนไปอยู่ในสภาพแบบล๊อกหรือ RWLS ทันทีในการที่จะมาที่สภาพ LWLS นี้ ได้ก็มี 2 กรณีคือ เมื่ออยู่ในสภาพ local ธรรมดา (LOCS) แล้วได้รับคำสั่ง LLO หรืออยู่ใน RWLS แล้วได้รับคำสั่ง GTL

1.7 การขอบริการและการตรวจสอบ (Service Request and Polling)

เมื่อตัวควบคุมได้รับ SRQ เป็น LOW จะให้อุปกรณ์ส่งข้อมูลแสดงสถานะการทำงานซึ่งมีอยู่ 2 วิธีคือ

1. การตรวจสอบแบบอนุกรมซึ่งมีขั้นตอนดังนี้

1.1 ATN ถูกดึงเป็น LOW หลังจากได้รับ LOW จากสายสัญญาณ SRQ

1.2 คำสั่ง UNL ถูกสั่งไปยังอุปกรณ์

1.3 ตัวควบคุมจะแจ้งรหัสตัวรับของตน และกำหนดรหัสตัวส่งของอุปกรณ์ที่จะตรวจสอบไปที่บัส

1.4 ตามด้วยคำสั่ง SPE และสาย ATN กลายเป็น HI ซึ่งอุปกรณ์ที่ถูกเรียกจะส่งข้อมูลแสดงสถานะออกมา 1 ไบต์โดยบิตที่ 7 จะเป็นตัวชี้ว่าอุปกรณ์เส้นเป็นตัวขอบริการ ถ้าจัดเป็นส่วนบิตอื่น ๆ ก็ใช้บอกข้อมูลอื่นๆ ซึ่งมิได้กำหนดเฉพาะ

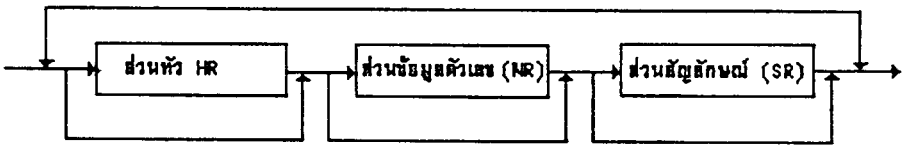
1.5 สาย ATN ถูกดึงเป็น LOW อีกที เพื่อส่งคำสั่งยกเลิกการตวจสอบคือ SPD

1.6 จากนั้นคำสั่ง UNT ถูกส่งไปยังอุปกรณ์เพื่อยกเลิกการเป็นตัวส่งซึ่งถ้าหาก SQR ยังคงเป็น LOW จะมีการตรวจสอบไปยังอุปกรณ์ตัวอื่นๆ ต่อไป ตามขั้นตอนเดิม

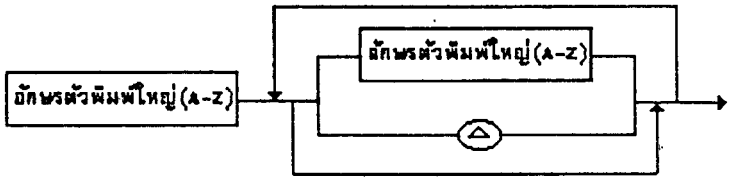
2. การตรวจสอบแบบขนาน สามารถทำได้เร็วกว่าแบบอนุกรม ทั้งนี้ก็สามารถอ่านข้อมูลเพียงไบต์เดียวก็สามารถรู้ได้ทันที ว่าอุปกรณ์ตัวใดเป็นผู้ขอบริการ

1.8 รูปแบบของข้อมูล

โดยทั่วไปข้อมูลจากอุปกรณ์ (device message) แบ่งได้ออกเป็น 3 ส่วนดังแสดงในรูปที่ 1.5 อันได้แก่ส่วนหัว (HR) ซึ่งจะอยู่ส่วนหน้าสุดเป็นตัวบอกชนิดข้อมูลส่วนประกอบแสดงในรูปที่ 1.6 จะเห็นว่าประกอบด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ ช่องว่าที่เป็นเว้นวรรค( ) ปกติจะมีอักษรประมาณ 1-3 ตัว

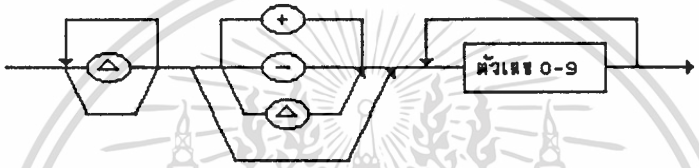


รูปที่ 1.5 แสดงส่วนต่างๆของข้อมูล

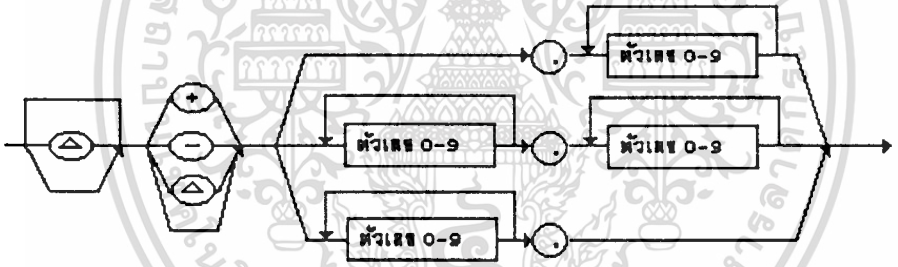


รูปที่ 1.6 แสดงส่วนประกอบของ HR

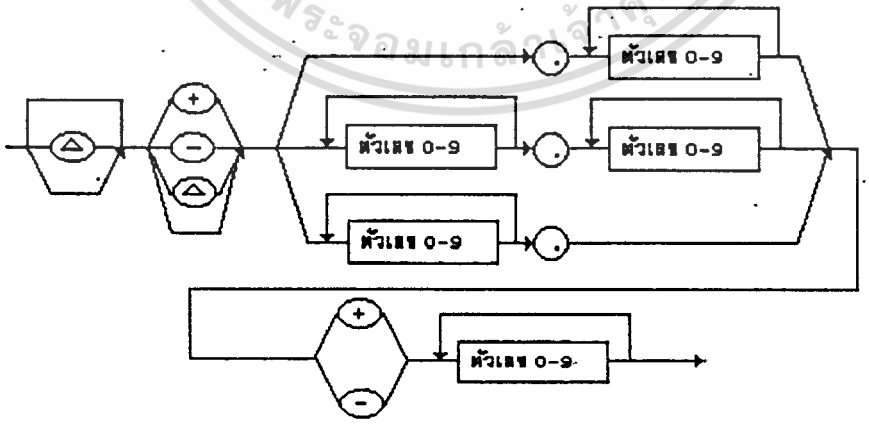
ส่วนที่ 2 คือเนื้อหาข้อมูล (NR) ซึ่งแสดงว่าค่าตัวเลขมีอยู่ 3 แบบคือ NR1, NR2, และ NR3 ดังแสดงในรูปที่ 1.7 ส่วนท้ายของ NR ยังอาจมีตัวอักษรแสดงหน่วยตามมา



แบบ NR1 ตัวเลขจำนวนเต็ม



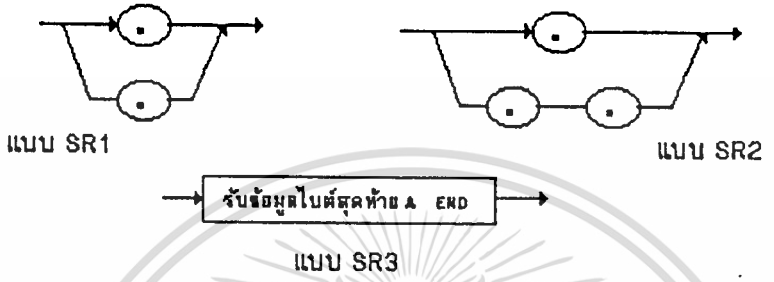
แบบ NR2 ตัวเลขจำนวนจริง



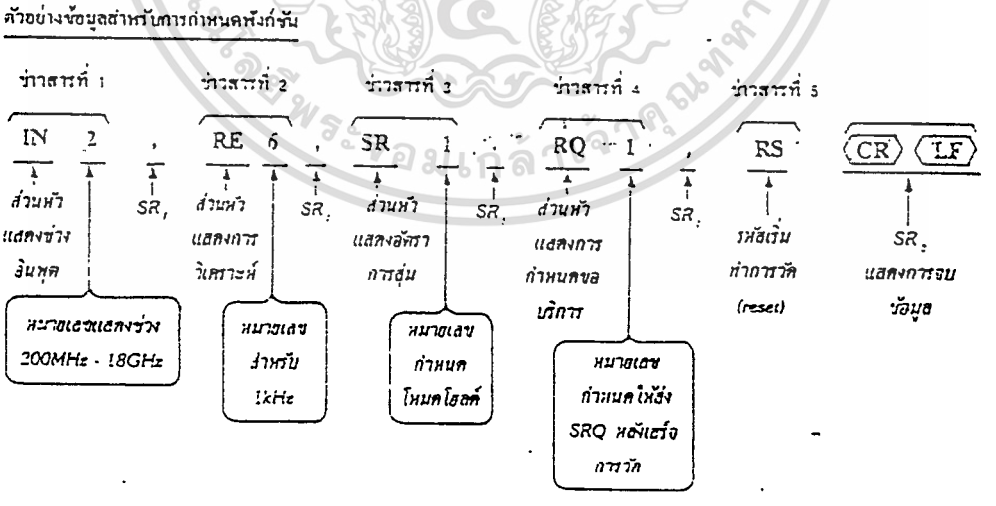
แบบ NR3 ตัวเลขยกกำลัง

รูปที่ 1.7 ส่วนเนื้อหาข้อมูล (NR)

ส่วนที่ 3 คือสัญญาณแบ่งข้อมูลแต่ละชุด (SR) ดังแสดงในรูปที่ 1.8 โดย SR1 ใช้แสดงการต่อเนื่องของข้อมูล (ข้อมูลยังมีต่อ) SR2 และ SR3 แสดงการสิ้นสุดของข้อมูล แต่ SR3 เป็นการบอกการเสร็จสิ้นข้อมูลทั้งหมดจากการวัดและรูปที่ 1.9 เป็นตัวอย่างข้อมูลสำหรับการกำหนดฟังก์ชั่นให้เครื่องวัดความถี่และข้อมูลความถี่ที่วัดได้



รูปที่ 1.8 แสดงส่วนของ (SR)



รูปที่ 1.9 แสดงตัวอย่างของข้อมูล

บทที่ 2

การจัดแอดเดรสสำหรับหน่วยความจำและ I/O

ในบทนี้จะกล่าวถึงการจัดแอดเดรส สำหรับหน่วยความจำและพอร์ท I/0 ต่างๆ ภายใน IBM/PCซึ่งจะแสดงถึงแอดเดรสต่างๆที่ถูกใช้งานโดยพอร์ท I/O(Input/OuptPort) และหน่วยความจำ นอกจากนี้จะได้กล่าวถึงเทคนิคการดีโค้ด (Decode)แอดเดรสในแบบต่างด้วย

2.1 การจัดแอดเดรสสำหรับพอร์ท I/Oใน IBM/PC

ในหัวข้อนี้จะกล่าวถึงวิธีการอ้างและใช้งานแอดเดรสต่างๆ ของพอร์ท I/O ที่ใช้งานอยู่ใน IBM/PC

2.2 การอ้างแอดเดรสของพอร์ท

ในการควบคุมและตรวจสอบสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิพซัพพอร์ทหรือการ์ดต่างๆที่ใช้ในระบบของ IBM/PC นั้น จะกระทำโดยผ่านทางพอร์ท ของระบบดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกรณ์เหล่านี้ จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ท I/O ต่างๆ ของระบบด้วย และเนื่องจากการควบคุมหรือติดต่อกับพอร์ทเหล่านี้ต้องกระทำโดยการอ้างถึงแอดเดรสของพอร์ท เหล่านั้นโดยตรงเราจึงจำเป็นต้องศึกษาถึงหลักการอ้างแอดเดรสของ 8088 ใน IBM/PC ด้วย

สำหรับแอดเดรสของพอร์ท I/O ต่างๆนั้น จะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 ซึ่งแอดเดรสเหล่านี้เป็นแอดเดรสที่จัดไว้สำหรับพอร์ท I/O โดยเฉพาะ คือแยกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ส่วนการส่งข้อมูลให้กับพอร์ทเหล่านี้จะทำได้โดย การใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลนั้นไปยังแอดเดรสของพอร์ทที่ต้องการ และสำหรับการตรวจสอบหรือการอ่านข้อมูลจากพอร์ทก็จะทำได้โดยการใช้คำสั่ง IN ของ8088 อ่านข้อมูลจากแอดเดรสของพอร์ทที่ต้องการเช่นกัน

ภายในไมโครโปรเซสเซอร์เบอร์ 8088 นี้จะมีแอดเดรสสำหรับใช้กับพอร์ท I/Oอยู่ทั้งสิ้น 65,536 หรือ 64K แอดเดรส (ในขณะที่มีแอดเดรสสำหรับหน่วยความจำอยู่ 1Mbyte) ซึ่งทำให้การอ้างแอดเดรสของพอร์ท I/O ที่ทำงานร่วมกับ 8088 นั้น ต้องใช้จำนวนเส้นแอดเดรสในบัสแอดเดรสในบัสแอดเดรสทั้งสิ้น 16 เส้น คือ A0-A15 แต่สำหรับใน IBM/PC นี้ถูกออกแบบมา

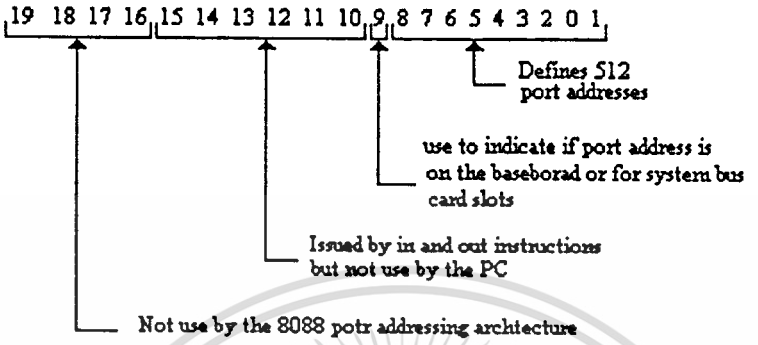
ให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่าง คือ A0-A9 เท่านั้นดังนั้นในการอ้างถึงแอดเดรสของพอร์ทของอุปกรณ์หรือชิพซัพพอร์ทใดๆ ที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้นแอดเดรสเพียง 10 เส้นด้วย โดยเส้นแอดเดรสที่เหลือคือ A10-A15 นั้นจะไม่ถูกนำไปใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้นแอดเดรส A10-A15 นี้จะไม่ถูกนำไปใช้งานแต่ค่าแอดเดรสบนเส้นแอดเดรสเหล่านี้ยังคงเปลี่ยนแปลงตามค่าแอดเดรสของพอร์ทที่กำหนดไว้ในคำสั่ง OUT หรือ IN อยู่ด้วย เพียงแต่ไม่ได้ถูกนำมาดีโค้ดร่วมกับแอดเดรส A0-A9 เท่านั้น ตัวอย่างเช่นในการใช้คำสั่ง OUT ส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0010H นั้นจะให้ผลเหมือนกับการส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0410 H, 0810H, 0C10H ทั้งนี้เนื่องจากแอดเดรส 6 บิทบนไม่ได้ถูกใช้งาน จึงทำให้การเปลี่ยนแปลงค่าแอดเดรสบนเส้นแอดเดรส A10-A15 นั้นไม่ทำให้เกิดความแตกต่างใดๆ ขึ้น

เนื่องจากใน IBM/PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้น(คือ A0-A9) ดังนั้นจึงสามารถที่จะอ้างแอดเดรสของพอร์ทได้สูงสุดเพียง 1024 พอร์ท (จากจำนวน 64K พอร์ท) เท่านั้นนอกจากนี้ในกรณีที่เป็นการอ่านข้อมูลจากพอร์ทของ IBM/PCข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ททั้ง 1024 พอร์ทออกเป็น 2 ส่วน(ส่วนละ 512 พอร์ท)อีกด้วย กล่าวคือถ้าข้อมูลในบิท A9เป็น “0” แล้วเราจะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทของอุปกรณ์หรือชิพซัพพอร์ทต่างๆ ที่อยู่บนเมนบอร์ด (Main Borad) ของ IBM/PC เช่น 8253-5 8237-5 หรือ 8259A เท่านั้น แต่ถ้าข้อมูลในบิท A9 นี้เป็น “1” ก็จะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทที่อยู่บนการ์ดต่างๆเท่านั้น

จากที่ได้กล่าวมานั้นจะสรุปได้ว่าพอร์ทบน IBM/PC ทั้ง 1024 พอร์ทถูกแบ่งออกเป็น 2 กลุ่มโดยที่กลุ่มแรกเป็นกลุ่มของพอร์ทที่อยู่บนเมนบอร์ด และกลุ่มที่สองเป็นกลุ่มที่จัดเตรียมไว้สำหรับพอร์ทที่อยู่บนการ์ดต่างๆ

สำหรับในกรณีของการส่งข้อมูลให้กับพอร์ททั้ง 1024 พอร์ท เราสามารถที่จะเลือกส่งไปยังพอร์ทใดๆ ใน IBM/PC ได้ ดังนั้นการเลือกแอดเดรสสำหรับพอร์ทที่อยู่บนการ์ดจึงสามารถทำได้โดยสะดวกแต่อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงก็คือถ้าแอดเดรสที่เราเลือกให้กับพอร์ทนี้ตรงกับค่าแอดเดรสเดิมที่มีอยู่บนเมนบอร์ดแล้ว เมื่อเราทำการส่งข้อมูลให้กับพอร์ทที่อยู่ในตำแหน่งแอดเดรสนี้ก็จะเท่ากับเป็นการส่งข้อมูลให้กับทั้งพอร์ทที่อยู่บนเมนบอร์ดและพอร์ทที่อยู่บนการ์ดด้วย ซึ่งในกรณีเช่นนี้อาจจะก่อให้เกิดความผิดพลาดขึ้นได้เช่นกัน ดังนั้นในการกำหนดค่าแอดเดรสให้กับพอร์ทที่ถูกสร้างขึ้นบนการ์ดต่างๆ จึงควรจะใช้ค่าแอดเดรสที่แอดเดรสบิท A9 มีค่าเป็น “1” คือ แอดเดรส 0EF00H จนถึง 0FFFFH เท่านั้น(แอดเดรสบิท A0-A15 ไม่ถูกใช้ในการดีโค้ด แต่เพื่อความสะดวกจึงกำหนดให้มีค่าเป็น “1” ในฐานสองทั้งหมด แต่ในการใช้งานจริงอาจเปลี่ยนให้แอดเดรส A0-A15 แต่ละบิทมีค่าเป็น “1” หรือ “0”ก็ได้)

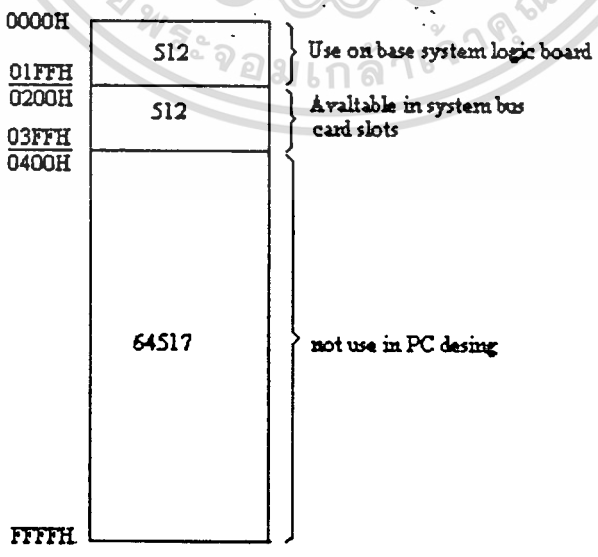
สำหรับรูปที่ 2.1 นี้ จะแสดงถึงการใช้งานแอดเดรสบิทต่างๆ ในการอ้างแอดเดรสของพอร์ทใน IBM/PC



รูป 2.1 การใช้แอดเดรสบิทต่างๆในการอ้างแอดเดรสของพอร์ทใน

2.3 การใช้งานแอดเดรสสำหรับพอร์ท I/O ใน IBM/PC

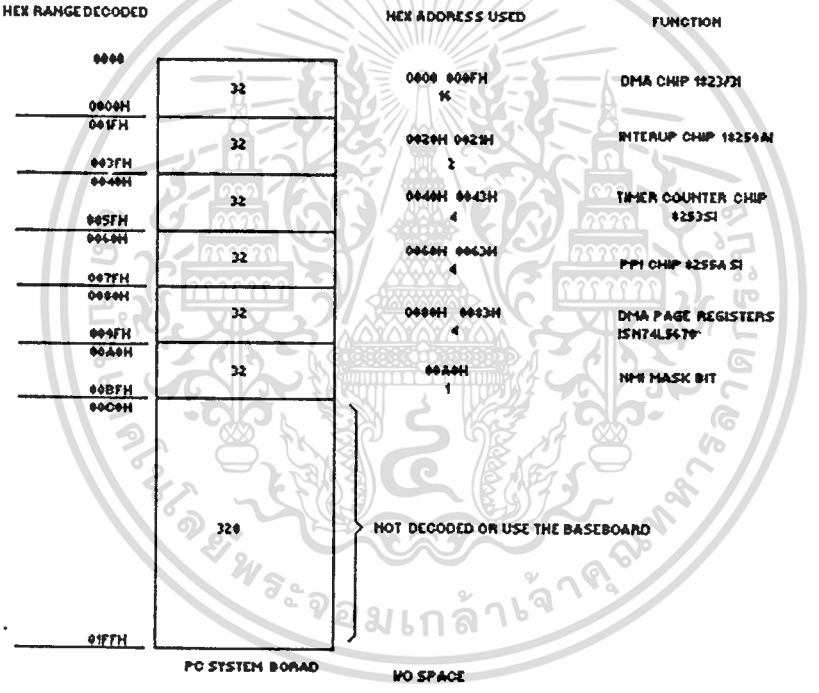
จากที่ได้กล่าวไว้ในหัวข้อที่ผ่านมานั้นพอร์ท I/O ทั้ง 1024 พอร์ทใน IBM/PC จะถูกแบ่งออกเป็น 2 กลุ่มๆละ 512 พอร์ท สำหรับในหัวข้อนี้จะกล่าวถึงการใช้งานพอร์ทต่างๆ เหล่านี้โดยจะแบ่งออกเป็น 2 กลุ่มตามที่ได้อธิบายไว้ในหัวข้อที่ผ่านมาดังนี้



รูปที่ 2.2 การใช้งานแอดเดรสของพอร์ทบน IBM/PC

1. ในกลุ่มแรกนี้เป็นกลุ่มของพอร์ท I/O ที่อยู่เมนบอร์ดของ IBM/PC ซึ่งจะมีแอดเดรสอยู่ในตำแหน่ง 0000H จนถึง 01FFH (ขอให้ระลึกอยู่เสมอว่า A10- A15 นั้นไม่ถูกใช้งาน)หรือแอดเดรสที่มีบิท 9 เป็น "0"นั่นเอง

สำหรับแอดเดรสของพอร์ท I/O ในกลุ่มนี้จะถูกใช้ ในการอ้างแอดเดรสของชิพซัพพอร์ทและอุปกรณ์ที่เป็น I/O ต่างๆบนเมนบอร์ดของ IBM/PC เช่น แอดเดรส 000H จนถึง 000FH จะถูกใช้เป็นแอดเดรสสำหรับ 8237-5 DMA Controller เป็นต้น ในรูปที่ 2-3 จะแสดงถึงการใช้งานแอดเดรสต่างๆ ตั้งแต่ 000FH จนถึง 01FFH ในการอ้างแอดเดรสของชิพซัพพอร์ทและอุปกรณ์ต่างๆที่ทำหน้าที่เป็น I/O บนเมนบอร์ดของ IBM/PC

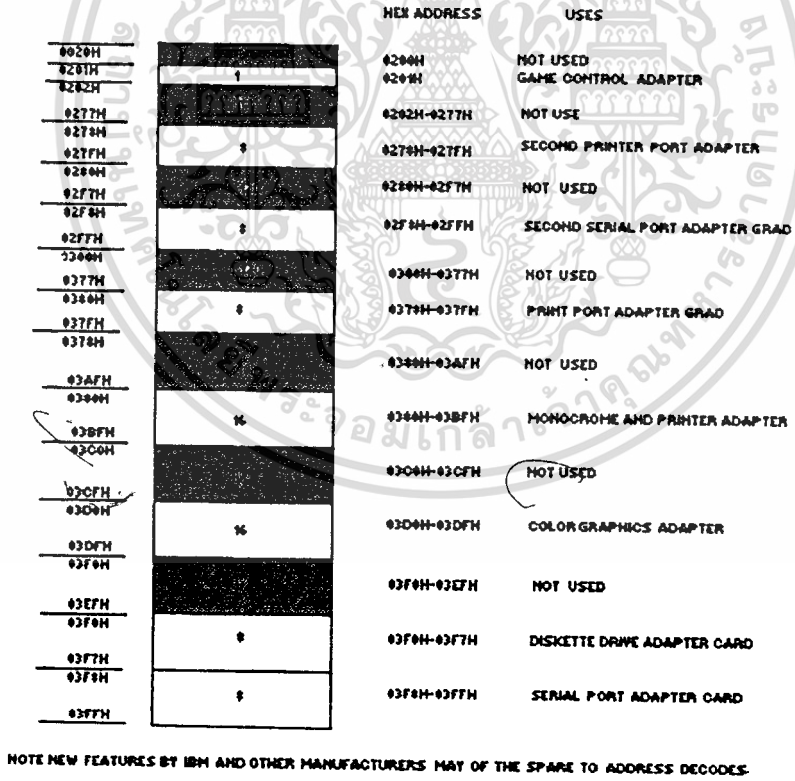


รูปที่ 2.3 การใช้งานแอดเดรสต่างๆสำหรับพอร์ท I/O ของ IBM/PC

จากรูปข้างต้นจะเห็นว่าแอดเดรส 00C0H จนถึงแอดเดรส 01FFH นั้นไม่ได้ถูกใช้งานบนเมนบอร์ดของ IBM/PC ดังนั้นในกรณีนี้เราก็สามารถที่จะใช้งานแอดเดรสต่างๆ เหล่านี้ได้ แต่อย่างไรก็ตามแอดเดรสเหล่านี้ยังคงถูกดีโค้ดให้เป็นแอดเดรสที่ใช้ในการอ่านข้อมูลจากพอร์ท I/O บนเมนบอร์ดเท่านั้น ดังนั้นการใช้ค่าแอดเดรส 00C0H-01FFH กับพอร์ทบนการ์ดหรือวงจรอินเทอร์เฟสที่เราสร้างขึ้น ต้องเป็นพอร์ทเอาท์พุทเพียงชนิดเดียวเท่านั้นกล่าวคือจะทำการอ่านข้อมูลจากพอร์ท I/O (ที่ไม่ได้อยู่บนเมนบอร์ด)ที่มีค่าแอดเดรสอยู่ในช่วง 00C0H-01FFHไม่ได้

2. ในกลุ่มที่ 2 นี้จะเป็นกลุ่มของพอร์ท I/O ที่ถูกใช้งานอยู่บนการ์ดที่ใช้เสียบบนสล็อตต่างๆของ IBM/PC สำหรับแอดเดรสของพอร์ท เหล่านี้จะเริ่มต้นจากแอดเดรส 0200H จนถึง 03FFH ซึ่งก็คือแอดเดรสที่มีบิท A9 เป็น "1" นั่นเองสำหรับการใช้งานแอดเดรสของพอร์ท I/O ในกลุ่มนี้จะแสดงได้ดังรูป 2-4

อย่างไรก็ตาม การใช้งานแอดเดรสในกลุ่มนี้อาจจะเปลี่ยนแปลงไปได้ ทั้งนี้ขึ้นอยู่กับการใช้งานการ์ดต่างๆร่วมกับ IBM/PC โดยการ์ดที่ถูกออกแบบผลิตขึ้นใหม่นั้นอาจจะใช้ค่าแอดเดรสต่างๆที่เหลืออยู่นี้ได้ดังนั้นก่อนที่จะทำการออกแบบวงจรอินเตอร์เฟสที่จำเป็นต้องใช้ค่าแอดเดรสสำหรับพอร์ท I/O จึงควรจะตรวจสอบดูก่อนว่าการ์ดต่างๆ ที่ใช้อยู่ในระบบของ IBM/PCที่เราใช้งานอยู่นั้นมีการ์ดใดบ้าง และการ์ดเหล่านั้นใช้งานแอดเดรสใดบ้าง จากนั้นจึงทำการออกแบบวงจรอินเตอร์เฟสโดยเลือกใช้เฉพาะแอดเดรสที่ยังไม่ถูกใช้งาน



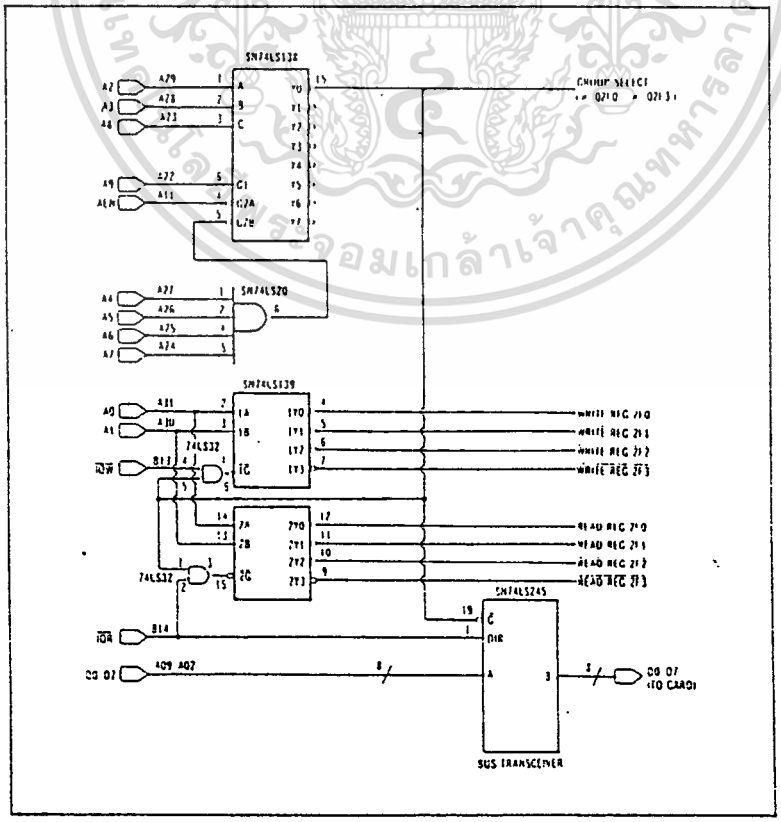
รูปที่ 2.4 การใช้งานแอดเดรสสำหรับพอร์ท I/O บนการ์ดต่างๆ

2.4 เทคนิคในการดีโค้ดแอดเดรสสำหรับพอร์ท I/O

ในหัวข้อต่างๆ ที่ผ่านมาข้างต้นนั้นได้กล่าวถึงการอ้างแอดเดรสและการใช้งานแอดเดรสต่างๆของพอร์ท I/O ใน IBM/PC สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการต่างๆที่ใช้ในการดีโค้ดให้เป็นไปตามที่เราต้องการ

2.5 การดีโค้ดแบบ Fixed

วิธีการดีโค้ดแบบนี้เป็นวิธีง่ายๆและสะดวกในการดีโค้ดแอดเดรสหรือกลุ่มแอดเดรสของพอร์ท I/O ซึ่งวิธีนี้เป็นการกำหนดจำนวนของแอดเดรสที่เราต้องการใช้ จากนั้นจึงทำการเลือกบล็อกของแอดเดรสที่ยังไม่ถูกใช้งานโดยการ์ดหรือวงจรอินเตอร์เฟสอื่นๆ (บล็อกของแอดเดรสที่เลือกต้องมีจำนวนแอดเดรสเพียงพอกับจำนวนที่เราต้องการใช้งาน) แล้วจึงออกแบบวงจรที่ทำการดีโค้ดแอดเดรสที่เราต้อง การสำหรับตัวอย่างวงจรที่ใช้ในการดีโค้ดแอดเดรสในแบบนี้จะแสดงได้ดังรูป 2.5



รูปที่ 2.5 ตัวอย่างวงจรดีโค้ดแอดเดรสแบบ Fixed

จากรูปจะเห็นได้ว่าวงจรที่ใช้นี้เป็นวงจรที่สามารถทำการดีโค้ดแอดเดรสได้ 8 กลุ่มโดยแต่ละกลุ่มจะมีจำนวนแอดเดรส 4 แอดเดรสซึ่งแอดเดรสทั้ง 8 กลุ่มจะแสดงได้ตารางดังกล่าว

กลุ่ม	แอดเดรส
0 (Y0)	02F0H-02F3H
1 (Y1)	02F4H-02F7H
2 (Y2)	02F8H-02FBH
3 (Y3)	02FCH-02FFH
4 (Y4)	03F0H-03F3H
5 (Y5)	03F4H-03F7H
6 (Y6)	03F8H-03FBH
7 (Y7)	03FCH-03FFH

สำหรับในตัวอย่างนี้จะเลือกใช้การดีโค้ดแอดเดรสในกลุ่ม 0 (เริ่มจากแอดเดรส 02F0H จนถึง 02 F3H) คือใช้สัญญาณเอาท์พุท(สัญญาณ GROUPSELECT) จากขา Y0 (ขา 15) ของ74LS138 ไปทำการ OR กับสัญญาณ IOR และ IOW เพื่อสร้างเป็นสัญญาณอีนาเบิลวงจรดีโค้ด(74LS139) แอดเดรสอีก 4 แอดเดรสซึ่งแบ่งเป็น 2 ชุดคือชุดที่เป็น WRITE REG ซึ่งจะแอคทีฟ (ลอจิก "0") เมื่อ CPU ต้องการจะส่งข้อมูลให้กับวงจรภายนอก(สัญญาณ IOR แอคทีฟ) และชุดที่เป็น READ REG ซึ่งจะแอคทีฟเมื่อ CPU ต้องการจะอ่านข้อมูลจากวงจรภายนอก (สัญญาณ IOR แอคทีฟ) สัญญาณ WRITE REG และ READ REG นี้โดยทั่วไปจะนำไปเป็นสัญญาณสโตรบ (Strobe)ให้กับวงจรภายนอกที่เกี่ยวข้อง เพื่อให้สามารถส่งหรือรับข้อมูลจาก CPU ได้ในช่วงเวลาที่เหมาะสม นอกจากนี้สัญญาณ GROUPSELECT ยังถูกนำไปใช้ในการอีนาเบิลบัฟเฟอร์ 74LS245 ด้วยเพื่อให้ CPU สามารถส่งหรือรับข้อมูลจากอุปกรณ์ภายนอกได้ เมื่อแอดเดรสในกลุ่มนี้ถูกเลือกสำหรับทิศทางของข้อมูลจะถูกควบคุมโดยสัญญาณ IOR ส่วนสัญญาณ AEN จะถูกนำมาใช้ในการดิสเอเบิลวงจรดีโค้ดโดยถ้าสัญญาณ AEN เป็น "1"ซึ่งเป็นช่วงเวลาของขบวนการ DMA นั้น 74LS318 จะถูกดิสเอเบิลทันที ทั้งนี้ก็เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นเนื่องจากการดีโค้ดแอดเดรสของพอร์ทในระหว่างขบวนการ DMA นั่นเอง (ในระหว่างนี้แอดเดรสบนบัสแอดเดรสจะเป็นแอดเดรสของหน่วยความจำ คือสัญญาณ MEMW หรือ MEMR จะแอคทีฟ แต่ในขณะเดียวกันสัญญาณ

IOR หรือ IOW ก็จะแอคทีฟด้วยดังนั้นถ้าไม่ดิสเอเบิลวงจรดีโค้ดไว้แล้ว อาจจะทำให้วงจรดีโค้ดคิดว่าแอดเดรสบนบัสแอดเดรสเป็นแอดเดรสของพอร์ท I/O ก็ได้)

ในการดีโค้ดแอดเดรสของพอร์ท I/O เราจะต้องคำนึงถึงช่วงเวลาของสัญญาณที่เกิดขึ้นในขบวนการอ่านหรือเขียนข้อมูลลงบนพอร์ท I/O ดังนี้

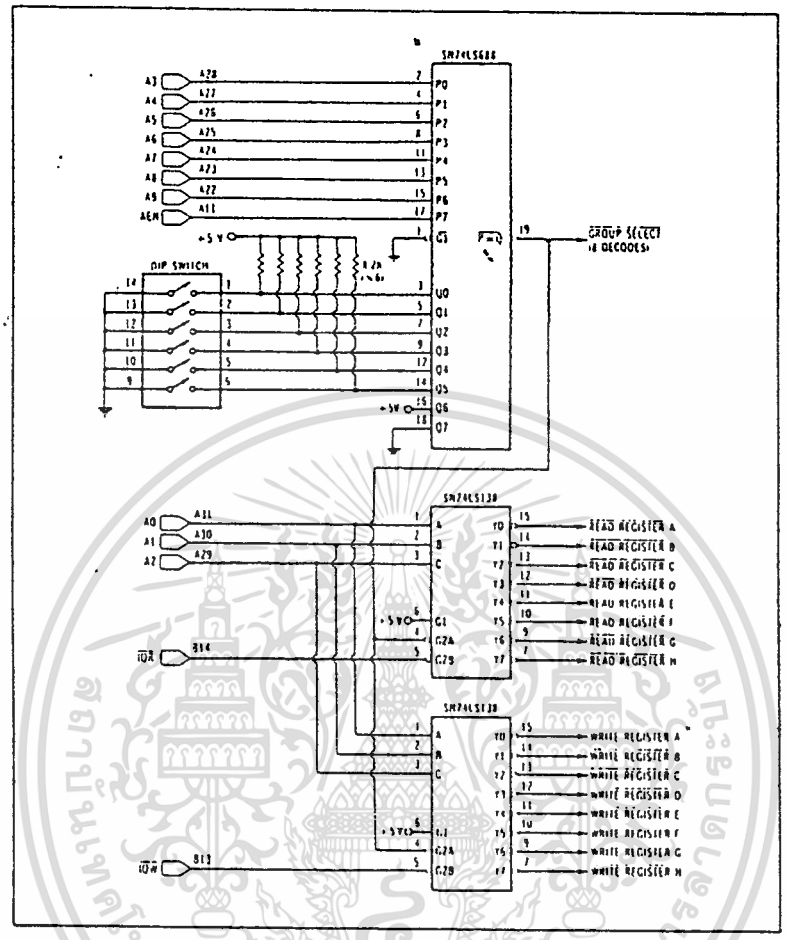
1. ในช่วงเริ่มต้นของบัสไซเคิลที่เกี่ยวกับพอร์ท I/O ดังนั้นถ้าสัญญาณจากวงจรดีโค้ดมีการหน่วงเวลา (Delay) มากเกินไปอาจจะทำให้สัญญาณ ดีโค้ดนี้เกิดขึ้นหลังจากที่สัญญาณIOR หรือ IOW แอคทีฟ และเนื่องจากค่าแอดเดรสบนบัสแอดเดรสนั้นเปลี่ยนแปลงได้ตลอดเวลา ดังนั้นก่อนที่ค่าแอดเดรสที่ถูกต้องจะถูกส่งออกมาบนบัสแอดเดรสนั้นวงจรดีโค้ดจะได้รับค่าแอดเดรสอื่นๆอยู่ ซึ่งถ้าหากวงจรดีโค้ดมีการหน่วงเวลามากไปแล้ว สัญญาณดีโค้ดแอดเดรสที่ไม่ถูกต้องนี้อาจจะถูกหน่วงเวลาจนเกิดในช่วงเวลาที่สัญญาณ IOR หรือ IOW เกิดขึ้นแล้วก็ได้ทำให้ข้อมูลบนบัสข้อมูลนั้นถูกส่งไปยังพอร์ทที่ไม่ถูกต้องสำหรับใน IBM/PC จะถูกออกแบบให้การหน่วงเวลาในวงจรดีโค้ดนั้นมีค่าไม่เกิด 92 nanosec.

2. ในช่วงท้ายของบัสไซเคิลในการเขียนข้อมูลลงบนพอร์ท I/O นั้นถ้าสัญญาณ IOW มีการหน่วง เวลาออกไปและวงจรดีโค้ดมีความเร็วในการทำงานสูงแล้วอาจจะทำให้ข้อมูลในบัสไซเคิลนี้ถูกส่งให้กับพอร์ท I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสในบัสไซเคิลต่อไปก็ได้สำหรับใน IBM/PC สัญญาณ IOW จะมีการหน่วงเวลาไปไม่เกิน 200 nanosec

อย่างไรก็ตามช่วงเวลาที่ต้องสนใจมากอีกช่วงเวลาหนึ่งก็คือช่วงเวลาระหว่างขอบขาขึ้นของสัญญาณ LOW กับช่วงเวลาที่ข้อมูลที่ถูกต้องที่ถูกส่งออกมาบนบัสข้อมูลถ้าสัญญาถูกหน่วงเวลาไปเกิน 120 nanosec. แล้วอาจทำให้พอร์ท I/O ได้รับข้อมูลที่ไม่ถูกต้องก็ได้และสำหรับสัญญาณ IOR นั้นถ้ามีการหน่วงเวลาเกิดขึ้นแล้วก็จะให้ความเร็วในการอ่านข้อมูลถูกลดลง

2.6 การดีโค้ดโดยใช้สวิทช์เลือก

การดีโค้ดในแบบ Fixed ที่ได้กล่าวไว้ในหัวข้อที่ผ่านมานั้น มีข้อเสียอยู่บางประการคือแอดเดรสที่เราเลือกใช้งานไว้นั้น อาจจะซ้ำกับแอดเดรสของการ์ดอื่น ที่เรานำมาเพิ่มเข้าไปในระบบในภายหลังก็ได้ซึ่งกรณีเช่นนี้เราต้องแก้ไขวงจรเพื่อหลีกเลี่ยงแอดเดรสอื่นที่ยังว่างอยู่และ ไม่ถูกใช้งานโดยการ์ดที่เพิ่มเข้าไปใหม่ซึ่งยุ่งยากและต้องเสียเวลามากขึ้น ปัญหาเช่นนี้เราสามารถแก้ไขได้โดยใช้วงจรดีโค้ดที่เราสามารถเปลี่ยนแปลงค่าแอดเดรสได้โดยเพียงแต่เปลี่ยนตำแหน่งสวิทช์ (ในที่นี้คือ DIP Switch) ที่เซ็ทไว้ในวงจรเท่านั้นดังรูปที่ 2.6



รูปที่ 2.6 ตัวอย่างวงจรดีโค้ดโดยใช้สวิทช์เลือก

จากรูป 2.6 เป็นวงจรที่ทำการดีโค้ดกลุ่มแอดเดรสขนาด 8 แอดเดรสซึ่งการเลือกกลุ่มแอดเดรสที่จะทำการดีโค้ดนี้จะทำได้โดยการเซ็ท DIP Switch ที่ขา Q0-Q5 ของ 74LS688

สำหรับหน้าที่ของ 74LS688 นี้จะทำการเปรียบเทียบค่าของอินพุท 2 ชุด ที่ถูกส่งเข้ามาทางขา P0-P7 และขา Q0-Q7 ถ้าอินพุททั้ง 2 ชุดนี้เท่ากันแล้วเอาท์พุทที่ขา P=Q จะให้เอาท์พุทเป็นลอจิก "0" จากในวงจรขา P0-P6 ของ 74LS688 ต่อกับแอดเดรสบิท A3-A9 ในขณะที่ขา Q0-Q5 ต่อกับความต้านทานที่ทำหน้าที่เป็น Pull Up (รักษาระดับแรงดันให้เป็น ลอจิก "1" ไว้ในกรณีที่ไม่มีอินพุทใดๆเข้ามา) และขา QO-Q5 นี้จะต่อกับปลายข้างหนึ่งของ DIP Switch ด้วยส่วนปลายอีกข้างหนึ่งของ DIP Switch นั้นจะต่อลง Ground (ลอจิก "0") ไว้ดังนั้นถ้าเราทำการ "ON" DIP Switch ที่ต่อกับขาใดขานั้นก็จะได้รับลอจิก "0" ในขณะที่ถ้า DIP Switch ที่ต่อกับขาใดถูก "OFF" ขานั้นก็จะได้รับลอจิก "1"และเนื่องจากอินพุทที่ขา (แอดเดรส A0-A9) ต้องเท่ากับอินพุทที่ขาQ0-Q5 ดังนั้นถ้าเราเปลี่ยนแปลงการเซ็ต DIP Switch เหล่านี้ก็จะทำให้แอดเดรสบิท

A3-A5 ซึ่งต่อกับขา P0-P5 นั้นต้องเปลี่ยนแปลงตามไปด้วยจึงจะทำให้เอาท์พุทของ 74LS688 แอคทีฟ ได้ทำให้เราสามารถเปลี่ยนค่าแอดเดรสที่ต้องการจะดีโค้ดได้ง่ายกว่าวิธีการดีโค้ดแบบ Fixed สำหรับขา Q6 นั้นจะต่อกับลอจิก "1" (+5 V)และขา P6 ต่อกับแอดเดรสบิท A9 ในกรณีเช่นนี้จึงเท่ากับเป็นการบังคับให้แอดเดรสที่ทำการดีโค้ดได้นั้นจะต้องมีแอดเดรสบิท A9 เป็น "1" เท่านั้นส่วนขา P7 จะต่อกับสัญญาณโดยมีขา Q7 ต่อกับลอจิก "0" การต่อในลักษณะนี้เพื่อป้องกันไม่ให้ 74LS688 ทำการดีโค้ดในระหว่างขบวนการ DMA นั้นเอง เอาท์พุทจากขา P=Q ของ 74LS688 นี้จะถูกนำไปใช้ในการอีนาเบิล ซึ่งทำหน้าที่ในการดีโค้ดแอดเดรส 8 แอดเดรสของกลุ่มแอดเดรสที่เราเลือก (โดยใช้ DIP Switch ดังที่ได้กล่าวในตอนต้น)

วงจรในลักษณะนี้เราสามารถจะนำไปใช้ในการดีโค้ดในแบบ Fixed ได้โดยการนำเอา DIP Switch ออก จากนั้นถ้าอินพุทใดต้องการลอจิก "0" จึงจะใช้ตัวนำเชื่อมต่อระหว่างขั้วทั้งสองแทนการเซ็ท DIP Switch ให้ แต่ถ้าอินพุทใดต้องการลอจิก "1" ก็ปล่อยขั้วทั้งสองนั้นไว้

2.7 การใช้งานหน่วยความจำใน IBM/PC

สำหรับไมโครโปรเซสเซอร์เบอร์ 8088 ซึ่งใช้เป็น CPUใน IBM/PC นี้จะมีความสามารถในการอ้างแอดเดรสของหน่วยความจำได้ถึง 1Mbyte และใน IBM/PC ก็ถูกออกแบบให้ใช้เนื้อที่ในหน่วยความจำเหล่านี้ เพื่อการทำงานในส่วนต่างๆของระบบด้วย ดังนั้นในการที่จะศึกษาถึงระบบของ IBM/PC จึงจำเป็นต้องศึกษาถึงการใช้เนื้อที่ในหน่วยความจำของ IBM/PC ด้วย

หน่วยความจำภายใน IBM/PC นั้นจะแบ่งออกเป็น 2 ส่วนที่เป็น ROM ซึ่งอยู่ตำแหน่งแอดเดรส 64K ไบท์บน (0F0000H-0FFFFFH) และส่วนที่เป็น RAM ซึ่งเริ่มต้นจากตำแหน่งแอดเดรสแรกคือ 00000H ขึ้นมา

สำหรับหน่วยความจำส่วนที่เป็น IBM/PCทั้ง 64Kbyte จะถูกใช้งานอยู่เพียง 40 Kbyte เท่านั้น โดยภายใน ROM ทั้ง 40 Kbyte นี้ จะประกอบด้วยโปรแกรมในส่วนที่เป็น BIOS, Diagnostics, Cassette Operting System (โปรแกรมส่วนนี้ใน IBM PC/XT จะไม่มี) ,Bootstrap Loader และ Interpreter ของภาษา BASIC

นอกจาก ROM 40 Kbyteที่กล่าวถึงนี้แล้วเรายังสามารถจะเพิ่ม ROM ให้กับระบบได้อีก 8 Kbyte โดยเสียบเพิ่มเข้าในซอกเก็ต (Socket) ที่เตรียมไว้บนเมนบอร์ดสำหรับแอดเดรสของหน่วยความจำในส่วนที่เหลืออยู่อีก 16Kbyte (64K-40K-8K=16K) นั้นเป็นส่วนที่ถูกดีโค้ดไว้แล้ว แต่ไม่ถูกใช้งานใดๆทั้งสิ้น (คือเป็นแอดเดรสส่วนที่สูญเปล่า) ในส่วนของแอดเดรสที่เป็น

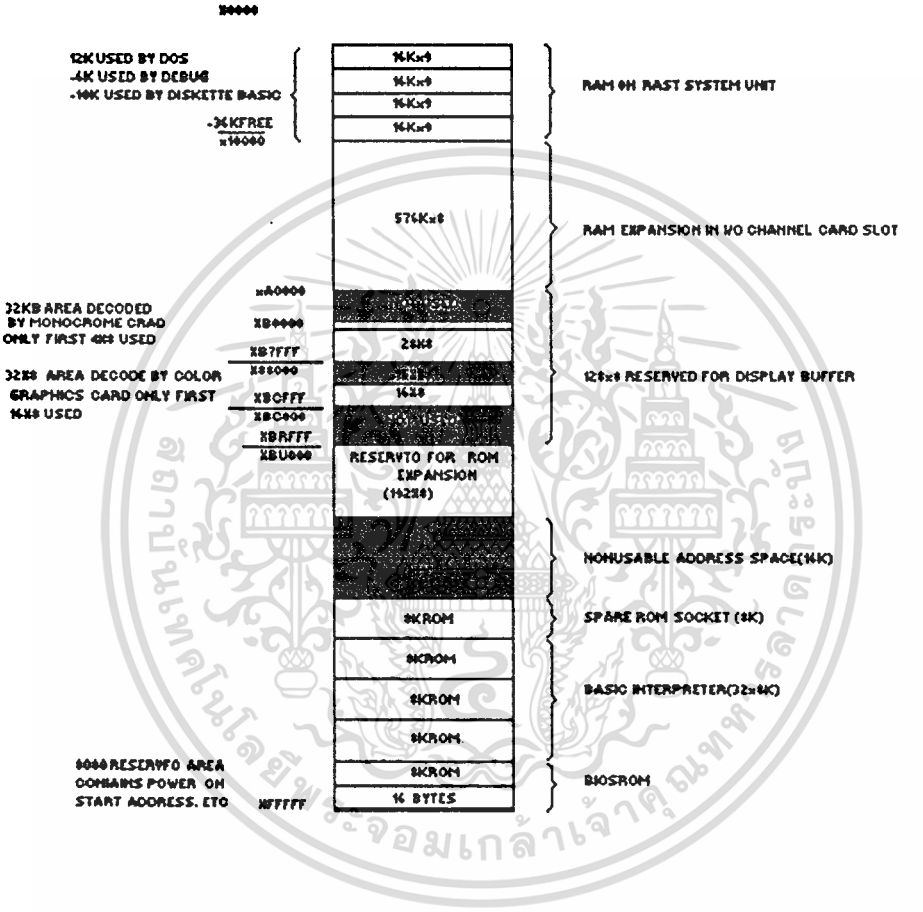
หน่วยความจำ ROM ทั้ง 64Kbyte นี้ เป็นส่วนที่ถูกดีโค้ดไว้เพื่อใช้งานบนเมนบอร์ดเท่านั้น ไม่สามารถนำไปใช้งานบนสล็อตของ IBM/PC ได้

หน่วยความจำในอีกส่วนหนึ่งซึ่งเป็นส่วนที่เป็น RAM นั้น จะเริ่มต้นจากแอดเดรส 00000H โดยแอดเดรสในส่วนของ 64Kbyte แรก (00000H-0FFFFH : ในกรณีของ IBM/PC/XT จะเป็น 256Kbyte) จะถูกดีโค้ดบนเมนบอร์ด นอกจากนั้นจะเป็นส่วนของ RAM ที่ถูกเพิ่มเติมขึ้นโดยการ์ดต่างๆ บนสล็อตของ IBM/PC

สำหรับการใช้งานหน่วยความจำ 64Kbyte แรกนี้จะแบ่งออกเป็นส่วนๆ เช่นเดียวกับในหน่วยความจำส่วนที่เป็น ROM โดยหน่วยความจำ 12Kbyte แรกจะใช้สำหรับเก็บโปรแกรมส่วนที่เป็น DOS เมื่อมีการบูท DOS เข้ามาในระบบ และเมื่อมีการโหลดโปรแกรม DEBUG เข้ามาในระบบ IBM/PC ก็จะใช้เนื้อที่ในอีก 6Kbyte ต่อมาในการเก็บโปรแกรม DEBUG นี้ และในกรณีที่มีการโหลดโปรแกรมที่เป็น Interpreter ของAdvanceBASI เข้ามาในระบบด้วย ก็จะใช้เนื้อที่ในหน่วยความจำอีก 10Kbyte ในการเก็บส่วนของโปรแกรมนี้ ดังนั้นในส่วนของหน่วยความจำ 64 Kbyte แรกนี้จะเหลือเนื้อที่สำหรับใช้งานอยู่อีก 36Kbyte อย่างไรก็ตามเนื้อที่ว่างที่เหลืออยู่นี้จะเปลี่ยนแปลงไปตามเวอร์ชัน (Version) ของซอฟท์แวร์ต่างๆที่ใช้ในระบบ

นอกจากหน่อยความจำทั้ง 2 ส่วนที่กล่าวถึงแล้ว ยังมีหน่วยความจำในส่วนอื่นๆอีกโดยเฉพาะในส่วนที่เกี่ยวกับการแสดงผลออกทางจอภาพ หน่วยความจำในส่วนนี้เป็นส่วนที่เก็บข้อมูลซึ่งใช้ในการแสดงผลออกทางจอภาพโดยทั่วไปจะเรียกหน่วยความจำนี้เป็นDisplay Buffer หน่วยความจำในส่วนนี้จะเป็นหน่วยความจำ RAM ที่ถูกดีโค้ด และใช้งานบนการ์ดที่ทำหน้าที่ในการแสดงผลออกบนจอภาพ (Display Card) ซึ่งมีอยู่ 2 ลักษณะคือการ์ดที่แสดงผลบนจอภาพสี(Color Monitor) หรือที่เรียกโดยทั่วไปว่า Color Graphics Card ซึ่งจะใช้หน่วยความจำขนาด 16Kbyte เป็น Display Buffer (จากที่เตรียมไว้ 32Kbyteสำหรับ 16Kbyte ที่เหลือจะไม่ถูกใช้งาน)ส่วนการ์ดอีกลักษณะหนึ่งจะเป็นการ์ดที่แสดงผลออกบนจอ Monochrome หรือที่เรียกโดยทั่วไปว่า Monochrome Display Cardซึ่งจะใช้เนื้อที่ในหน่วยความจำเพื่อใช้เป็น เพียง4Kbyteเท่านั้น (จากที่เตรียมไว้ถึง 32Kbyteโดยแอดเดรสของหน่วยความจำที่เหลืออีก 28Kbyte นั้นจะไม่ถูกใช้งานใด) อย่างไรก็ตามเนื่องจากการ์ดที่ใช้ในการแสดงผลนี้ โดยทั่วไปจะเลือกใช้เพียงชนิดเดียวเท่านั้น ดังนั้นเมื่อเราเลือกใช้การ์ดแสดงผลแบบใดแบบหนึ่งในระบบ เราก็จะสามารถใช้แอดเดรสของหน่วยความจำที่เป็น Display Buffer ของอีกการ์ดหนึ่งเพื่อนำไปใช้ในงานอื่นๆได้ นอกจากแอดเดรสของหน่วยความจำที่จัดเตรียมไว้สำหรับใช้เป็น Display Buffer แล้วในIBM/PC ยังกัน(Reserve) เนื้อที่อีกส่วนหนึ่งไว้ด้วย เพื่อเตรียมไว้ให้กับอุปกรณ์หรือการ์ดอื่นๆ ที่อาจจะถูกสร้างขึ้นในอนาคต ดังนั้นจึงควรที่จะหลีกเลี่ยงการใช้หน่วยความจำในส่วนนี้ไว้ด้วย ทั้งนี้ก็เพื่อให้ระบบของ

เรานั้นสามารถใช้งานอุปกรณ์หรือการ์ดที่ถูกสร้างขึ้นใหม่เหล่านี้ได้ สำหรับรูปที่ 2-7 นี้จะแสดงถึงแผนผังการใช้งานหน่วยความจำต่างๆ ภายในระบบของ IBM/PC



รูปที่ 2.7 แผงผังการใช้งานหน่วยความจำ IBM/PC

2.8 การใช้งานหน่วยความจำใน IBM PC/XT

ภายใน IBM PC/PX จะมีการใช้งานหน่วยความจำในลักษณะที่คล้ายกับใน IBM/PC เพียงแต่ขนาดของหน่วยความจำที่จัดไว้นั้นจะแตกต่างกันกล่าวคือหน่วยความจำส่วนที่เป็นROM นั้นจะมีขนาด 64 Kbyte (จาก 0F0000H-0FFFFFH) ซึ่งใช้สำหรับเก็บ BIOS, Interpreteของภาษา

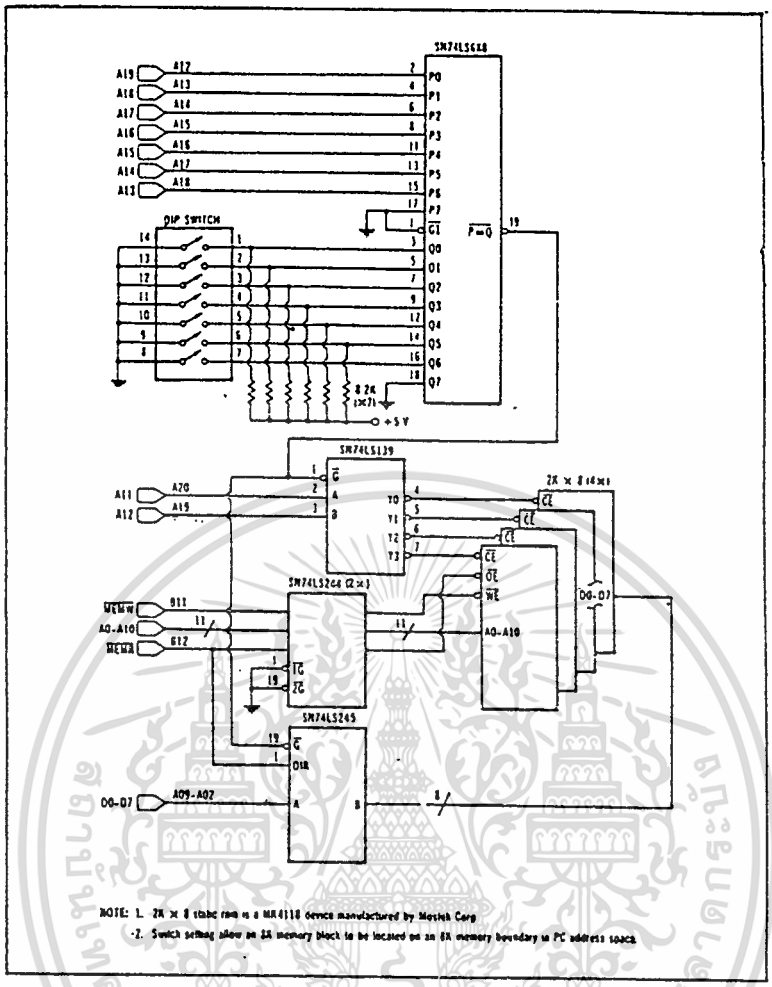
BASIC ฯลฯ เช่นเดียวกับใน IBM/PCนอกจากนี้เตรียมเนื้อที่ในหน่วยความจำอีก 192Kbyte (จาก 0C0000H-0EFFFFH) ไว้สำหรับหน่วยความจำ ROM ที่เพิ่มเติมขึ้นในภายหลังอีกด้วย

สำหรับหน่วยความจำ RAM บนเมนบอร์ดจะถูกเพิ่มขึ้นให้มีขนาดตั้งแต่ 128Kx9 จนถึง 256x9 (บิทที่ 9 ซึ่งเกินมานั้นจะถูกใช้เป็นบิท Parity ของข้อมูลในแต่ละไบท์) โดยแอดเดรสของหน่วยความจำ RAM ในส่วนนี้จะอยู่ในช่วง 00000H-3FFFFH สำหรับในกรณีที่มีความจำเป็นต้องใช้หน่วยความจำที่มีขนาดมากกว่า 256Kbyte นี้เราก็สามารถจะทำได้โดยการเสียบการ์ดที่ใช้ในการเพิ่มหน่วยความจำให้กับ IBM PC/XT ลงบนสล็อตบนเมนบอร์ดของ IBM PC/XT ซึ่งหน่วยความจำที่สามารถจะเพิ่มเข้าไปในระบบนี้จะมีขนาด 384Kbyte คือตั้งแต่แอดเดรส 4000H-9FFFFH (ภายในการ์ดบางรุ่นอาจสามารถเพิ่มจำนวนหน่วยความจำได้มากกว่า 384Kbyte)

นอกจากนี้ยังมีหน่วยความจำ RAM อีกส่วนหนึ่ง ซึ่งเตรียมไว้สำหรับการแสดงผลคือใช้เป็น Display Buffer อีก 128Kbyte จากแอดเดรส 0A0000H-0BFFFFH แต่จะถูกจัดไว้สำหรับการแสดงผลบน Monochrome Card และ Color Graphics เพียงการ์ดละ 32Kbyte เท่านั้น โดยบน Monochrome Card จะจัดไว้ตั้งแต่ 0B8000H-0BFFFFH แต่จะถูกใช้งานจริงเพียง 16 คือจาก 0 B8000H จนถึง 0BBFFFH เท่านั้น

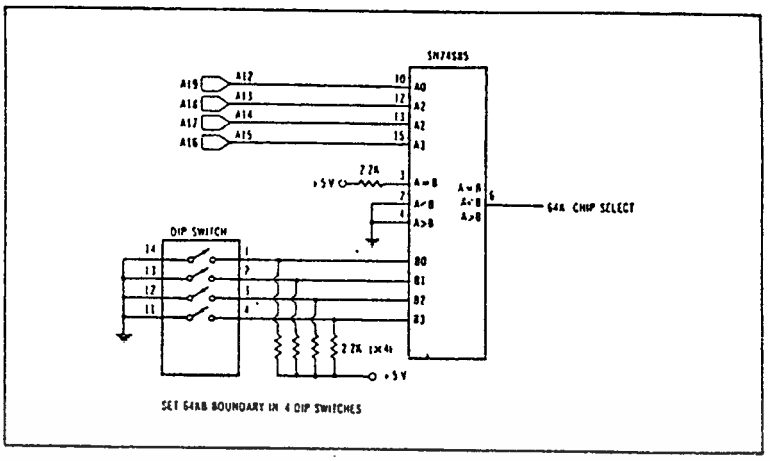
2.9 การดีโค้ดแอดเดรสของหน่วยความจำ

การดีโค้ดแอดเดรสของหน่วยความจำนี้ จะมีลักษณะที่คล้ายคลึงกับวิธีการ ดีโค้ดแอดเดรสของพอร์ท I/O มากอย่างไรก็ตามการดีโค้ดแอดเดรสของหน่วยความจำยังมีข้อแตกต่างจากการดีโค้ดแอดเดรสของพอร์ท I/O อยู่บางประการ กล่าวคือการดีโค้ดแอดเดรสของหน่วยความจำนี้ต้องทำการดีโค้ดแอดเดรสทั้ง 20 บิท เพื่อให้สามารถอ้างแอดเดรสได้ครบทั้ง 1Mbyte และ การดีโค้ดแอดเดรสของหน่วยความจำไม่ใช้สัญญาณ AEN ในการควบคุมวงจรดีโค้ดเหมือนกับในการดีโค้ดแอดเดรสของพอร์ท I/O สำหรับวงจรที่ใช้ในการดีโค้ดแอดเดรสของหน่วยความจำนั้นเราสามารถจะนำเอาการดีโค้ดแบบ Fixed และ แบบสวิทช์เลือกเข้ามาใช้งานได้เช่นกัน



รูปที่ 2.8 ตัวอย่างวงจรที่ใช้ในการดีโค้ดแอดเดรสของหน่วยความจำ

จากรูปที่ 2.8 นั้นจะแสดงตัวอย่างวงจรที่ทำการดีโค้ดแอดเดรสของหน่วยความจำ Static RAM ขนาด 8Kbyte ภายในวงจรนี้จะมีวงจรในส่วนที่เป็น บัฟเฟอร์รวมอยู่ด้วยซึ่งในวงจรจะแสดงถึงในการอีเบิลและควบคุมทิศทาง การส่งผ่านข้อมูลของบัฟเฟอร์เหล่านี้ด้วยอย่างไรก็ตามวงจรดีโค้ดในลักษณะนี้จะใช้ได้ดีสำหรับการดีโค้ดหน่วยความจำที่จำนวนแอดเดรสไม่มากนักแต่ในกรณีที่จำเป็นต้องทำการดีโค้ดบล็อกของแอดเดรส ที่มีจำนวนมากเราอาจจะใช้วงจรในรูป 2.8 ช่วยได้โดยใช้ร่วมกับหน่วยความจำประเภท DRAM



รูปที่ 2.9 ตัวอย่างวงจรดีโค้ดบล็อกแอดเดรสของหน่วยความจำ

จากวงจรในรูปที่ 2.9 เป็นวงจรที่ทำการดีโค้ดบล็อกของแอดเดรสหน่วยความจำขนาด 64 Kbyte โดยตำแหน่งแอดเดรสของบล็อกของหน่วยความจำ ขนาด 64Kbyte นี้อาจจะกำหนให้อยู่ในตำแหน่งใดก็ได้ โดยการเซ็ทตำแหน่งของ DIP Switch นี้ต่ออยู่กับขา B0-B3 และ 74S85

2.10 รูปแบบและความหมายของคำสั่งของเครื่องมือวัดต่างๆ

2.11 HM8142 PROGRAMABLE POWER SUPPLY

HM8142 เป็นแหล่งจ่ายไฟกระแสตรงที่สามารถควบคุมจากภายนอกผ่านทาง GBIB ได้

คำสั่งของ HM8142

HM/RMO	:รีโมต/ไม่มีรีโมต
LK1/LK0	:LOCAL
SU1	:ตั้งค่าแรงดันส่วนที่ 1 SET VOLTAGE 1
SU2	:ตั้งค่าแรงดันส่วนที่ 2 SET VOLTAGE 2
SI1	:ตั้งค่ากระแสส่วนที่ 1
SI2	:ตั้งค่ากระแสส่วนที่ 2
RU1	:เรียกดูค่าแรงดันส่วนที่ 1 ที่ตั้งไว้
RU2	:เรียกดูค่าแรงดันส่วนที่ 2 ที่ตั้งไว้
RI1	:เรียกดูค่ากระแสส่วนที่ 1 ที่ตั้งไว้
RI2	:เรียกดูค่ากระแสส่วนที่ 2 ที่ตั้งไว้
MU1	:วัดค่าแรงดันจริงของส่วนที่ 1
MU2	:วัดค่าแรงดันจริงของส่วนที่ 2

MI1 :วัดค่ากระแสจริงของส่วนที่ 1

MI2 :วัดค่ากระแสจริงของส่วนที่ 2

MX1/MX0 :ให้สามารถ/ไม่สามารถ ควบคุมจากมือและจาก GPIB พร้อมกันได้

OP1/OP0 :ให้เอ้าพุทออก/ไม่ออก

TRU :ใช้การปรับค่าแรงดันแบบพร้อมกันทั้ง 2 ส่วน

TRI :ใช้การปรับค่ากระแสแบบพร้อมกันทั้ง 2 ส่วน

SR1/SR0 :ให้สามารถส่ง SERVICE REQUEST ได้/ไม่ได้

ABT1 :ARBITARY WAVEFRAM MODE

RUN :START ARBITARY

STP :STOP ARBITARY

ABX :EXIT ARBITARY

VER :ให้สั่งชื่อรุ่น

STA :ให้ส่งสถานะของตัวเอง

ID? :ให้ส่งชื่อของเครื่อง

RM1/RM0

รูปแบบ :RM1

หน้าที่ :เป็นการให้ HM8142 มาอยู่ในสภาวะรีโมตที่ทำการควบคุมโดยผ่านระบบGPIBทำให้ปุ่มควบคุมบนหน้าปัดไม่สามารถใช้งานได้ สามารถเลิกคำสั่งได้โดย RM0 หรือกดปุ่ม LOCAL

รูปแบบ :RM0

หน้าที่ :ใช้ยกเลิกการรีโมต ให้ HM8142 มาอยู่ในสภาวะ LOCAL หรือ การควบคุมด้วยมือ

MX1/MX0

รูปแบบ :MX1

หน้าที่ :เปลี่ยน HM8142 จากสถานะรีโมตมาสู่แบบผสม คือสามารถคุมได้ทั้งจาก GPIB และจากหน้าปัด

รูปแบบ :MX0

หน้าที่ :เปลี่ยน HM8142 จากสถานะแบบผสมมาสู่การรีโมตแบบปกติ

LK1/LK0

รูปแบบ :LK1

หน้าที่ :เปลี่ยน HM8142 มาสู่รีโมตแบบบล็อก ซึ่งจะทำให้ปุ่ม LOCAL ไม่สามารถใช้บังคับให้ HM8142 กลับไปสู่สภาวะ LOCAL ได้

รูปแบบ :LK0

หน้าที่ :ให้ HM8142 ออกจากสภาวะรีโมตแบบบล็อกมาสู่สภาวะรีโมตแบบธรรมดา

SU1/SU2

รูปแบบ :SU1 VV.mVmV or SU2 12.34

หน้าที่ :ตั้งค่าแรงดันส่วนที่ 1 หรือ ส่วนที่ 2

ตัวอย่าง :SU1 4.02 ผล : แรงดัน U1 มีค่าเป็น 4.02

:SU2 5.65 ผล : แรงดัน U2 มีค่าเป็น 5.65

SI1/SI2

รูปแบบ :SI1 A.mAmAmA or Su2 0.543

หน้าที่ :ตั้งค่ากระแสสูงสุด (Current limit) ส่วนที่ 1 หรือ ส่วนที่ 2

ตัวอย่าง :SI1 1.00 ผล : กระแสสูงสุดของ U1 มีค่าเป็น 1.00 A

RU1/RU2

รูปแบบ :RU1 หรือ RU2

ค่าที่ตอบกลับ :U1:12.34V หรือ U2:5.65V

หน้าที่ :เรียกดูค่าแรงดันส่วนที่ 1 หรือ ส่วนที่ 2 ที่ตั้งไว้ ทำให้ HM8142 กลับมาสู่สภาวะเริ่มต้นคือค่าแรงดันและกระแสเป็นศูนย์ เอ้าพุทไม่จ่ายออก

VER

รูปแบบ :VER

ค่าที่ตอบกลับ :swVx.xhwVx.xxxxxxxHAMEG/Paris KPR&VM

หน้าที่ :ให้แสดงรุ่นของ HM8142

ID?

รูปแบบ :ID?

ค่าที่ตอบกลับ :HM8142-1

หน้าที่ :ให้แสดงชื่อ HM8142

2.12 HM8112-2 PROGRAMMABLE MULTIMETER

สามารถใช้กับ Keyboard และ IEEE-488 Bus ได้

HM8112-2 PRG7IEEE Setting of device address and termination

กรณีที่เราใช้ PRG7 เครื่องจะอยู่ในสถานะ "set device address" หน้าจอเครื่องจะแสดงผล เช่น "IEEE.07.8" หมายถึงเครื่องนี้มี address = 7 และใช้ End character แบบที่ 8 โดยเราสามารถเปลี่ยน address ของอุปกรณ์ได้โดยการกดปุ่ม "UP" ซึ่งเราสามารถเลือกค่า address ได้ตั้งแต่ 0 ถึง 30 หลังจาก 30 จะเป็น "-" แสดงว่าเครื่องอยู่ในสถานะ "TALK ONLY" และเครื่องจะสิ้นสุดกระบวนการเมื่อกดปุ่ม "UP" อีกครั้ง นอกจากนี้เรายังสามารถเลือก End character โดยการกด "DOWN"

ความหมายของ End character เป็นค่าของตัวอักษรที่ใช้ปิดท้ายข้อมูลซึ่ง HM112-2 มีได้ 8 แบบ ดังตาราง

แบบที่	แบบของตัวอักษรปิดท้าย
0	CR + EO1
1	CR
2	LF + EO1
3	LF
4	CR +LF + EO1
5	CR + LF
6	LF + CR + EO1
7	LF + CR
8	EO1

ตารางที่ 2.1

คำสั่งสำหรับ HM8112-2

VD เลือกการวัดค่าแรงดันไฟตรง

VA เลือกการวัดค่าแรงดันไฟสลับ

O2 เลือกการวัดค่าความต้านทานแบบ 2 สาย (ถ้าวัดแบบ 4 สายใช้ O4)

ID เลือกการวัดค่ากระแสไฟตรง

IA เลือกการวัดค่ากระแสไฟสลับ

TC,TK,TF เลือกการวัดค่าอุณหภูมิ

RX กำหนด RANGE สำหรับค่าต่าง ๆ ในการวัดตามตาราง

RI1/RI2

รูปแบบ :RI1 หรือ RI2

ค่าที่ตอบกลับ :I1_1.00A หรือ 12_0.012A

หน้าที่ :เรียกดูค่ากระแสสูงสุดส่วนที่ 1 หรือส่วนที่ 2 ที่ตั้งไว้

MU1/MU2

รูปแบบ :MU1 หรือ MU2

ค่าที่ตอบกลับ :U1:12.34V หรือ U2:5.00V

หน้าที่ :วัดค่าแรงดันจริงของส่วนที่ 1 หรือ 2

MI1/MI2

รูปแบบ :MI1 หรือ MI2

ค่าที่ตอบกลับ :I1+1.000A หรือ I2-0.565A

หน้าที่ :วัดค่ากระแสจริงของส่วนที่ 1 หรือ 2

TRU

รูปแบบ :TRU VV.mVmV

หน้าที่ :ใช้การปรับค่าแรงดันแบบพร้อมกันทั้งสองส่วน

ตัวอย่าง :TRU 12.34 ผล : แรงดัน U1 = U2 = 12.34V

TRI

รูปแบบ :TRI A.mAmAmA

หน้าที่ :.ใช้การปรับค่ากระแสสูงสุดแบบพร้อมกันทั้งสองส่วน

SR1/SR0

รูปแบบ :SR1

หน้าที่ :อนุญาตให้ส่ง SERVICE REQUEST

รูปแบบ :SR0

หน้าที่ :ไม่ควรอนุญาตให้ส่ง SERVICE REQUEST

STA

รูปแบบ :STA

ค่าที่ตอบกลับ :OP1/0 SQ1/0 RE0/1 CV1/CC1 CV2/CC2 RM0/1

หน้าที่ :ตรวจสอบสภาพของ HM 8142

OP1/OPO

รูปแบบ :OP1

หน้าที่ :ให้เอ้าพุทออก

รูปแบบ :OP2

หน้าที่ :ให้เอ้าพุทไม่ออก

Clear

รูปแบบ :Clr

Range	V_{dc}/V_{ac}	Resistance (kΩ)	I_{ac}/I_{dc} (mA)
R1	0.2	0.2	-
R2	2	2	2
R3	20	20	-
R4	200	200	-
R5	1000	2000	2000
R6	-	12000	-

ตารางที่ 2.2

A0/A1 กำหนดสถานะ AUTORANGE โดย A1 เป็นการให้ AUTORANGE และ A0 (ศูนย์) เป็นแบบ MANUAL

TX กำหนด integration time และจำนวนหลัก (digits) ของตัวเลขในการแสดงผล

ZO เริ่มกระบวนการแก้ค่า offset

S0 (ศูนย์) START THE CONTINUOUS MEASURING SEQUENCE.

S1 STOP THE CONTINUOUS MEASURING SEQUENCE.

MO SELECT THE MULTIPLEXER CHANNEL
"F" = MULTIPLEXER OFF.

M0-M9 SELECTION OF CHANNELS 0-9

L0 (ศูนย์)	เลือกการส่งค่าแบบ SHORT FORMAT
L1	เลือกการส่งค่าแบบ LONG FORMAT
Q0 (ศูนย์)	ไม่อนุญาตให้ Multimeter ส่ง SQR (SERVICE REQUEST)
Q1	อนุญาตให้ Multimeter ส่ง SQR (SERVICE REQUEST)
NVXXXXXXXX	กำหนดค่าในการ calibration ผ่าน bus ของ IEEE - 488
P1	แสดงค่า offset ที่แก้ไขแล้ว R=X-C
P2	แสดงค่า %DEVIATION R=100*(X-C)/C
P3	แสดงค่า dB R=20*log (x/C)
P4	แสดงค่า dBru R=20*log (X/C) WITH

(C = 0.775 V. INTO 600 Ohm FOR VOLTAGES AND C = 1.29 mA FOR CURRENT)

PxEN	ENTER MEAS.VALUE FOR CONSTANT x AT P2-4; X=1,2,3
ID?	ให้ส่งชื่อของเครื่อง "HM8112-2"
STA?	ให้ส่งสถานะของเครื่อง (2ND DATA BLOCK)
D0/D1	0 = DISPLAY OFF; 1 = DISPLAY ACTIVE
Bx	RETURN THE CODE OF THE SWITCH, WHICH WAS DEPRESSED LAST.X=1,.....,9,A....F
EOI	EOI ACTIVE WIIEN SELECT
EOS1/EOS2	TERMINATES THE DEVICE MESSAGE; CODE 0,.....,8 SELECTED DETERMINE WHETHER THE STRING IS SENT WITH 1 OR 2 (EOS1/EOS2) TERMINATORS
END	TERMINATING CHARACTER (S) AS SELECT WHEN SETTING DMM ADDRESS, FOR TERMINATING CHA-RECTER NO.8, EOI IS TRANSMITTED TOGERTHER WITH THE FINAL (26th) STRING CILARACTER.

รูปแบบข้อมูลที่ส่งออกมา

+X.XXXXXXXE+X	VD	R1	A0	T1	S0	Q0	M0	X0	P0	B0
	VA	R2	A1	T2	S1	Q1	M1	X1	P0	B1
	O2	.		T3			M1	X2		.
	O4	.		T4			.	X3		.
	ID	.					.	X4		.
	LA	R6					M9			B9
	TC									
	TF									
	TK									

+/- :เครื่องหมายของ VD และ ID ซึ่งจะเป็นศูนย์สำหรับ VA<IA และ O2

X.XXXXXXX :ค่าที่วัดได้ ได้ 7 หลัก

E+X :ค่ายกกำลังของค่าที่วัดได้

VD,VA,O2,ID,LA :ฟังก์ชันที่วัด

SERVICE REQUEST FUNCTION (SRQ)

เมื่อฟังก์ชัน SRQ ทำงานแต่ละบิทจะมีความหมายดังนี้

BIT1 END OF MEASUREMENT

BIT3 OVERFLOW DURING MEASUREMENT

BIT4 ERROR MESSAGE

BIT6 RESET

BIT7 SRQ

2.13 HM8130 FUNCTION GENERATER

สามารถโปรแกรมอุปกรณ์ตัวนี้ผ่านการ interface แบบ IEEE-488 หรือ RS-232

คำสั่งที่ใช้กับ HM8130 โดยผ่าน IEEE-488

คำสั่งที่ไม่มีข้อมูล (Command without data)

SIN	เลือกรูปคลื่น Sine
TRI	เลือกรูปคลื่น Triangle
SQR	เลือกรูปคลื่น Squarc
PLS	เลือกรูปคลื่น Pulse
RMP	เลือกรูปคลื่น Sawtooth แบบ positive-going
RMN	เลือกรูปคลื่น Sawtooth แบบ negative-going
ARB	เลือกรูปคลื่น Arbitary
SW1/0	Sweep mode On/Off
CTM	Continuous mode
GTM	Gated mode
TRM	Triggered mode
OT1/0	ให้สัญญาณเอ้าพุท ออก/ไม่ออก
OF1/0	ให้มี/ไม่มี Offset
DFR	ให้แสดงความถี่สัญญาณ
DST	ให้แสดงความถี่เริ่มต้นในการกวาด
DSP	ให้แสดงความถี่สุดท้ายในการกวาด
DWT	ให้แสดง Pulse Width
DSW	ให้แสดงเวลาในการกวาด
DAM	ให้แสดง output amplitude
DOF	ให้แสดง offset voltage
RMO	Disable REMOTE status
LK1	Enable local-inhibit status
LK0	Disable local-inihibit status
TRG	Triggers a signal period(sweep off) หรือ complete sweep(sweep on)
CLR	Reset HM8130 (เหมือนคำสั่ง SDC ใน IEEE-488)

ARC	Delete all arbitary data และ Reset internal arbitary couter เป็น 0
ARE	Torminal the arbitary oditor

Dcfult scttings:

Frequoncy	1 kHz
Start Frequency	2 kHz
Stop Frequency	10 kHz
Amplitude	10 V.
Offset voltage	1 V.
Sweep time	100 ms.
Pulse width	50 us
signal shape	sine
sweep	off
offset	disable
positive pulse	
positive-going sawtooth	

คำสั่งที่เกี่ยวกับ floating-point data

FRQ:<Data>	ตั้งค่าความถี่เป็น <.....>Hz*
STT:<Data>	ตั้งค่าความถี่เริ่มต้นในการกวาดเป็น <.....>Hz*
STP:<Data>	ตั้งค่าความถี่สุดท้ายในการกวาดเป็น <.....>Hz*
SWT:<Data>	ตั้งค่าเวลาในการกวาดเป็น <.....>s*
WDT:<Data>	ตั้งค่าความกว้างของ pulse เป็น<.....>s*
AMP:<Data>	ตั้งค่า Amplitude เป็น<.....>V**
OFS:<Data>	ตั้งค่าแรงดัน offset เป็น<.....>V**
หมายเหตุ:	* หมายถึง จำนวนหลักสูงสุด 5 หลัก
	** หมายถึง จำนวนหลักสูงสุด 3 หลัก

คำสั่งที่เกี่ยวกับ whole-number vauler (จำนวนหลักสูงสุด 4 หลักและเครื่องหมาย +,-)

STO=<Data>	Store values <0...8>

RCL=<Data>	Read value <0...8>
ARB=<Data>	อ่านค่าข้อมูล arbitary แล้วเพิ่มค่าโดยข้อมูลจะอยู่ในช่วง -511 ถึง +512

คำสั่งขอข้อมูล (request command)

FRQ?	ถามค่าความถี่
STT?	ถามค่าความถี่เริ่มต้นในการกวาด
STP?	ถามค่าความถี่สุดท้ายในการกวาด
SWT?	ถามค่าเวลาในการกวาด
DWT?	ถามค่า Pulse Width
AMP?	ถามค่า Outpt Voltage
OFS?	ถามค่า Ofest
ARD?	ถามค่า Arbitary Data
ID?	ถามชื่อของเครื่อง
VER?	ถาม Version ของเครื่อง
STA?	ถามสถานะของเครื่อง

คำสั่ง CLR (CLR Command)

คำสั่งสำหรับถามสถานะของ Front Panel ผลที่ได้จะเป็น string ยาว 21 ตัวอักษรซึ่งมีลำดับดังนี้ (ช่องว่างเดิมไว้เพื่อให้อ่านง่ายเท่านั้น ไม่มีอยู่จริงในข้อมูล)

OT0	OF0	SW0	SIN	CTM	DFR	DAM
1	2	3	4	5	6	7

1	Output on/off
2	Offset on/off
3	Sweep on/off
4	Signal type
5	Operating mode
6	Display contents (right)
7	Display contents (left)

บทที่ 3
อุปกรณ์ที่ใช้ในการทดลอง

อุปกรณ์ที่ใช้ในการทดลองนี้มี 2 ชนิดคือ

- ทรานซิสเตอร์
- ทรานซิสเตอร์สนามไฟฟ้ามอสเฟต

3.1 ทรานซิสเตอร์ (Transister)

ทุกวันนี้ได้มีการใช้ทรานซิสเตอร์อย่างกว้างขวางออกไปทุกจึงเป็นการยากที่ทุกคนจะสามารถศึกษาและเข้าใจวงจรทุกๆวงจรได้ครบถ้วนสิ่งสำคัญและจำเป็นที่สุดสำหรับผู้เริ่มต้นคือต้องศึกษาวงจรพื้นฐานให้เข้าใจพอที่จะนำไปอธิบายหรือใช้วิเคราะห์วงจรแบบอื่นๆได้
เมื่อเราใช้ทรานซิสเตอร์เป็นตัวขยายสัญญาณหรือเป็นตัวควบคุมสัญญาณหรือเป็นตัวกำเนิดสัญญาณใดๆก็ตามสิ่งแรกที่เราต้องทำกับทรานซิสเตอร์ในวงจรเหล่านั้นก็คือการใช้ไบแอส(Bias)กับมันซึ่งก็คือให้ไฟตรงมาเลี้ยงทรานซิสเตอร์ให้ทำงานอยู่ในช่วงที่เราต้องการนั่นเอง

ตามที่ทราบมาแล้วในบทก่อนว่าช่วงการทำงานของทรานซิสเตอร์ทั่วไปมี3ช่วงคือช่วงคัทออฟ (Cut off) ช่วงนี้จะไม่มีกระแสไหลผ่านทรานซิสเตอร์หรือมีก็เป็นกระแสรั่วไหลซึ่งบางครั้งเราเรียกว่าทรานซิสเตอร์หยุดเฉย ช่วงต่อไปคือช่วงอิ่มตัว (Saturation region) ช่วงนี้จะมีกระแสไหลผ่านทรานซิสเตอร์เต็มที่จนมันอิ่มตัวคือกระแสจะไม่มากไปกว่านี้อีกแล้ว(กระแสอาจถูกจำกัดด้วยค่าตัวต้านทานภายนอก)ช่วงสุดท้ายคือช่วงแอกทีฟ(Activeregion)ช่วงนี้ทรานซิสเตอร์จะทำงานอยู่ระหว่างสองช่วงที่แล้ว โดยทำหน้าที่ให้กระแสคอลเลคเตอร์ที่เป็นสัดส่วนกับกระแสเบสในการใช้งานของทรานซิสเตอร์ในวงจรขยายทั่วไปมักใช้ในช่วงแอคทีฟฉะนั้นการไบแอสทรานซิสเตอร์ที่จะกล่าวถึงต่อไปจะเป็นการกล่าวถึงไบแอสทรานซิสเตอร์ในช่วงนี้เท่านั้น

3.2 กราฟแสดงการทำงานของทรานซิสเตอร์

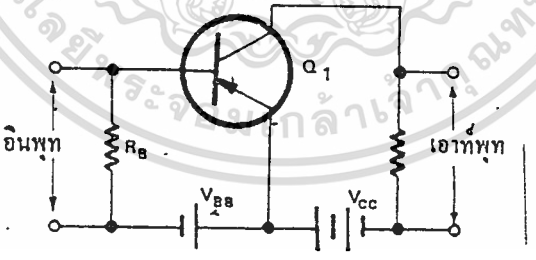
ตามที่ได้กล่าวมาแล้วทรานซิสเตอร์ที่มีใช้งานทั่วไปจะต้องมีลักษณะสมบัติโดยเฉพาะตัวที่จะให้คุณค่าคุณสมบัติหรือพารามิเตอร์แตกต่างกันเช่นทรานซิสเตอร์แบบเดียวกันยังมีค่าแอลฟ่า(α)และค่าเบต้า(β)ไม่เท่ากันแต่อย่างไรก็ดีเมื่อนำตัวทรานซิสเตอร์ไปต่ออยู่ในวงจรโดยให้ค่าองค์ประกอบของวงจรเป็นตัวกำหนดค่าพารามิเตอร์บางตัวจะทำให้เราสามารถที่จะกำหนดลักษณะ

ของวงจรได้นั่นคือวงจรจะต้องให้ค่ากระแสและแรงดันไบแอสที่มีค่าเฉพาะและค่าเหล่านี้จะแสดงให้เห็นถึงการทำงานของวงจรทรานซิสเตอร์คือ

1. แรงดันไบแอสระหว่างเบส-อิมิตเตอร์
2. แรงดันไบแอสระหว่างคอลเลคเตอร์-เบส
3. กระแสไบแอสที่มีอีมิตเตอร์
4. กระแสเบส
5. กระแสไบแอสที่คอลเลคเตอร์
6. ความต้านทานอินพุท
7. ความต้านทานเอาท์พุท

ทรานซิสเตอร์แต่ละตัวจะมีลักษณะสมบัติประจำตัวมันอยู่แล้วซึ่งเขียนเป็นกราฟลักษณะสมบัติได้และสามารถนำมารับการกำหนดจุดทำงาน ที่มีค่าแรงดันและกระแสแตกต่างกันออกไป ซึ่งแต่ละค่าของแรงดันและกระแสก็จะต้องสัมพันธ์กับกราฟลักษณะสมบัติการทำงานของทรานซิสเตอร์นั่นคือเราสามารถกำหนดจุดการทำงานลงบนกราฟได้นั่นเอง

กราฟลักษณะสมบัติของทรานซิสเตอร์ที่มีประโยชน์ในการหาลักษณะการทำงานของทรานซิสเตอร์คือกราฟลักษณะสมบัติทางคอลเลคเตอร์ที่บอกค่าความสัมพันธ์ของกระแสคอลเลคเตอร์ กระแสเบสและแรงดันคอลเลคเอตร์- อิมิตเตอร์

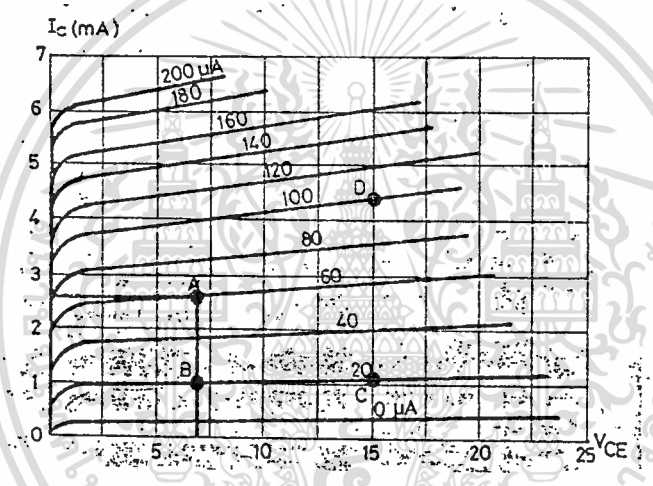


รูปที่ 3.1 แสดงวงจรการทำงานของทรานซิสเตอร์
V_{BB} และ V_{CC} เป็นแรงดันใช้ในการไบแอส

สมมติว่าทรานซิสเตอร์ตัวหนึ่งเมื่อต่อในวงจรให้ทำงานในลักษณะของวงจรขยายสัญญาณขั้วระหว่างรอยต่อทางด้านเบส-อิมิดิเตอร์จะต้องต่อในลักษณะไบแอสตรงและถ้าวงจรต่อในลักษณะอิมิตเตอร์ร่วมการต่อไบแอสที่ขั้วคอลเลคเตอร์จึงต่อให้ทิศทางของกระแสไหลผ่านคอลเลคเตอร์ได้ตามปกติไปยังอิมิตเตอร์ด้วยค่าแรงดัน V_{CC} ดังรูปที่ 3.1

จากวงจรที่3.1 ถ้าสมมติว่าป้อนกระแสเบส I_B ที่มีค่าคงที่ค่าหนึ่ง(กระแสเบสจะขึ้นอยู่กับค่าแรงดันไบแอสตรงที่ขั้วเบสและอิมิตเตอร์ V_{BE}) จะมีค่าคอลเลคเตอร์ไหล การไหลของกระแสคอลเลคเตอร์จะขึ้นอยู่กับค่าแรงดันคอลเลคเตอร์ V_{CE} และกระแสเบส นั่นคือความสัมพันธ์ของกระแสคอลเลคเตอร์ I_CและกระแสเบสI_Bจะต้องเป็นไปตามค่าพารามิเตอร์ของทรานซิสเตอร์

จากกราฟลักษณะสมบัติที่แทนตัวทรานซิสเตอร์ (ดูรูปที่3.2) การที่เราจะกำหนดจุดทำงานที่จุดใดจุดหนึ่งบนกราฟได้จุดนั้นจะแสดงถึงค่าความสัมพันธ์ของตัวพารามิเตอร์ 3 ตัวที่เกี่ยวข้องกันคือ กระแสเบส I_B กระแสคอลเลคเตอร์ I_C และแรงดันอิมิตเตอร์-คอลเลคเตอร์ V_{CE}



รูปที่3.2 กราฟแสดงการทำงานของทรานซิสเตอร์จุดทำงาน A แสดงว่าเมื่อมีกระแสเบส จะเกิดแรงดันคร่อมระหว่างคอลเลคเตอร์ อิมิตเตอร์ V_{CE}7 โวลท์ และให้กระแสคอลเลคเตอร์ มีค่า 2.5 มิลิแอมแปร์

เมื่อทราบค่าจุดทำงานของทรานซิสเตอร์เราอาจจะหากระแสส่วนอื่นๆ ที่เหลือและแรงดันที่ตัวทรานซิสเตอร์ได้ ซึ่งเราจะทราบว่าภาวะการทำงานขณะนี้จะเป็นอย่างไร

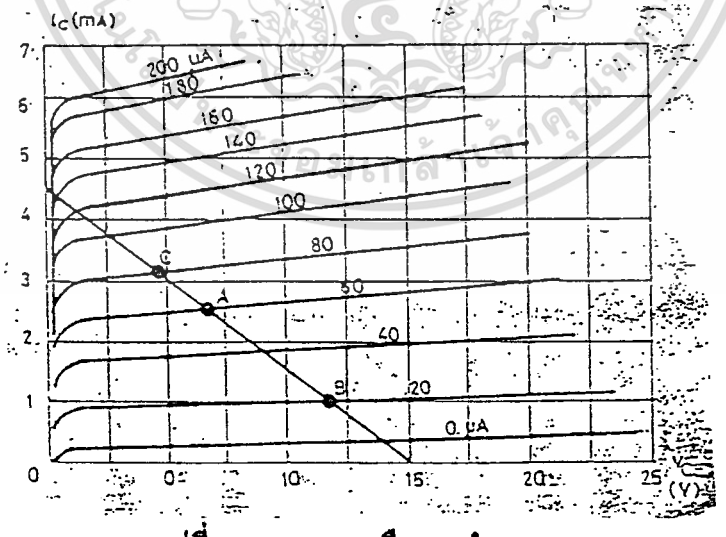
สมมติว่าถ้าใช้วงจรดังที่แสดงให้เห็นแล้วในรูปที่3.1และเปลี่ยนค่าความต้านทานระหว่างเบส-อิมิตเตอร์ ที่อยู่ในวงจรเพื่อที่จะให้กระแสเบสไหลมีค่าเป็น 20ไมโครแอมแปร์นั่นคือกระแสเบสเปลี่ยนแปลงจาก 60 ไมโครแอมแปร์ มาเป็น 20 ไมโครแอมแปร์ ซึ่งจะเป็นผลทำให้แรงดันคอลเลคเตอร์อิมิตเตอร์มีค่าเป็น 7โวลท์และได้จุดทำงานแห่งใหม่คือที่ B เมื่อพิจารณาจากกราฟจะเห็นว่า ในขณะนี้จะมีกระแสคอลเลคเตอร์ไหลในวงจรเป็น 1 มิลิแอมแปร์และทำนองเดียวกันถ้าให้กระแสเบสไหลมีค่าคงที่ที่ 20 ไมโครแอมแปร์แต่เปลี่ยนค่าแรงดันระหว่างคอลเลคเตอร์ - อิมิตเตอร์ไปเป็น15โวลท์กระแสคอลเลคเตอร์จะเปลี่ยนตำแหน่งไปเป็นจุดC ซึ่งมีกระแสคอลเลคเตอร์

ไหลเกินกว่า 1 มิลลิแอมแปร์เล็กน้อยซึ่งแสดงให้เห็นว่าแรงดันระหว่างคอลเลคเตอร์ - อิมิตเตอร์มีผลต่อการไหลของกระแสคอลเลคเตอร์เพียงเล็กน้อยเท่านั้น ถ้าให้กระแสเบสมีค่าคงที่

สมมติว่าถ้าปรับแรงดัน15โวลท์เข้าระหว่างขั้วคอลเลคเตอร์และอิมิตเตอร์แล้วทำการปรับค่าความต้านทาน R_Bจนกระทั่งได้กระแสเบสมีค่า100ไมโครแอมแปร์(จุด D)จะพบว่ากระแสคอลเลคเตอร์จะมีค่าประมาณ 4.33 มิลลิแอมแปร์ซึ่งจากจุดทำงาน2จุดคือจุดCและจุดD แสดงให้เห็นว่าเมื่อเปลี่ยนแปลงกระแสเบสไป80ไมโครแอมจะเป็นผลทำให้กระแสคอลเลคเตอร์เปลี่ยนแปลงไปถึง3.25 มิลลิแอมหรือ3250ไมโครแอมแปร์ซึ่งเทียบเท่ากับค่าอัตราขยายเบต้าเท่ากับ 40

3.3 การเลือกจุดทำงาน

ในการกำหนดจุดทำงาน จุดทำงานบางจุดจะให้ผลดีกว่าจุดอื่นๆ การจะเลือกจุดทำงานที่ใดในกราฟลักษณะสมบัติขึ้นอยู่กับจุดประสงค์ของวงจร ดังตัวอย่างเช่น ในวงจรขยายที่ต้องการให้แรงดันเอาท์พุทแกว่งมีค่า 10 โวลท์จากจุดต่ำสุดถึงจุดสูงสุดเราจะต้องเลือกจุดทำงานเพื่อแสดงให้กระแสคอลเลคเตอร์เปลี่ยนแปลงอยู่ระหว่าง 1 -14 โวลท์ โดยให้แรงดันสามารถแกว่งได้ข้างละ 5 โวลท์ในแต่ละด้านของจุดทำงานการแกว่งจะต้องไม่ถึงจุดสูงสุดของเส้นสมการ โหลดหรืออยู่ในสภาวะที่เรียกว่าคัทออฟหรืออิ่มตัว บนกราฟลักษณะสมบัติของทรานซิสเตอร์



รูปที่ 3.3 แสดงการเลือกจุดทำงาน

สมมติถ้าเราเลือกจุด B เป็นจุดทำงานของตัวทรานซิสเตอร์ที่จุดนี้จะเป็นผลทำให้แรงดันคอลเลคเตอร์อิมิตเตอร์มีค่าประมาณ12โวลท์ถ้าให้สัญญาณแกว่งข้างละ 5 โวลท์จะเห็นว่าในด้านหนึ่งของสมการเส้นโหลดแรงดันสามารถแกว่งมาได้เพียง3โวลท์สูงสุดเท่านั้นและสัญญาณจะถูก

ขลิบหลังจากที่แกว่งเกิน 3 โวลท์ ดังนั้นสัญญาณเอาท์พุทที่ปรากฏออกมาจึงมีลักษณะเพี้ยนไปจากเดิมแต่ในบางวงจรเราอาจจะนำเอาลักษณะสมบัติข้อนี้ไปใช้ในการทำเป็นวงจรขลิบสัญญาณในกรณีที่เราต้องการให้ขยายสัญญาณได้สูงสุดในกรณีที่สัญญาณเอาท์พุทที่มีการแกว่งในทางเดียว เช่นสัญญาณพัลส์บวกการเลือกจุดทำงานจุด B ก็จะเหมาะสม

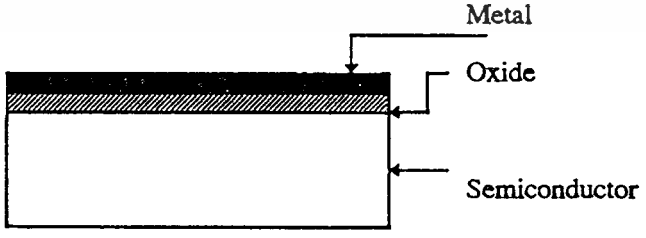
หรือถ้าสัญญาณพัลส์นั้น เป็นสัญญาณพัลส์ลบการเลือกจุดทำงานที่ C ก็จะเหมาะทั้งนี้เพราะสัญญาณจะแกว่งไปได้สูงสุดหรือมีขอบเขตการทำงานได้กว้าง

เมื่อพิจารณาดูจุดทำงานที่ A ซึ่งมีกระแสเบสไหลในวงจร 60 ไมโครแอมแปร์ ณ จุดนี้จะเป็นจุดกึ่งกลางของช่วงการทำงานของทรานซิสเตอร์ ซึ่งเป็นผลทำให้ในกรณีที่สัญญาณเข้าเป็นสัญญาณสมมาตรสัญญาณสามารถแกว่งไปยังจุดสูงสุดและต่ำสุดได้เท่ากันเป็นผลทำให้การทำงานได้เต็มที่โดยไม่มีการขลิบด้านใดด้านหนึ่งก่อน

3.4 มอสเฟต MOSFET

สิ่งประดิษฐ์สารกึ่งตัวนำประเภท MOS ถูกจัดอยู่ในกลุ่มของสิ่งประดิษฐ์สารกึ่งตัวนำที่มีการควบคุมการทำงานด้วยผลสนามไฟฟ้า (field Effect Devices)ซึ่งมีลักษณะเด่นคือ มีความต้องการพลังงานในขณะใช้งานแต่มีประสิทธิภาพในการทำสูง เมื่อเปรียบเทียบกับสิ่งประดิษฐ์สารกึ่งตัวนำประเภทอื่นๆ ที่มีลักษณะการใช้งานเช่นเดียวกัน จึงทำให้เป็นที่นิยมใช้กันทั่วไปโดยเฉพาะในวงจรอิเล็กทรอนิกส์ที่ต้องการพลังงานต่ำ หรือในโครงสร้างของวงจรรวม (Integrated Circuits IC) ที่มีจำนวนประกอบ (Component) มากๆ เช่น IC ระดับ LSI และ VLSI เป็นต้น

คำว่า MOS ย่อมาจาก Metal Oxide Semiconductor ซึ่งบอกถึงลักษณะโครงสร้างดังแสดงในรูปที่ 3.4



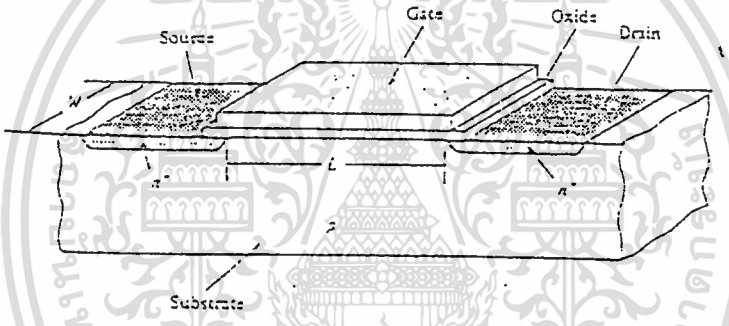
รูปที่ 3-4 แสดงโครงสร้างของ MOS

จะเห็นได้ว่าลักษณะดังกล่าวนี้เหมือนกับตัวเก็บประจุ (Capacitor) สิ่งประดิษฐ์สารกึ่งตัวนำประเภท MOS ที่เป็นรู้จักกันดีได้แก่ MOSFET (Metal Oxide Semiconductor Field Effect transistor) ซึ่งมีการทำงานด้วยประจุไฟฟ้าเพียงชนิดเดียว (Unipolar)

3.5 ลักษณะโครงสร้างของ MOSFET

MOSFET ประกอบด้วย 4 ส่วน ที่สำคัญดังแสดงในรูปที่ 3-5 คือ

1. ส่วนเดรน (Drain Region : D)
2. ส่วนซอส (Source Region : S)
3. ส่วนเกท (Gate : G)
4. ฐานรอง (Substrate : Sub.)



รูปที่ 3.5 แสดงส่วนประกอบของ MOSFET

โดยที่ D และ S จะเป็นสารกึ่งตัวนำชนิดเดียว (เช่น n-type Semiconductor) แต่จะต่างชนิดกับ Sub. (เช่น Sub. นั้นเป็น p-type Semiconductor แต่ D และ S เป็น n-type Semiconductor) สำหรับส่วน G จะใช้เป็นตัวควบคุมการเหนี่ยวนำประจุให้สะสมที่ผิวหน้าของฐานรองบริเวณช่องระหว่าง D และ S เพื่อให้เกิดเป็นช่องทางเดินกระแส (Channel) หรือไม่ให้เกิดตามต้องการ

3.7 ชนิดของ MOSFET

ถ้าแบ่งตามลักษณะการทำงานจะ แบ่งได้ 2 ชนิด คือ

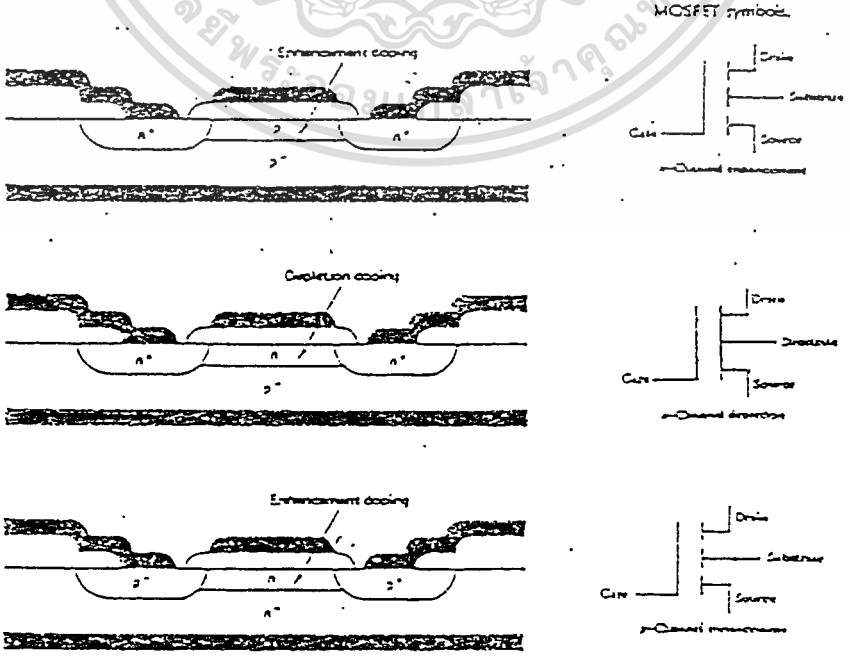
1. แบบ Enhancement คือ MOSFET ที่บริเวณช่องทางกระแสมีคุณสมบัติเป็นสารกึ่งตัวนำชนิดเดียวกับฐานรอง

2. แบบ Depletion คือ MOSFET บริเวณช่องทางเดินกระแสมี คุณสมบัติเป็นสารกึ่งตัวนำชนิดเดียวกับส่วนเดรนและซอส

ถ้าแบ่งตามชนิดของประจุไฟฟ้าที่เกิด จะแบ่งได้เป็น 2 ชนิดคือ

1. N-Channel MOSFET (NMOS) คือ MOSFET ที่ใช้การเคลื่อนที่ของประจุลบหรืออิเลคตรอนจึงกำหนดให้ส่วน D และ S เป็นสารกึ่งตังนำชนิดเอ็น (n-type Semiconductor) ดังนั้นถ้ากระแสอิเลคตรอนจะเคลื่อนที่ระหว่าง D และ S ได้บริเวณช่องทางเดินกระแสจะต้องมีสภาพเป็นสารกึ่งตัวนำชนิดเอ็นด้วย

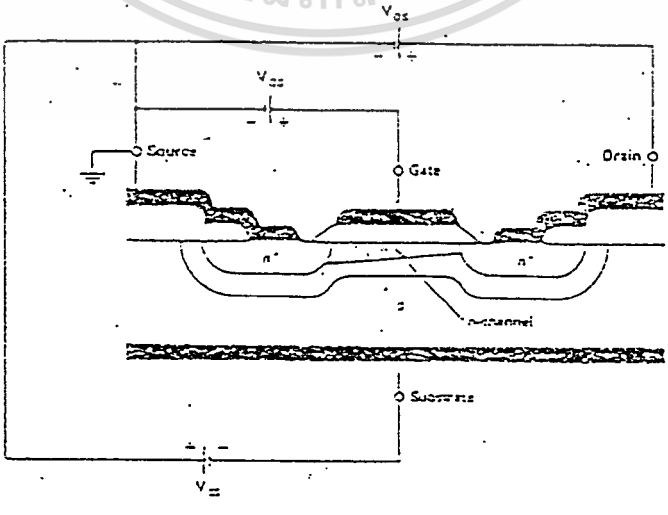
2. P-Channel MOSFET (PMOS) คือ MOSFET ที่ใช้การเคลื่อนของประจุบวกหรือโฮล จึงกำหนดให้ส่วน D และ S เป็นสารกึ่งตัวนำชนิดพี (P-type Semiconductor) ดังนั้นถ้ากระแสโฮลจะเคลื่อนที่ระหว่าง D และ S ได้ บริเวณช่องทางเดินกระแสจะต้องมีสภาพเป็นสารกึ่งตัวนำชนิดพีด้วย โครงสร้างของสัญลักษณ์ MOSFET แต่ละแบบแสดงได้ดังรูป 3.6



รูปที่ 3.6 แสดงโครงสร้างและสัญลักษณ์ของ MOSFET แบบต่าง ๆ

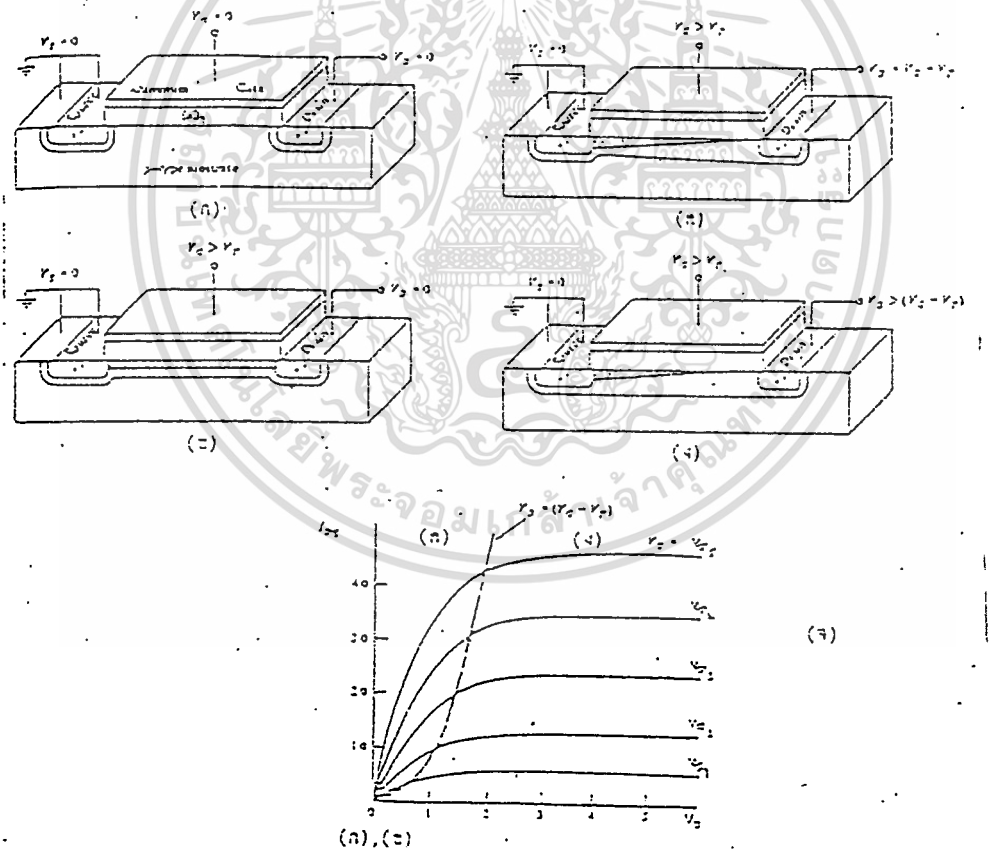
3.7 หลักการให้ไบอัส MOSFET

ในการที่จะทำให้มอสทรานซิสเตอร์ ทำงานอย่างมีประสิทธิภาพตามต้องการนั้น จำเป็นอย่างยิ่งที่จะต้องให้ไบอัสกับส่วนต่างๆของมันอย่างเหมาะสมโดยคำนึงถึงหน้าที่หลักของแต่ละส่วนดังกล่าวมาแล้วข้างต้นเช่นส่วนซอสซึ่งกำหนดให้เป็นแหล่งจ่ายประจุพาหะส่วนมากที่ทำให้เกิดกระแสแล้ว ประจุเหล่านี้จะเคลื่อนที่ผ่านช่องทางเดินกระแสไปยังส่วนเดรนออกไปสู่วงจรไฟฟ้าภายนอกทำให้เกิดเป็นกระแสเดรน-ซอส ดังนั้นในการให้แรงดันไบอัสที่เหมาะสมระหว่างส่วนเดรนกับส่วนซอส จึงหมายถึงการให้ศักดาไฟฟ้าแก่ส่วนซอสเมื่อเทียบกับส่วนเดรนแล้ว ส่วนซอสจะต้องเป็นแหล่งจ่ายประจุพาหะส่วนมากเสมอเช่นในกรณีของเอ็น-แชนแนลมอสทรานซิสเตอร์ซึ่งมีอิเลคตรอนหรือประจุลบ เป็นประจุพาหะส่วนมากที่ทำให้เกิดกระแสดังนั้นจะต้องให้ศักดาไฟฟ้าเป็นลบ ที่ส่วนซอสเมื่อเทียบกับส่วนเดรน เพื่อให้ส่วนซอสทำหน้าที่เป็นแหล่งจ่ายอิเลตตรอนนั้นเอง ดังแสดงในรูปที่ 3-7 หรือในกรณีของพี-แชนแนลมอสซิสเตอร์หรือมีโฮลหรือประจุบวกเป็นประจุพาหะส่วนมากที่ทำให้เกิดกระแส ก็จะให้ศักดาไฟฟ้าเป็นบวก ที่ส่วนซอสเมื่อเทียบกับส่วนเดรนสำหรับส่วนเกทจะต้องให้ศักดาไฟฟ้า (เมื่อเทียบกับฐานรอง) ในลักษณะที่สามารถควบคุมการเปิดหรือปิดช่องทางเดินกระแสได้ โดยใช้การพิจารณาหลักเดียวกับการให้ศักดาไฟฟ้าแก่ตัวเก็บประจุไฟฟ้า (Capator) ทั่ว ๆ ไป เช่น ตัวอย่างที่แสดงในรูปที่ 3-7 เป็น เอ็น-แชนแนล เอ็นฮานค์เมนต์โหมด มอสทรานซิสเตอร์ ซึ่งโดยปกติถ้าไม่มีศักดาไฟฟ้าให้ที่ส่วนเกท ช่องทางเดินกระแสจะเปิดอยู่ ทำให้ส่วนเดรนกับส่วนซอสแยกออกจากกันดังนั้นจะมีความต่างศักดาระหว่างส่วนเดรนกับส่วนซอสที่เกิดขึ้นกระแสก็ไม่สามารถไหลได้



รูปที่ 3-7 แสดงการให้ไบอัสแก่ Transister

ในกรณีนี้จะต้องให้ศักดาไฟฟ้าที่ส่วนเกทเป็นบวก เมื่อเทียบกับฐานรองเพื่อให้เกิดการสะสมประจุลบในช่องทางกระแสอันจะเป็นการควบคุมช่องทางเดินกระแสให้มีคุณสมบัติเป็นสารกึ่งตัวนำชนิดเอ็นมากหรือน้อยได้ตามต้องการนั้นก็คือการควบคุมการเปิด-ปิด ช่องทางเดินกระแสนั้นเองหรือในกรณีของเอ็น-แชนแนล ดีพลีทชั่นโหมด มอสทรานซิสเตอร์ ซึ่งโดยปกติขณะที่แรงดันไบอัสที่ส่วนเกท สารกึ่งตัวนำบริเวณช่องทางเดินกระแสจะมีสภาพเป็นสารกึ่งตัวนำชนิดเอ็นอยู่แล้ว ทำให้เกิดการเชื่อมต่อระหว่างส่วนเดรนกับส่วนซอส ดังนั้นเมื่อมีความต่างศักดาระหว่างส่วนเดรนกับส่วนซอสเกิดขึ้นจะทำให้กระแสไหลได้ทันที



รูปที่ 3.8

(ก) (ค) แสดงเอ็น-แชนแนล เอ็นฮานค์เมนต์โหมดภายใต้เงื่อนไขการไบอัสในช่วงต่างๆ

(จ) แสดงกราฟคุณสมบัติ ของเอ็น-แชนแนลเอ็นฮานค์เมนต์โหมด

ฉนั้นในการควบคุมปริมาณกระแส ก็ทำได้โดยการควบคุมความนำไฟฟ้าของช่องทางเดินกระแสนั้นโดยการให้แรงดันไบอัสส่วนเกทในลักษณะที่ทำให้เกิดการเปลี่ยนแปลงการสะสมประจุลบ หรืออิเลคตรอน เช่นถ้าทำให้ส่วนเกทนี้มีศักดาไฟฟ้าเป็นบวกเมื่อเทียบกับฐานรอง ทำให้เกิดการสะสมประจุลบที่ช่องทางเดินกระแสมากขึ้นเป็นผลให้ความนำไฟฟ้ามีค่ามากขึ้นกระแสก็ไหลได้มากขึ้น แต่ถ้าส่วนเกทมีศักดาไฟฟ้าเป็นลบ เมื่อเทียบกับฐานรองจะทำให้เกิดการสะสมประจุบวกที่ช่องทางเดินกระแสเป็นผลให้ความนำไฟฟ้าของสารกึ่งตัวนำชนิดเอ็นบริเวณช่องทางเดินกระแสลดลงดังนั้นกระแสจะไหลได้น้อยลงด้วยส่วนในกรณีของพีก็พิจารณาได้ในทำนองเดียวกัน

โดยหลักการที่กล่าวมาข้างต้นเป็นการเตรียมพร้อมที่จะให้มอสทรานซิสเตอร์ทำงานตามที่ต้องการ ซึ่งกลไกการเปลี่ยนแปลงคุณสมบัติทางไฟฟ้าภายในของมัน พออธิบายได้ดังนี้ตัวอย่างเช่นในกรณีของเอ็น-แชนแนล เอ็นฮานด์เมนต์โหมดมอสทรานซิสเตอร์เมื่อได้รับแรงดันไบอัสที่เหมาะสมจะมีการเปลี่ยนแปลงคุณสมบัติทางไฟฟ้าดังแสดงในรูปที่ 3.8 (ก),(ข) ซึ่งจะเห็นว่าในรูปที่3.8 (ก) จะแสดงสภาพปกติ ของมอสทรานซิสเตอร์ที่กำลังพิจารณา ในขณะนี้สารกึ่งตัวนำประเภทช่องทางเดินกระแสเป็นชนิดพี ทำให้ส่วนเดรนกับส่วนซอสแยกออกจากกัน ทางไฟฟ้าดังนั้นถึงแม้ว่าจะมีความต่างศักดาไฟฟ้าจะเกิดขึ้นระหว่างเดรนกับซอสประจุพาหะส่วนมากก็จะเคลื่อนที่จากซอสไปเดรนไม่ได้ นั้นคือกระแส เป็นศูนย์ในรูปที่3.8 (ข) เมื่อให้ศักดาไฟฟ้าที่เกทมีค่ามากกว่าค่าแรงแรงดันขีดเริ่มของมันแล้วจะเกิดการเหนี่ยวนำประจุลบขึ้น ที่ส่วนฐานรองบริเวณผิวสัมผัส กับออกไซด์ส่วนเกททำให้สารกึ่งตัวนำบริเวรนั้นมีคุณสมบัติเหมือนสารกึ่งตัวนำชนิดเอ็นเชื่อมต่อระหว่างส่วนเดรนกับส่วนซอสซึ่งเรียกส่วนนี้ว่าช่องทางเดินกระแสให้สังเกตว่าตรงรองต่อระหว่างสารกึ่งตัวนำชนิดเอ็นกับชนิดพี จะมีบริเวณปลอดประจุพาหุเสมอ(จากทฤษฎีของรอยต่อพี- เอ็น) ในรูปที่3.8 5 (ค) เมื่อให้ความต่างศักดาไฟฟ้าระหว่างส่วนเดรนกับซอสมีค่าเป็น แล้วแต่เนื่องจากส่วนซอสต่อเชื่อมสัมผัสบางไฟฟ้า อยู่กับฐานรองดังนั้นจึงเกิดการไบอัสย้อนกลับขึ้นระหว่างรอยต่อพี-เอ็นที่ส่วนฐานรองรอบๆส่วนเดรน เป็นผลให้เกิดการเปลี่ยนแปลงจำนวนประจุพาหะที่ช่องทางเดินกระแส บริเวณใกล้ๆกับส่วนเดรน แต่เนื่องจากแรงดันไบอัส พอดีเท่ากับแรงดัน ที่ทำให้เกิดทางเดินกระแส ดังนั้นช่องทางเดินกระแสจึงยังคงเชื่อต่ออยู่พอดีกับส่วนเดรนเสมือนเป็นความต้านทานตัวหนึ่ง ฉนั้นกระแสจึงเพิ่มขึ้นอย่างเป็นเชิงเส้นกับค่าแรงดัน ที่เพิ่มขึ้นจาก ในรูปที่3-8 (ง)เป็นการให้ศักดาไฟฟ้าที่ส่วนเดรนมีค่ามากกว่าแรงดันซึ่งจะทำให้ช่องทางเดินกระแสถูกแยกออกจากส่วนเดรนโดยสนามไฟฟ้าของรอยต่อพี-เอ็นที่เกิดจากไบอัสย้อนกลับด้วยแรงดัน ไบอัสนั่นเองในกรณีนี้ ความต่างศักดาไฟฟ้าระหว่างส่วนเดรนกับส่วนซอสส่วนใหญ่จะปรากฏอยู่ ที่รอยต่อพี-เอ็นดังนั้นถึงแม้จะเพิ่มแรงดันให้มากขึ้นอีกแต่ความต่างศักดาไฟฟ้าระหว่างปลายทั้งสองของ

สารกึ่งตัวนำ บริเวณช่องทางเดินกระแสก็ยังคงมีค่าประมาณเท่าเดิมดังนั้น กระแสจึงมีค่าประมาณคงที่กราฟความสัมพันธ์ระหว่างกระแส กับแรงดัน ในขั้นตอนต่างๆ แสดงได้ดังในรูปที่ 3.8 (จ)

บทที่ 4
การออกแบบโปรแกรมที่ใช้ในการควบคุม

4.1 การควบคุมโดยใช้ Program SGPIB.C

SGPIB.C ออกแบบมาเพื่อประยุกต์ใช้งานอุปกรณ์ GPIB การออกแบบ Program แบ่งออกเป็นสองส่วนคือ

1. ส่วนที่ใช้ควบคุมตัวอุปกรณ์ต่างๆเช่น Oscilloscope,Functiongen฿
2. ส่วนที่เป็น Program ประยุกต์ เช่น Diode Characteristic฿

4.2 การออกแบบ Program ส่วนที่ใช้ควบคุมอุปกรณ์ OSILLOSCOPE

Pgrammable Oscilloscope มี Memory ที่ใช้เก็บ Data ที่ได้จากการวัดค่าต่างๆอยู่ทั้งหมด 4096 Byte หรือ 4 Kbyte ใน 4Kbyte แบ่งออกเป็น 2 ส่วนคือ Channel 1และ Channel 2. ข้อมูลของ Channel 1 อยู่ที่ตำแหน่ง 0 - 2047 ส่วน Channel 2 ข้อมูลอยู่ที่ 2048 - 4096

หลังจากที่ระบบต่างได้ต่อไว้ถูกต้อง และ Driver ของ GPIB ถูก Install แล้วการต่อ Programmable Oscilloscope สามารถทำได้ 2 ทางคือ

4.3 ติดต่อผ่าน Program IBIC.EXE ที่ให้มาพร้อมเครื่อง ซึ่งมีขั้นตอนการติดต่อดังนี้

1.1 ibfind devc

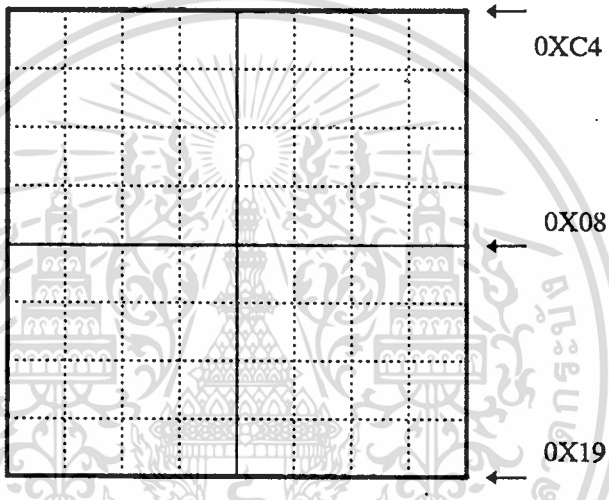
1.2 ibwrt "DIG 1 "

1.3 ibrd 2048

-ibfind เป็นการเลือกตัว อุปกรณ์ (Device) ที่เราต้องการติดต่อ. ในที่นี้ Device ที่เราจะติดต่อคือ devc(Ossillosclpe) ตามปกติเมื่อเลือกตัว อุปกรณ์ ที่ต้องการติดต่อแล้วต้องตามด้วย ibclr แต่สำหรับ Oscilloscope การใช้คำสั่ง ibclr จะทำให้ไม่สามารถติดต่อกับข้อมูลในหน่วยความจำของ Oscilloscope ได้.

-ibwrt เป็นการเขียนข้อมูลหรือส่งข้อมูลเข้าไปยังตัว อุปกรณ์ ในที่นี้เป็นการเลือก Channelที่เราต้องการติดต่อ. DIG 1 หมายความว่าเป็นการติดต่อกับ Memory ของ Channel 1 ส่วน DIG 2 เป็นการติดต่อกับ Memory ของ Channel 2 และ DIG 3 เป็นการติดต่อกับ Memory แบบ Dual

-ibrd เป็นการอ่านข้อมูลจาก ออสซิลโลสโคป ข้อมูลที่อ่านมา หนึ่งหน้าของ ออสซิลโลสโคป จะเป็นข้อมูลที่ได้จากการ สุ่ม (Sampling) ตามแนวนอนจำนวน 2048 จุด ตามรูปที่ 4-1



รูปที่ 4-1 แสดงตำแหน่ง แบบ ASSCII Code หน้าจอออสซิลโลสโคป

ข้อมูลที่ได้จาก ibrd จะอยู่ในรูปของ ASSCII Code เราต้องนำข้อมูลเหล่านั้นมาแปลงให้อยู่ในรูปของตัวเลขถึงจะได้ระดับของสัญญาณอย่างแท้จริง. เมื่อระบบถูกต่ออยู่ในสภาพสมบูรณ์ การตรวจสอบ(Check) ค่าสัญญาณที่อ่านได้จาก ออสซิลโลสโคป สามารถทำได้ตามขั้นตอนดังนี้

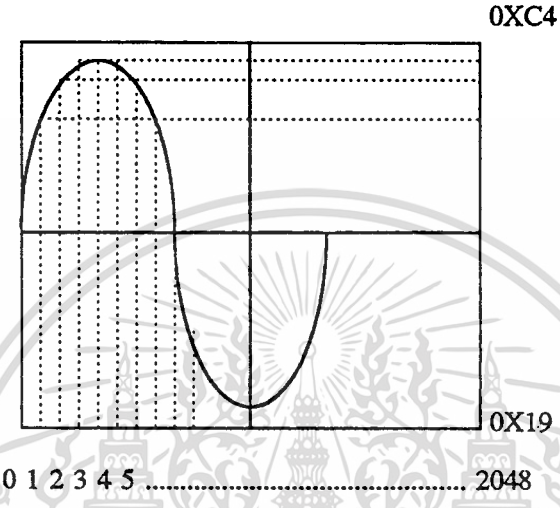
1 กดปุ่มของ Channel 1 ให้อยู่ที่ตำแหน่ง กราวด์(Gound) เลื่อนตำแหน่งของเส้น Scan ของ ออสซิลโลสโคป ให้อยู่กลางจอพอดี

2 Run Program IBIC.EXE เขียนคำสั่งดังนี้.

- ibfind devc
- ibwrt "DIG 1 "
- ibrd 2048

ค่าที่ได้จากการอ่านจะเป็นชุดของรหัส ASCII Code ทางด้านขวาสุด(รูปที่) เมื่อนำค่าดังกล่าวมาแปลงค่าจำนวนจริง จะได้ค่าเท่ากับ 128. คือตำแหน่งออสซิลโลสโคป ในตำแหน่งอื่นๆก็สามารถทำได้เช่นกัน.

ตัวอย่างการอ่านค่าสัญญาณ Sine wave ดังรูปที่ 4-2



รูปที่ 4-2 แสดงการอ่านค่าสัญญาณจาก ออสซิลโลสโคป

ลำดับที่	ค่าที่อ่านได้
0	128
1	130
2	135
3	140
4	144
5	140
.	.
.	.
.	.
.	.

ตารางที่ 4-1 แสดงลำดับและค่าสัญญาณที่อ่านได้

4.4 การติดต่อผ่านภาษา C สามารถทำได้ตาม Program ดังนี้

```
#include <stdio.h>
main()
    {
    int ud;
    unsigned char value[2048];
    ud=infind ("DEVC");
    ibwrt(ud,"DIG 1 ",5);
    ibrd(ud,value,2048);
    }
```

เมื่อทำ Project file และ Run Program จากภาษา C ข้อมูลจาก Channel 1 จะถูกอ่านเข้ามาเก็บไว้ในตัวแปล value ในรูปของ ASCII Code

ตัวแปล value จำเป็นต้องประกาศเป็นแบบ unsigned เพื่อเป็นการ make sure ข้อมูลที่อ่านมาจะต้องเป็นจำนวนเต็ม ที่เป็นบวกเท่านั้น. ถ้าไม่ใช้ตัวแปลแบบ unsigned จะเกิดปัญหาขึ้นได้ในขณะที่นำค่าที่อ่านได้มาใช้งาน.

4.5 ตัวอย่าง Program ที่ใช้งานจริง.

```
1   unsigned char value[4096];      /*-- Raw data from oscilloscope   --*/
                                    /*-- memory and make sure unsigned --*/
                                    /*-- value --*/
2   int scopex=70;
3   int scopey=70;
4   union REGS regs;
5   main()
6   {
7   int ud;
8   int gdrive=DETECT,gmode,error;      /*-- Set driver mode --*/
9   register int x,y,buttons;
```

```
10      char Cx[3],Cy[3];
11      initgraph(&gdrive,&gmode,"");
12      error=graphresult();
13      if(error !=grOk)
14      exit(1);
15      ud=ibfind("DEVC ");
16      regs.x.ax=0;
17      int86(0x33,&regs,&regs);                    /*-- Mouse initial --*/
18      cleardevice();
19      Screen(scopex,scopey);
20      Pannal();
21      DScreen();
22      do{
23      regs.x.ax=3;
24      int86(0x33,&regs,&regs);                    /*-- Mouse information --*/
25      buttons=regs.x.bx&3;
26      x=regs.x.cx;
27      y=regs.x.dx;
28        }
29      setfillstyle(1,7);
30      bar(505,22,560,35);
31      setcolor(4);
32      sprintf(Cx,"%3d",x);
33      sprintf(Cy,"%3d",y);
34      outtextxy(508,23,Cx);
35      outtextxy(538,23,Cy);
36      }
37        if(buttons==1){
38        regs.x.ax=2;
39      int86(0x33,&regs,&regs);      /*-- Mouse hident --*/
```

```
40      MenuSelect(x,y);
41      }
42      if(SubMenu[0][0]==0 && SubMenu[0][1]==0 &&
43      SubMenu[1][0]==0 && Ready==1){
44      ibwrt(ud,"DIG 3",5);                    /*-- Get data in memory 3 --*/
45      ibrd(ud,value,4096);                    /*-- Reading data 4096 byte --*/
46      putimage(scopex,scopey,p,COPY_PUT);
47      PlotWave(14,1);
48      PlotWave(13,2);
49      }
50      regs.x.ax=1;
51      int86(0x33,&regs,&regs);                /*-- Mouse show --*/
52      if(MenuOn==4) buttons=3;
53      }while(buttons!=3);
54      free((void *)p);
55      closegraph();
56      ibloc(ud);
57      return(0);
58      }
59      void PlotWave(int Color,int Channel)
60      {
61      int i,ch=0;
62      int ReadValue;
63      }
64      if(Channel==2)   ch=2048;
65      else  ch=0;
66      ReadValue=(int)(1.27*(double)(value[ch]-0x80));
67      /*-- over spec limitted --*/
68      if(ReadValue>  128)     ReadValue=128;
69      if(ReadValue< -128)     ReadValue=-128;
```

```
70      }
71      setcolor(Color);
72      moveto(scopex,scopey+128-ReadValue);;
73      for(i=2;i<318;i++){
74      ReadValue=(int)(1.27*(double)(value[ch+(int)(double)i*6.4]-0x80));
75       /*-- over spec limitted --*/
76      if(ReadValue>  128)     ReadValue=128;
77      if(ReadValue< -128)     ReadValue=-128;
78      /*-- Plot wave form --*/
79      lineto(scopex+i,scopey+128-ReadValue);
80      }
81      if(AutoRead==1)  ReadLevel(Color,Channel);
82      }
```

4.6 การทำงานของโปรแกรม.

ตัวแปล value เป็นตัวแปลแบบ Character มีขนาด 4Kbyte ใช้ Transfer ข้อมูลจาก ออสซิลโลสโคป เข้าสู่ PC.

บรรทัดที่ 15 ติดต่อกับ ออสซิลโลสโคป.

บรรทัดที่ 44 ติดต่อกับ Memory ของ ออสซิลโลสโคป ซึ่งเป็นการติดต่อแบบ Dual.

บรรทัดที่ 45 อ่านค่าจาก Memory เข้ามาเก็บในตัวแปล value จำนวน 4Kbyte.

บรรทัดที่ 46 วาดจอภาพ ออสซิลโลสโคป ด้วยคำสั่ง putimage ที่ต้องใช้ putimage แทนการวาดรูปจอภาพซ้ำ ก็เพื่อป้องกันการกระพริบของจอภาพ .

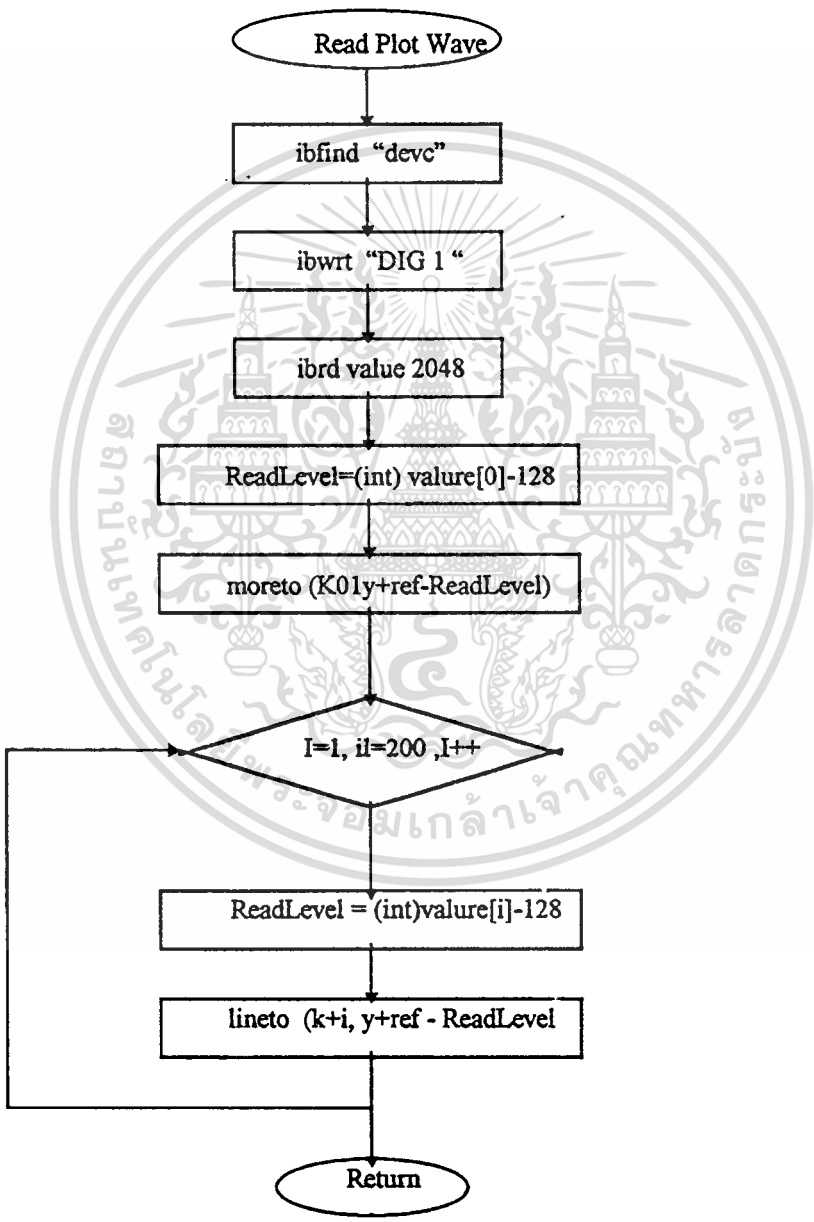
บรรทัดที่ 47 ,48 Plot Wave form Channel 1,Channel 2 ตามลำดับ.

Function PlotWave()

ค่าของตัวแปลที่ส่งเข้ามาใน function คือ Color ที่เป็นตัวแปลสำหรับกำหนดสีที่จะPlot wave form และ Channel เป็นตัวแปลที่บอก Channel ที่จะ Plot. ค่าที่อ่านมาได้จากออสซิลโลสโคป มีจำนวนทั้งหมด 2048 ค่า. แต่จอ (Mornitor) ของ คอมพิวเตอร์ ทางด้านแนวนอนมีค่าเพียง 640 Piccel. เพราะฉนั้นค่าที่จะ Plot จึงต้อง สุ่ม (Sampling) มา .จากโปรแกรมเป็นการPlot ทางแนวนอนเพียง 320 Piccel. ค่า จากการ สุ่มที่นำมาใช้ก็คือ 640 Piccel

ส่วนทางด้านแนวตั้งจะต้องคูณด้วย 1.27 เพราะอัตราส่วนทางด้านแนวตั้งกับแนวนอนของจอ Mornitor กล่าวคือระยะทาง 2 Piccel ทางด้านแนวตั้งกับ 2 Piccel ทางด้านแนวนอนไม่เท่ากันแต่จอภาพของ ออสซิลโลสโคป แนวตั้งกับแนวนอนเท่ากัน.

Flow Chart การ Plotwave form.



รูปที่ 4-3 โฟล์ชาร์ทที่ใช้ในการพล็อทรูปสัญญาณ

4.7 การทำงานของการ Plot ค่า

1. ReadValue=(int)(value[ch]-128); ค่าของตัวแปล value ตามปกติจะอยู่ในรูปของ ASCII Code ฉะนั้นจึงต้องเปลี่ยนค่า ให้เป็นตัวเลข .

ReadValue=(int)value[ch];

ค่า 128 เป็นค่าที่ Center ของ ออสซิลโลสโคป เราต้องการยึดจุดเนี้ให้เป็น จุดอ้างอิงดังนั้นค่าที่อ่านได้จากจึงต้องลบด้วย 128 เช่น ค่าที่อ่านมามีค่าเป็น 128 ค่า Read Value จะมีค่าเป็นศูนย์.

2. moveto(x,y+128-ReadValue); เริ่มจุดแรกของการ Plot wave form. +128 เป็นค่าครึ่งหนึ่งของจอภาพ ออสซิลโลสโคป บนจอ คอมพิวเตอร์

3. loop plot ค่า พล็อท ทั้งหมด 320 ครั้ง.

4. ReadValue = (int)(1.27*(double)(value[ch+(int)(double)i*6.4]-128);ค่าของตัวแปร value ถูกสุ่มมาทุกๆ 6.4 ค่า.

5. lineto(x+i,y+128-ReadValue);

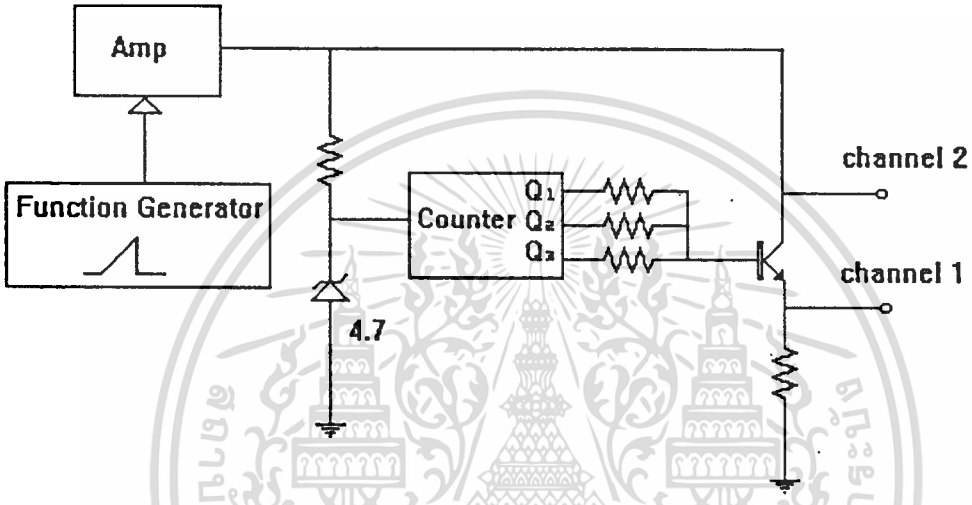
6. วนลูป.

บทที่ 5

การออกแบบวงจรวัดค่า Characteristic ของอุปกรณ์

5.1 การวัดค่า Characteristic ของ Transister

5.1.1 การวัดค่า Characteristic โดยใช้ระบบ GPIB

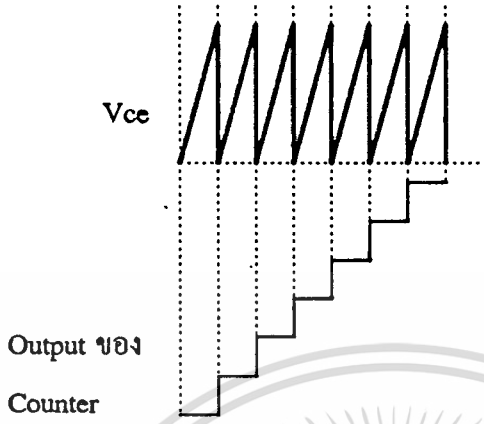


รูปที่ 5.1 วงจรที่ใช้ในการวัดหาค่า Charecteristic ของ Transister

5.1.2 การทำงานของวงจร

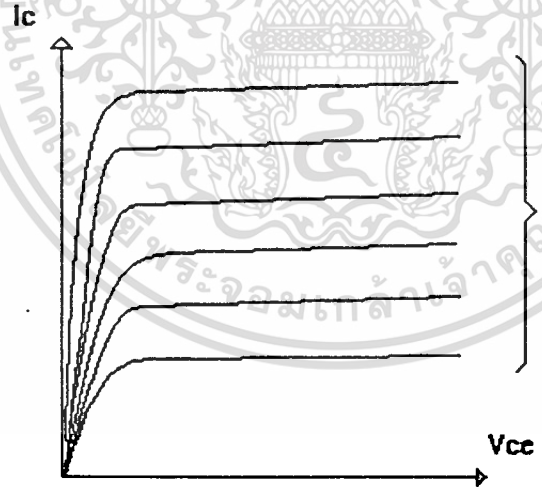
สัญญาณ Sawtooth จาก Function generator จะถูกขยายให้มีกระแสมากขึ้นโดยวงจร Amplifi ที่สร้างขึ้น เพื่อที่จะนำสัญญาณดังกล่าวไป Driver Transister ที่ต้องการวัดสัญญาณ ส่วนหนึ่งจะถูก Limit โดย Zener Diode ให้มีระดับแรงดันไม่เกิน 4.7 โวท์ล เพื่อนำไปเป็น Input ของวงจร Counter

เมื่อสัญญาณ Sawtooth ถูกป้อนเข้ามาโดย Function Generator และถูกขยายมาแล้ว สัญญาณจะตกคร่อมที่ขา CE ของ Transister ที่ต้องการทดสอบ โดยค่า โวลตเตจ ที่ป้อนเข้ามาจะเปลี่ยนแปลงระหว่าง 0 โวลต์ ถึง 10 โวลต์ ในขณะเดียวกัน Output ของ Counter ก็จะ Active ที่ Q_0 ("1") จะเกิดกระแสไหลเข้าขา เบส ของ Transister ค่าหนึ่ง เมื่อสัญญาณ Sawtooth เข้ามาอีกระดับ Ib ก็จะเปลี่ยนไป



รูปที่ 5.2 แสดงอินพุทที่ป้อนให้กับวงจร

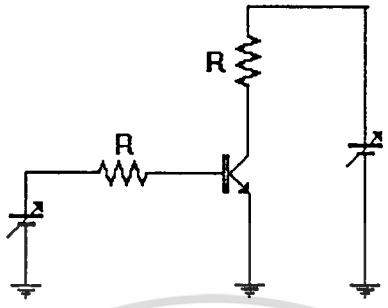
เลือก Scope ไว้ที่ Mode XY เพื่อวัดฐานเวลาออก ก็จะได้ Curve ของ Transister ดังรูป 5.3



Output ที่ได้จากการทดลอง

รูปที่ 5.3 แสดง Output ที่ได้จากการวัด ของ ทรานซิสเตอร์

5.1.3 การวัดค่า Characteristic ของ Transistor โดยไม่ใช้ระบบ GPIB



รูปที่ 5.4 วงจรที่ใช้วัดหาค่า Characteristic ของ Transistor

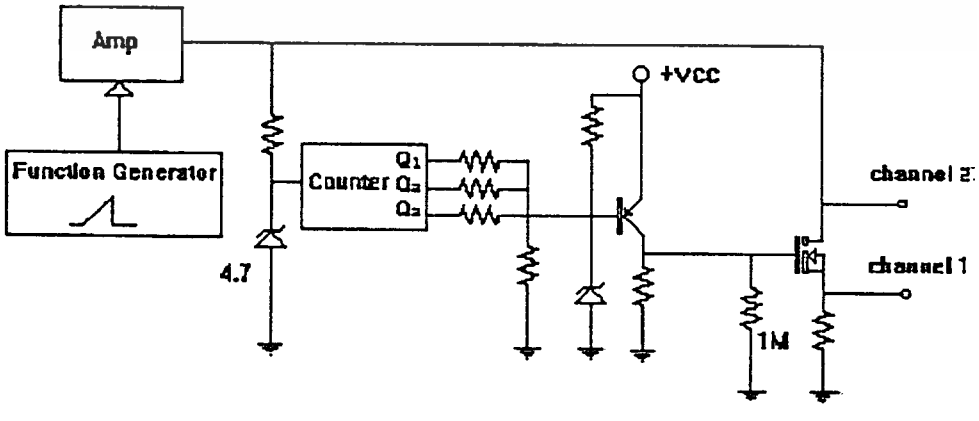
5.1.4 การทำงานของวงจร

1. Fix I_B ค่าหนึ่งแล้วปรับVccจาก 0-10 โดยปรับเป็น Step โดยวัดค่า Ic ของแต่ละ Step
2. เปลี่ยน I_B เพิ่มขึ้นแล้วปรับ Vce เป็น Ic เหมือนข้อ 1 บันทึกค่า Ic ของแต่ละ Step
3. นำค่า Vce และ Ic ที่ I_B ค่าต่างๆ มา Plot Graph

จะเห็นว่าการวัดด้วยระบบ GPIB จะให้ผลที่วัดเร็วมากกว่า การวัดแบบธรรมดามากๆและเราสามารถเห็นผลของกราฟทันที โดยไม่ต้องรอผล พล็อตกราฟ และยังสามารถเก็บข้อมูลต่างๆได้อย่างรวดเร็ว

5.2 การวัดค่า VI Curve ของ MOSFET

5.2.1 การวัดค่า VI Curve โดยใช้ระบบ GPIB

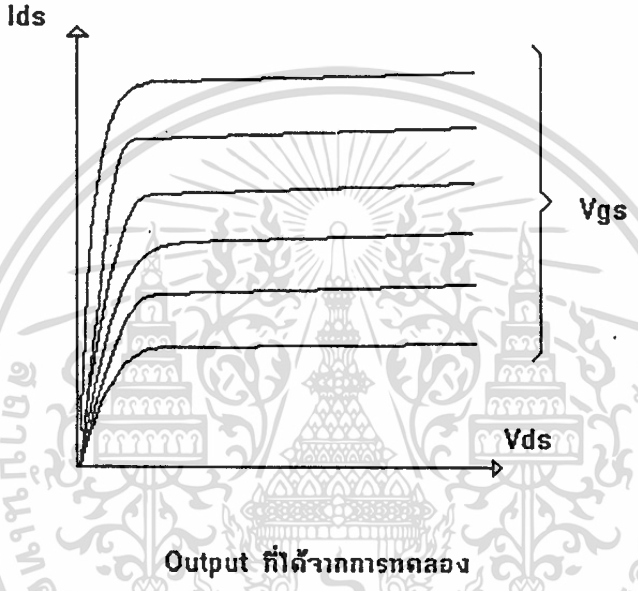


รูปที่ 5.5 วงจรวัดหาค่า VI Curve ของ MOSFET

5.2.2 การทำงานของวงจร

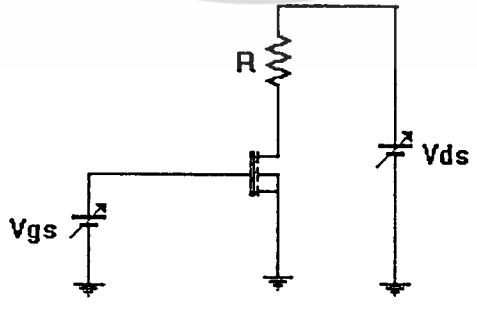
สัญญาณ Sawtooth จะถูกแบ่งออกเป็น 2 ทาง ทางแรกจะเข้าทาง Counter เพื่อสร้างระดับแรงดัน Vgs และ อีกส่วนหนึ่งคือ Vds ลักษณะการทำงานเหมือน Transister จะต่างกันตรงที่ Transister นั้นควบคุมด้วย Ic ด้วย Ib แต่ MOSFET จะ Bias ด้วย แรงดันผลการทำงานของวงจร

output ที่ได้จะแสดงดังรูป 5.6



รูปที่ 5.6 แสดง Output ที่ได้จากการวัด

5.2.3 การวัด VI โดยไม่ใช้ GPIB (FET Ids-Vds Charecteristic)

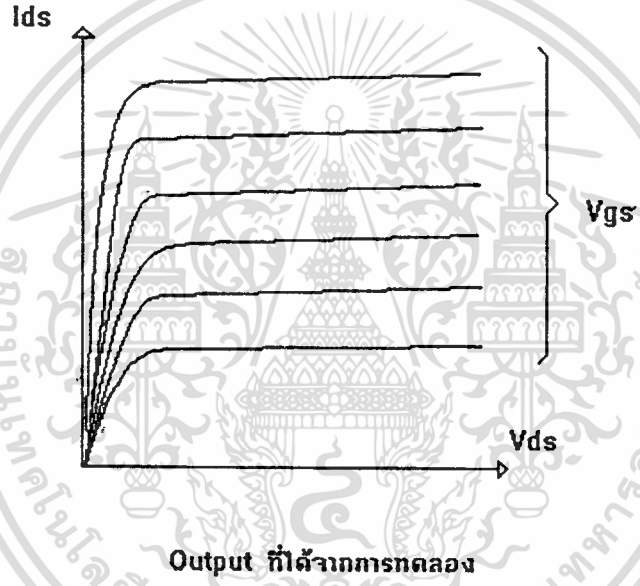


รูปที่ 5.7 วงจรของ มอสเฟส

5.2.4 การทำงานของวงจร

1. ปรับค่าแรงดัน Vgsเป็น0 และปรับค่าVdsเป็น0-15 ขึ้นทีละ Step บันทึกค่ากระแส Ids
2. ปรับค่าแรงดัน Vgs เป็น Vgs+0.5 และปรับ Vds เหมือนข้อ 1
3. ทำซ้ำข้อ 2 จนได้ Step ของ Vgs 7-8 Step
4. นำค่า Ids, Vds มาพล็อตกราฟ

จะเห็นว่าการวัดด้วยระบบ GPIB จะให้ผลที่วัดเร็วมากกว่า การวัดแบบธรรมดามากๆและเราสามารถเห็นผลของกราฟทันที โดยไม่ต้องรอผล พล็อตกราฟ และยังสามารถเก็บข้อมูลต่างๆได้อย่างรวดเร็ว

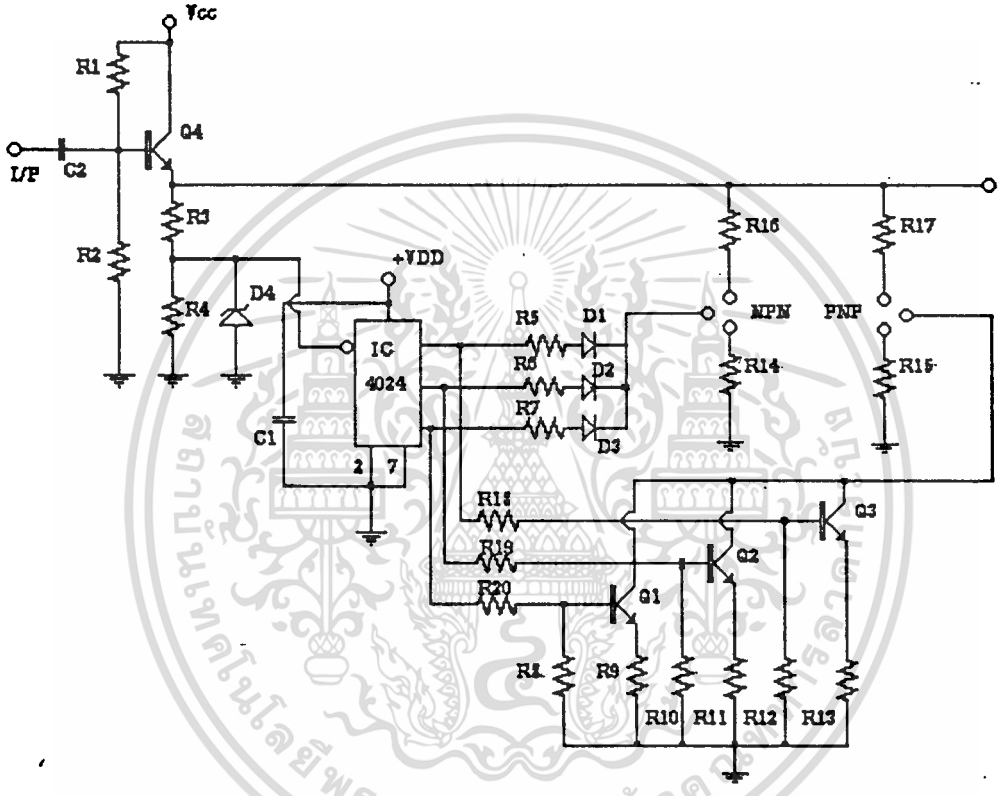


รูปที่ 5.8 แสดง Output ที่ได้จากการวัด ของ มอสเฟส

บทที่ 6

การใช้งานและผลการทดลอง

6.1 วงจรวัดค่า Characteristic ของ Transistor



รูปที่ 6.1 วงจรหาค่า Charecteristic ของ Transister

6.2 การทำงานของวงจร

สัญญาณ Sawtooth จาก Function generator จะถูกขยายให้มีกระแสมากขึ้นโดยวงจร Amplifi ที่สร้างขึ้น เพื่อที่จะนำสัญญาณดังกล่าวไป Driver Transister ที่ต้องการวัดสัญญาณ ส่วนหนึ่งจะถูก Limit โดย Zener Diode (D4) ให้มีระดับแรงดันไม่เกิน 4.7 โวท์ล เพื่อนำไปเป็น Input ของวงจร Counter (Ic1 4024)

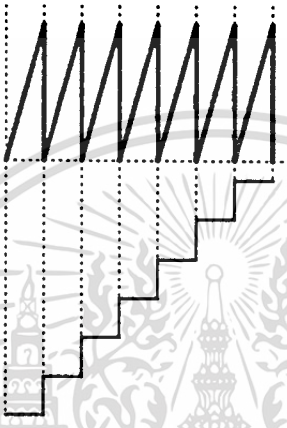
เมื่อสัญญาณ Sawtooth ถูกป้อนเข้ามาโดย Function Generator และถูกขยายโดย Q4 สัญญาณที่ขา E ของ Transister Q4 จะนำไปเป็นสัญญาณที่ตกคร่อม CE ของ Transister ที่ต้องการทดสอบ โดยค่า Voltage ที่ป้อนเข้ามาจะเปลี่ยนแปลงระหว่าง 0 โวลต์ ถึง 10 โวลต์ ใน

ขณะเดียวกัน Output ของ Counter(Ic1 4024) ก็จะ Active ที่ Q_0 ("1") จะเกิดกระแสไหลเข้าขา Base ของ Transister ที่ต้องการทดสอบ เมื่อสัญญาณ Sawtooth เข้ามาอีกระดับ Ib ก็จะเปลี่ยนไป

T1 ใช้วัด Transister วัดค่าแบบ NPN

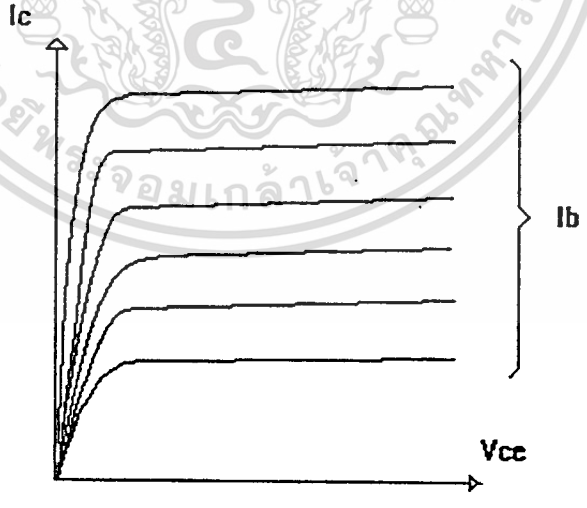
T2 ใช้วัด Transister วัดค่าแบบ PNP

Transister Q1,Q2,Q3 ทำหน้าที่สร้าง input ให้กับ Transister PNP



รูปที่ 6.2 แสดงสัญญาณ อินพุทและ เอาท์พุท

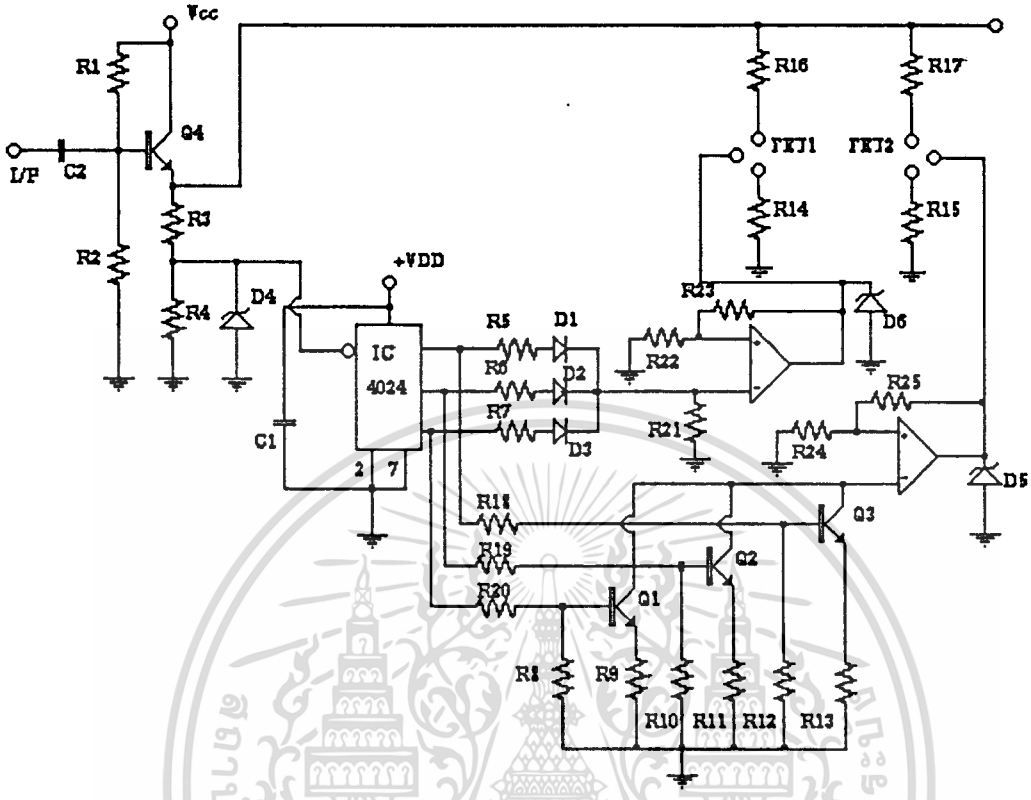
เลือก Scope ไว้ที่ Mode XY เพื่อวัดฐานเวลาออก ก็จะได้ Curve ของ Transister ดังรูป 6.3



Output ที่ได้จากการทดลอง

รูปที่ 6.3 กราฟเอาท์พุทที่ได้จากการทดลอง

6.3 วงจรวัดค่า VI Curve ของ FET



รูปที่ 6.4 วงจรหาค่า VI Curve ของ MOSFET

6.4 การทำงานของวงจร

สัญญาณ Sawtooth จาก Function generator จะถูกขยายให้มีกระแสมากขึ้นโดยวงจร Amplifi ที่สร้างขึ้น เพื่อที่จะนำสัญญาณดังกล่าวไป Driver Transister ที่ต้องการวัดสัญญาณ ส่วนหนึ่งจะถูก Limit โดย Zener Diode (D4) ให้มีระดับแรงดันไม่เกิน 4.7 โวท์ล เพื่อนำไปเป็น Input ของวงจร Counter (Ic1 4024)

เมื่อสัญญาณ Sawtooth ถูกป้อนเข้ามาโดย Function Generator และถูกขยายโดย Q1 สัญญาณที่ขา E ของ Transister Q1 จะนำไปเป็นสัญญาณที่ตกคร่อม DS ของ FET ที่ต้องการทดสอบ โดยค่า Voltage ที่ป้อนเข้ามาจะเปลี่ยนแปลงระหว่าง 0 โวลต์ ถึง 10 โวลต์ ในขณะเดียวกัน Output ของ Counter(Ic1 4024) ก็จะ Active ที่ Q_0 ("1") จะเกิด Voltage ตกคร่อม R17 และถูกขยายโดย Ic2 เพื่อนำไปเป็น Input ของ FET ของ Transister ที่ต้องการทดสอบ เมื่อสัญญาณ Sawtooth เข้ามาอีกระดับ Vgs ก็จะเปลี่ยนไป

T1 ใช้วัด FET แบบ N Channel

T2 ใช้วัด FET แบบ P Channel

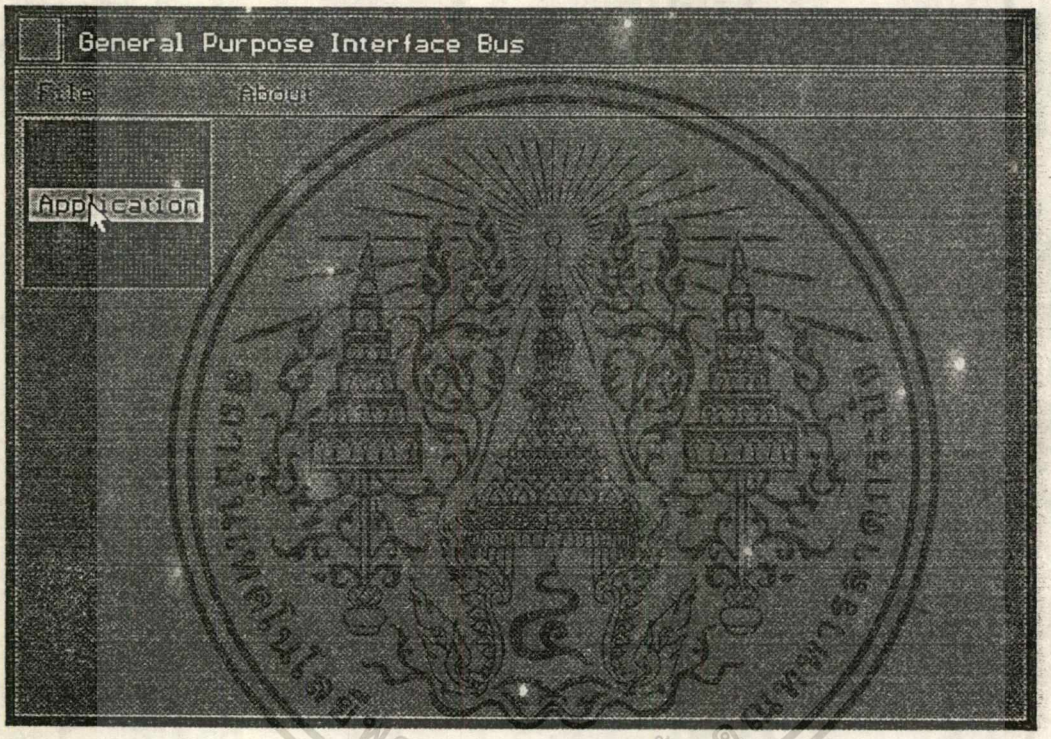
Transister Q2,Q3,Q4 ทำหน้าที่สร้าง input ให้กับ FET แบบ P Channel

6.5 การใช้งานโปรแกรม

การหาค่า VI Curve ของ Device

MenuFile

1. เลือก File จาก Main Menu ดังรูปที่ 6.5



รูปที่ 6.5

2. เลือก Application Menu ดังรูปที่ 6.5 จะปรากฏภาพจอ Oscilloscope ซ้อนกัน 2 จอ ดังรูปที่ 6.6 ด้านบนของจอ Oscilloscop จะมีข้อความบอกหน้าที่ของจอ ถัดลงมาคือปุ่มเลือกการทำงานของจอภาพ

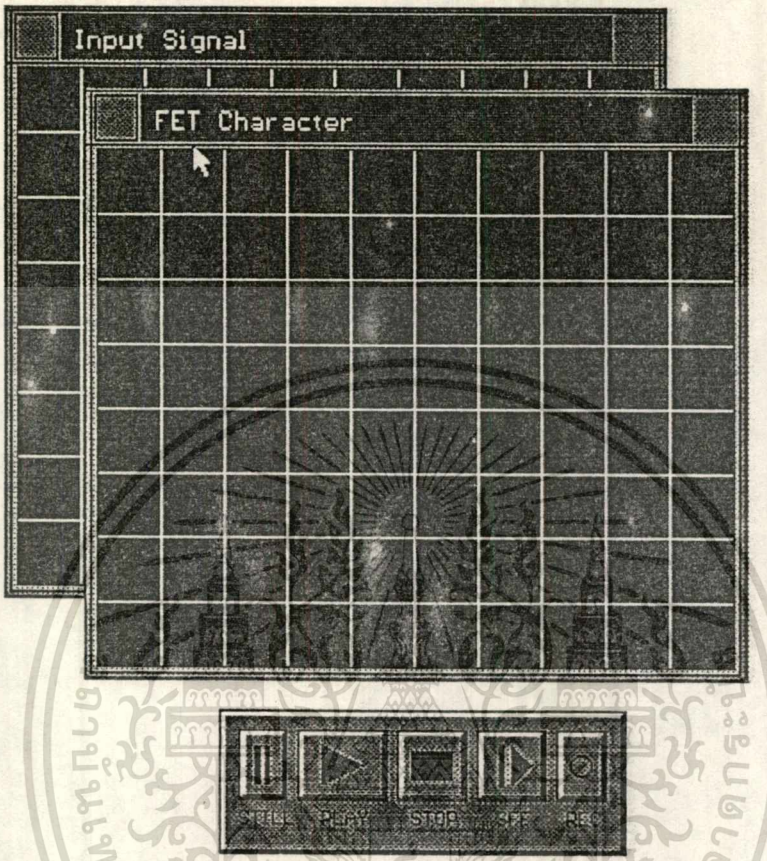
-Play ใช้แสดงสัญญาณแบบ Real time

-Pause หยุดการแสดงภาพชั่วคราว

- Stop ออกจาก Application

- SFF แสดงข้อมูลแบบช้า

- REC ใช้บันทึกสัญญาณที่ต้องการ



รูปที่ 6.6

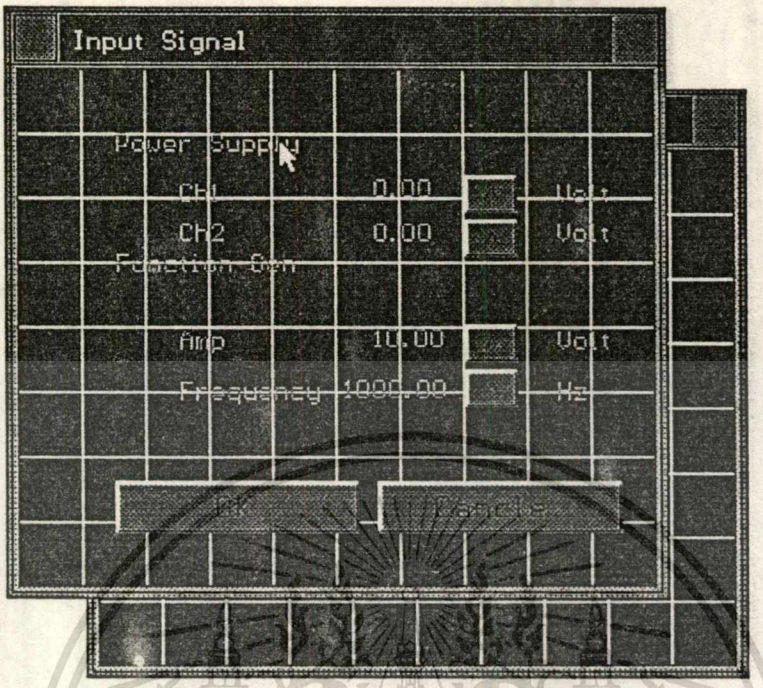
3. Input Signal เป็นส่วนการแสดงผลของสัญญาณที่จุดต่างๆ และยังสามารถ Set ค่าต่างๆ ของ Function Gen และ Power Supply ได้ โดยใช้ Mouse คลิ๊กที่ แถบสี ของ Lable Input signal การ Set ค่าต่างๆทำได้ดังนี้ ดังรูปที่ 6.7

3.1 ใช้ Mouse คลิ๊กที่กรอบสี่เหลี่ยมด้านซ้ายจอภาพของ Input Signal

3.2 จะปรากฏ Menu ให้ Set ค่าต่างๆตามต้องการ เมื่อ Set ค่าได้แล้วกดปุ่ม OK เมื่อต้องการค่าที่ Set หรือกดปุ่ม Cancel เมื่อไม่ต้องการ

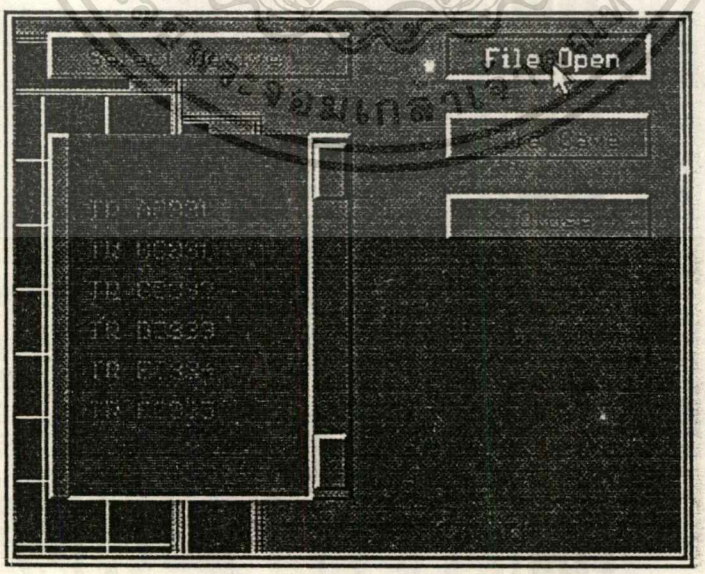
4.เมื่อ Set ค่าต่างๆได้ตามต้องการแล้ว ก็สามารถทำการวัดค่าต่างๆที่ต้องการได้

5.การ Save ค่าสัญญาณที่ต้องการ โดยการเลือกปุ่ม Rec และใส่ Number ของ Device ที่ทำการบันทึก



รูปที่6.7

6.การ Load ข้อมูลเก่ากลับมาดู เลือก Menu File เลือก File Open แล้วเลือก Number ของ Device ที่ต้องการดูค่า ดังรูปที่ 6.8



รูปที่ 6.8

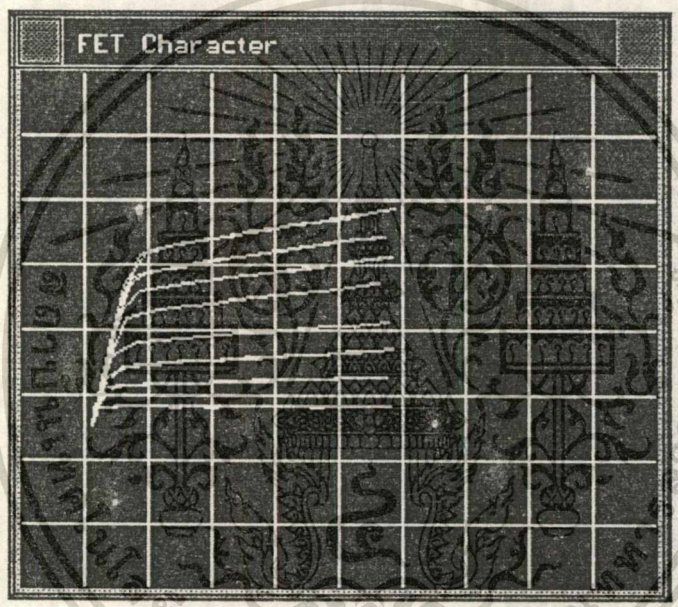
Menu About

Menu About เป็น Menu ที่ให้ข้อมูลเกี่ยวกับตัวโปรแกรมดังต่อไปนี้

1. Abstrac เป็นบทคัดย่อของ Project ที่ทำ

2. How to use ประกอบด้วย GPIB History ของระบบ GPIB และความสามารถของโปรแกรม

ตัวอย่างผลของการวัดทรานซิสเตอร์ เบอร์ 2SC458



รูปที่ 6.9 Charecteristic Transister 2SC 458

```
#include <stdio.h>
#include <graphics.h>
#include <alloc.h>
#include <dos.h>
#include <stdlib.h>

void Block(int Xtop,int Ytop,int Xdown,int Ydown,int Cwid,
int iCleft,int iCcenter,int iCright);
void Screen(int sx,int sy);    /*-- Exist memory --*/
void PlotWave(int Channel,int Color,int sx,int sy);
void XYPlot(int sx,int sy,int Color);
void BackWin(int Tsx,int Tsy,int Lsx,int Lsy,int ColorL,int ColorB,
char *Message);
void Display(void);
void WinMain(void);
void MainMenu(void);
void DisplayMainMenu(void);
void Position(int x,int y);
int  DisplaySubMenu(void);
int  ApplicationMenu(void);
void InstrumentMenu(void);
void AbstracMenu(void);
void HowtouseMenu(void);
int  WaitLoop(int Lx,int Rx,int Ty,int Ly);
void Dis_Play(int x,int y);
void Playing(int i,int j);
int  HoldStatus(void);
void MouseShow(void);
void MouseHide(void);
void MovetoArea(int x,int y);
```

```
void ButtonREC(int x,int y,int Cleft,int Ccenter,int Cright);
void ButtonsSFF(int x,int y,int Cleft,int Ccenter,int Cright);
void ButtonSREW(int x,int y,int Cleft,int Ccenter,int Cright);
void ButtonPlay(int x,int y,int Cleft,int Ccenter,int Cright);
void ButtonPause(int x,int y,int Cleft,int Ccenter,int Cright);
void ButtonSTOP(int x,int y,int Cleft,int Ccenter,int Cright);
void ButtonSTOP(int x,int y,int Cleft,int Ccenter,int Cright);
int  WorkingArea(int x,int y,char *Message);
void SignalSet(void);
void UpA(int x,int y);
void DnA(int x,int y);
void FileWork(void);
void OPenFile(void);


#define TOTALMENU 14
struct Heading{
char *Choice;
};

struct MenuStruct{
int iFrame[4];
int iRow[6];
int iCol;
struct Heading cItem[6];
int iLast;
int iNoNum;
};
```

```
struct MenuStruct MenuS[TOTALMENU];
struct MenuStruct Instrument;

union REGS      regs;
unsigned        size;
char far        *Video;
char far        *Tool;
unsigned char   cWave[4096];
int             MenuNum;
int             MenuItem;

float           Bch1=0;
float           Bch2=0;
float           GAmp=10.00;
float           Gfrq=1000.00;
int             Delay=0;


int main(void)
{
int gdrive=DETECT,gmode;
register int x,y,buttons;
int Mdefaul;
FILE *fp;

initgraph(&gdrive,&gmode,"");
MainMenu();
WinMain();
DisplayMainMenu();
```

```
regs.x.ax=0;
int86(0x33,&regs,&regs);                    /*-- Mouse initial --*/
regs.x.ax=4;
regs.x.cx=45;
regs.x.dx=50;
int86(0x33,&regs,&regs);                    /*-- Mouse show   --*/


regs.x.ax=1;
int86(0x33,&regs,&regs);                    /*-- Mouse show   --*/

do{
regs.x.ax=3;
int86(0x33,&regs,&regs);                    /*-- Mouse information --*/
buttons=regs.x.bx&3;
x=regs.x.cx;
y=regs.x.dx;
/*Position(x,y);*/
/*-- Menu 0 Active Area --*/
if(x>=14 && x<=51 && y>=42 && y<=60){
MenuNum=0;  MenuItem=0;
Mdefaul=WaitLoop(14,51,42,60);
if(Mdefaul==1)
DisplaySubMenu();
}
/*-- Menu 1 Active Area --*/
if(x>=113 && x<=160 && y>=42 && y<=60){
MenuNum=1;  MenuItem=0;
Mdefaul=WaitLoop(113,160,42,60);
if(Mdefaul==1)
```

```
DisplaySubMenu();
}
/*--- Buttons Active ---*/
if(x>=601 && x<=619 && y>=15 && y<=34
&& buttons==1){
buttons=HoldStatus();
}

}while(buttons!=3);
closegraph();
return(0);
}

void PlotWave(int Channel,int Color,int sx,int sy)
{
int i,y,x;
int Ref=256;
x=sx;  y=sy;
if(Channel==1){
setcolor(Color);
(int)cWave[(int)2*6.4];
moveto(x,sy+Ref-y);
for(i=2;i<=318;i++){
(int)cWave[(int)i*6.4];
lineto(x+i,sy+Ref-y);
delay(Delay);
}
}else if(Channel==2){
setcolor(Color);
(int)cWave[2048+(int)2*6.4];
```

```
moveto(x,sy+Ref-y);
for(i=2;i<=318;i++){
(int)cWave[2048+(int)i*6.4];
lineto(x+i,sy+Ref-y);
delay(Delay);
}
}
}
void XYPlot(int sx,int sy,int Color)
{
int x,y,i;
int OldY;
int Ref=256;
setcolor(Color);
(int)cWave[(int)2*6.4];
(int)cWave[2048+(int)2*6.4];
OldY=y;
moveto(sx+x,sy+Ref-y);
for(i=2;i<=218;i++){
delay(Delay);
(int)cWave[(int)i*6.4];
(int)cWave[2048+(int)i*6.4];
if(y<OldY){
moveto(sx+x,sy+Ref-y);
OldY=y;
}else{
lineto(sx+x,sy+Ref-y);
OldY=y;
}
```

```
}
}
void BackWin(int Tsx,int Tsy,int Lsx,int Lsy,int ColorL,int ColorB,char *Message)
{
regs.x.ax=2;
int86(0x33,&regs,&regs);
settextstyle(2,0,5);
setcolor(ColorB);
setfillstyle(1,ColorB);
bar(Tsx-6,Tsy-30,Lsx+6,Lsy+6);
setcolor(7);
setfillstyle(1,7);
bar(Tsx-1,Tsy-26,Tsx+20,Tsy-4);        /*-- Left buuton --*/
setcolor(ColorL);
setfillstyle(1,ColorL);
bar(Tsx+22,Tsy-26,Lsx-21,Tsy-4);       /*-- Label Message --*/
setcolor(15);
outtextxy(Tsx+30,Tsy-20,Message);
setcolor(0);
rectangle(Tsx-6,Tsy-30,Lsx+6,Lsy+6);
rectangle(Tsx-6,Tsy-28,Lsx+6,Tsy-2);   /*-- Envelop --*/
rectangle(Tsx-3,Tsy-28,Lsx+3,Lsy+3);
rectangle(Tsx-1,Tsy-26,Tsx+20,Tsy-4);
regs.x.ax=1;
int86(0x33,&regs,&regs);
}

void Screen(int sx,int sy)
{
```

```
int x,y,i;

MouseHide();
x=sx,y=sy; setcolor(9);
setfillstyle(1,9);
bar(x,y,x+320,y+256);
setcolor(15);
for(i=1;i<=9;i++){
line(x,y,x+320,y);

}
y=sy;
for(i=1;i<=11;i++){
line(x,y,x,y+256);

}
setcolor(0);
rectangle(sx,sy,sx+320,sy+256);
MouseShow();
}
void Display(void)
{
FILE *fp;
int x=50,y=100;
BackWin(x,y,x+320,y+256,0,7,"Main Signal");
Screen(x,y);
fp=fopen("SARN","r");
fgets(cWave,4096,fp);
fclose(fp);
```

```
PlotWave(1,1,x,y);
PlotWave(2,1,x,y);    /*--- Sawtooth ---*/

 y+=50;
BackWin(x,y,x+320,y+256,0,7,"Character Ristic");
putimage(x,y,Video,COPY_PUT);
/*XYPlot(x,y,0);*/
}
void WinMain(void)
{

BackWin(10,40,620,450,1,8,"General Purpose Interface Bus");
setcolor(7);
setfillstyle(1,7);
bar(8,39,622,452);
setcolor(0);
line(8,40,622,40);
line(8,62,622,62);
settextstyle(2,0,5);

}

void MainMenu(void)
{
MenuS[0].iRow[0]=45;
MenuS[0].iRow[1]=75;
MenuS[0].iRow[2]=100;
MenuS[0].iRow[3]=125;

MenuS[0].iFrame[0]=12;
```

```
MenuS[0].iFrame[1]=65;
MenuS[0].iFrame[2]=104;
MenuS[0].iFrame[3]=148;


MenuS[0].iCol=20;
MenuS[0].iNoNum=4;

MenuS[0].cItem[0].Choice="File       ";
MenuS[0].cItem[1].Choice="Instrument ";
MenuS[0].cItem[2].Choice="Application";
MenuS[0].cItem[3].Choice="Close      ";

MenuS[1].iRow[0]=45;
MenuS[1].iRow[1]=75;
MenuS[1].iRow[2]=100;
MenuS[1].iRow[3]=125;

MenuS[1].iFrame[0]=112;
MenuS[1].iFrame[1]=65;
MenuS[1].iFrame[2]=204;
MenuS[1].iFrame[3]=148;

MenuS[1].iCol=120;
MenuS[1].iNoNum=4;


MenuS[1].cItem[0].Choice="About      ";
MenuS[1].cItem[1].Choice="Abstract   ";
```

```
MenuS[1].cItem[2].Choice="How to use ";
MenuS[1].cItem[3].Choice="Close       ";


MenuS[2].iRow[0]=100;
MenuS[2].iFrame[0]=67;
MenuS[2].iFrame[1]=90;
MenuS[2].iFrame[2]=210;
MenuS[2].iFrame[3]=260;

MenuS[2].iCol   =75;
MenuS[2].cItem[0].Choice="Multimeter   ";

MenuS[3].iRow[0]=125;
MenuS[3].iCol   =75;
MenuS[3].cItem[0].Choice="Power Supply ";

MenuS[4].iRow[0]=150;
MenuS[4].iCol   =75;
MenuS[4].cItem[0].Choice="Functiongen  ";

MenuS[5].iRow[0]=175;
MenuS[5].iCol   =75;
MenuS[5].cItem[0].Choice="Oscilloscope ";

MenuS[6].iRow[0]=200;
MenuS[6].iCol   =75;
MenuS[6].cItem[0].Choice="Enter        ";

MenuS[7].iRow[0]=225;
```

```
MenuS[2].iCol   =75;
MenuS[7].iCol   =75;
MenuS[7].cItem[0].Choice="Close         ";

MenuS[8].iRow[0]=100;
MenuS[8].iFrame[0]=67;
MenuS[8].iFrame[1]=90;
MenuS[8].iFrame[2]=210;
MenuS[8].iFrame[3]=240;
MenuS[8].iCol   =75;
MenuS[8].cItem[0].Choice="FET Threshold    ";

MenuS[9].iRow[0]=125;
MenuS[9].iRow[1]=45;
MenuS[9].iCol   =75;
MenuS[9].cItem[0].Choice= "FET Character    ";

MenuS[10].iRow[0]=150;
MenuS[10].iCol   =75;
MenuS[10].cItem[0].Choice="Transistor Char ";

MenuS[11].iRow[0]=175;
MenuS[11].iCol   =75;
MenuS[11].cItem[0].Choice="Scope Application";

MenuS[12].iRow[0]=200;
MenuS[12].iCol   =75;
MenuS[12].cItem[0].Choice="Close Application";
```

```
}

void DisplayMainMenu(void)
{
int i;

setcolor(0);
settextstyle(2,0,5);
for(i=0;i<=1;i++)
outtextxy(MenuS[i].iCol,MenuS[i].iRow[0],
MenuS[i].cItem[0].Choice);

}

int DisplaySubMenu(void)
{
register int   x,y;
char  far *Sub;
int  i;
int  Mdefaul   =0;
int  EndSelect =0;
int  Rmain     =0;
FILE *fp;


setcolor(15);
settextstyle(2,0,5);
regs.x.ax=2;
int86(0x33,&regs,&regs);                    /*-- Mouse Hident   --*/
```

```
size=imagesize(MenuS[MenuNum].iFrame[0],MenuS[MenuNum].iFrame[1],
MenuS[MenuNum].iFrame[2],MenuS[MenuNum].iFrame[3]);
Sub=(char *)malloc(size);
getimage(MenuS[MenuNum].iFrame[0],MenuS[MenuNum].iFrame[1],
MenuS[MenuNum].iFrame[2],MenuS[MenuNum].iFrame[3],Sub);

setcolor(2);setfillstyle(1,2);
bar(MenuS[MenuNum].iFrame[0],MenuS[MenuNum].iFrame[1],
MenuS[MenuNum].iFrame[2],MenuS[MenuNum].iFrame[3]);
setcolor(0);
rectangle(MenuS[MenuNum].iFrame[0]+1,MenuS[MenuNum].iFrame[1]+1,
MenuS[MenuNum].iFrame[2]-1,MenuS[MenuNum].iFrame[3]-1);
for(i=1;i<=MenuS[MenuNum].iNoNum-1;i++)
outtextxy(MenuS[MenuNum].iCol,MenuS[MenuNum].iRow[i],
MenuS[MenuNum].cItem[i].Choice);
regs.x.ax=1;
int86(0x33,&regs,&regs);                    /*-- Mouse show   --*/
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);                    /*-- Mouse information --*/
x=regs.x.cx;
y=regs.x.dx;

/*--- Menu 0 Active ---*/
if(MenuNum==0){
if(x>=15 && x<=100 && y>=73 && y<=90){
MenuItem=1;
Mdefaul=WaitLoop(15,100,73,90);
if(Mdefaul==1){
```

```
MenuNum=2;
InstrumentMenu();
}
}
if(x>=15 && x<=100 && y>=99 && y<=115){
MenuItem=2;
Mdefaul=WaitLoop(15,100,99,115);
if(Mdefaul==1)
Rmain=ApplicationMenu();
if(Rmain==1){
EndSelect=1;
MenuItem=0; MenuNum=0;
}

}

if(x>=15 && x<=100 && y>=121 && y<=138){
MenuItem=3;
Mdefaul=WaitLoop(15,100,121,138);
if(Mdefaul==1){
EndSelect=1;
MenuItem=0; MenuNum=0;
}
}
}
/*--- Menu 1 Active ---*/
if(MenuNum==1){
if(x>=115 && x<=201 && y>=73 && y<=90){
MenuItem=1;
Mdefaul=WaitLoop(115,201,73,90);
```

```
if(Mdefaul==1)
AbstracMenu();

}
if(x>=115 && x<=201 && y>=99 && y<=115){
MenuItem=2;
Mdefaul=WaitLoop(115,201,99,115);
if(Mdefaul==1)
HowtouseMenu();
}
if(x>=115 && x<=201 && y>=123 && y<=137){
MenuItem=3;
Mdefaul=WaitLoop(115,201,123,137);
if(Mdefaul==1)
EndSelect=1;
}
}

}while(EndSelect==0);
MouseHide();
putimage(MenuS[MenuNum].iFrame[0],MenuS[MenuNum].iFrame[1],
Sub,COPY_PUT);
free((void *)Sub);
MouseShow();

MenuItem=0; MenuNum=0;
return(Rmain);

}
```

```
void Position(int x,int y)
{

char Cx[3],Cy[3],M[20];

setfillstyle(1,7);
bar(400,22,560,35);
setcolor(14);
sprintf(Cx,"%3d",x);
sprintf(Cy,"%3d",y);
sprintf(M,"%7lu",coreleft());
outtextxy(508,23,Cx);
outtextxy(538,23,Cy);
outtextxy(420,23,M);

}

void InstrumentMenu(void)
{
register int   x,y;
int   i;
char far  *cInstru;
int   Mdefaul   =0;
int   EndSelect =0;

MenuItem=0;
regs.x.ax=2;
int86(0x33,&regs,&regs);                    /*-- Mouse Hident --*/

size=imagesize(MenuS[2].iFrame[0],MenuS[2].iFrame[1],
```

```
MenuS[2].iFrame[2],MenuS[2].iFrame[3]);
cInstru=(char *)malloc(size);
getimage(MenuS[2].iFrame[0],MenuS[2].iFrame[1],
MenuS[2].iFrame[2],MenuS[2].iFrame[3],cInstru);


setcolor(5);setfillstyle(1,5);
bar(MenuS[2].iFrame[0],MenuS[2].iFrame[1],
MenuS[2].iFrame[2],MenuS[2].iFrame[3]);
setcolor(0);
rectangle(MenuS[2].iFrame[0]+1,MenuS[2].iFrame[1]+1,
MenuS[2].iFrame[2]-1,MenuS[2].iFrame[3]-1);

for(i=2;i<=7;i++)
outtextxy(MenuS[2].iCol,MenuS[i].iRow[0],
MenuS[i].cItem[0].Choice);

regs.x.ax=1;
int86(0x33,&regs,&regs);                /*-- Mouse Show --*/

do{
regs.x.ax=3;
int86(0x33,&regs,&regs);                /*-- Mouse information --*/
x=regs.x.cx;
y=regs.x.dx;
/*sition(x,y);*/
/*-- Instrument Menu 0 Active --*/
/*-- Multimeter Active --*/
if(x>=71 && x<=178 && y>=98 && y<=115){
MenuNum=2; MenuItem=0;
```

```
Mdefaul=WaitLoop(71,178,98,115);

}
/*-- Power Supply Active --*/
if(x>=71 && x<=178 && y>=123 && y<=140){
MenuNum=3; MenuItem=0;
Mdefaul=WaitLoop(71,178,123,140);
}

/*--- Function Generator Active --*/
if(x>=71 && x<=178 && y>=146 && y<=166){
MenuNum=4; MenuItem=0;
Mdefaul=WaitLoop(71,178,148,166);
}

/*--- Oscilloscope Active ---*/
if(x>=71 && x<=178 && y>=172 && y<=190){
MenuNum=5; MenuItem=0;
Mdefaul=WaitLoop(71,178,172,190);
}
/*--- Enter Active --*/
if(x>=71 && x<=178 && y>=197 && y<=214){
MenuNum=6; MenuItem=0;
Mdefaul=WaitLoop(71,178,197,214);
}
/*--- Close Active --*/
if(x>=71 && x<=178 && y>=222 && y<=239){
MenuNum=7; MenuItem=0;
Mdefaul=WaitLoop(71,178,222,239);
if(Mdefaul==1){
```

```
EndSelect=1; MenuNum=0;
}
}
}while(EndSelect==0);
regs.x.ax=2;
int86(0x33,&regs,&regs);                    /*-- Mouse Hident --*/
putimage(MenuS[2].iFrame[0],MenuS[2].iFrame[1],cInstru,COPY_PUT);
regs.x.ax=1;
int86(0x33,&regs,&regs);                    /*-- Mouse Show --*/

free((void *)cInstru);
}

int ApplicationMenu(void)
{
register int   x,y;
int   i;
char far  *cInstru;
int   Mdefaul  =0;
int   EndSelect =0;
int   RMain    =0;

MenuItem=0;
MouseHide();
size=imagesize(MenuS[8].iFrame[0],MenuS[8].iFrame[1],
MenuS[8].iFrame[2],MenuS[8].iFrame[3]);
cInstru=(char *)malloc(size);
getimage(MenuS[8].iFrame[0],MenuS[8].iFrame[1],
MenuS[8].iFrame[2],MenuS[8].iFrame[3],cInstru);
```

```
setcolor(3);setfillstyle(1,3);
bar(MenuS[8].iFrame[0],MenuS[8].iFrame[1],
MenuS[8].iFrame[2],MenuS[8].iFrame[3]);
setcolor(0);
rectangle(MenuS[8].iFrame[0]+1,MenuS[8].iFrame[1]+1,
MenuS[8].iFrame[2]-1,MenuS[8].iFrame[3]-1);
settextstyle(2,0,5);

for(i=8;i<=12;i++)
outtextxy(MenuS[8].iCol,MenuS[i].iRow[0],
MenuS[i].cItem[0].Choice);
MouseShow();
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);              /*-- Mouse information --*/
x=regs.x.cx;
y=regs.x.dx;
/*-- Application Menu Active --*/
/*-- FET Theshold --*/
if(x>=71 && x<=200 && y>=98 && y<=115){
MenuNum=8; MenuItem=0;
Mdefaul=WaitLoop(71,200,98,115);
if(Mdefaul==1)
if(Mdefaul==1){
WorkingArea(40,100,"FET Teshold");
EndSelect=1; MenuNum=0;
MouseHide();
setcolor(2);setfillstyle(1,2);
bar(MenuS[MenuNum].iFrame[0],MenuS[MenuNum].iFrame[1],
```

```
MenuS[MenuNum].iFrame[2],MenuS[MenuNum].iFrame[3]);
setcolor(0);
rectangle(MenuS[MenuNum].iFrame[0]+1,MenuS[MenuNum].iFrame[1]+1,
MenuS[MenuNum].iFrame[2]-1.MenuS[MenuNum].iFrame[3]-1);
for(i=1;i<=MenuS[MenuNum].iNoNum-1;i++)
outtextxy(MenuS[MenuNum].iCol,MenuS[MenuNum].iRow[i],
MenuS[MenuNum].cItem[i].Choice);
free((void *)cInstru);
MouseShow();
MovetoArea(20,122);
}
}
/*-- FET Charecteristic --*/
if(x>=71 && x<=200 && y>=123 && y<=140){
MenuNum=9; MenuItem=0;
Mdefaul=WaitLoop(71,200,123,140);
if(Mdefaul==1){
MenuItem=1;
WorkingArea(40,100,"FET Character");
EndSelect=1; MenuNum=0;
MouseHide();
setcolor(2);setfillstyle(1,2);
bar(MenuS[MenuNum].iFrame[0],MenuS[MenuNum].iFrame[1],
MenuS[MenuNum].iFrame[2],MenuS[MenuNum].iFrame[3]);
setcolor(0);
rectangle(MenuS[MenuNum].iFrame[0]+1,MenuS[MenuNum].iFrame[1]+1,
MenuS[MenuNum].iFrame[2]-1,MenuS[MenuNum].iFrame[3]-1);
for(i=1;i<=MenuS[MenuNum].iNoNum-1;i++)
outtextxy(MenuS[MenuNum].iCol,MenuS[MenuNum].iRow[i],
MenuS[MenuNum].cItem[i].Choice);
```

```
free((void *)cInstru);
MouseShow();
MovetoArea(20,122);
}
}

/*--- Transister Charecter --*/
if(x>=71 && x<=200 && y>=146 && y<=166){

MenuNum=10; MenuItem=0;
Mdefaul=WaitLoop(71,200,148,166);
if(Mdefaul==1){
WorkingArea(40,100,"Tansistor Character");
EndSelect=1; MenuNum=0;
MouseHide();
setcolor(2);setfillstyle(1,2);
bar(MenuS[MenuNum].iFrame[0],MenuS[MenuNum].iFrame[1],
MenuS[MenuNum].iFrame[2],MenuS[MenuNum].iFrame[3]);
setcolor(0);
rectangle(MenuS[MenuNum].iFrame[0]+1,MenuS[MenuNum].iFrame[1]+1,
MenuS[MenuNum].iFrame[2]-1,MenuS[MenuNum].iFrame[3]-1);
for(i=1;i<=MenuS[MenuNum].iNoNum-1;i++)
outtextxy(MenuS[MenuNum].iCol,MenuS[MenuNum].iRow[i],
MenuS[MenuNum].cItem[i].Choice);
free((void *)cInstru);
MouseShow();
MovetoArea(20,122);
}

}
```

```
/*--- Oscilloscope Active ---*/
if(x>=71 && x<=200 && y>=172 && y<=190){
MenuNum=11; MenuItem=0;
Mdefaul=WaitLoop(71,200,172,190);
if(Mdefaul==1)
FET_Theshold(40,100,"Oscilloscope");
}
/*--- Close Active --*/
if(x>=71 && x<=200 && y>=197 && y<=214){
MenuNum=12; MenuItem=0;
Mdefaul=WaitLoop(71,200,197,214);
if(Mdefaul==1){
MouseHide();
putimage(MenuS[2].iFrame[0],MenuS[2].iFrame[1],
cInstru,COPY_PUT);
MouseShow();
free((void *)cInstru);
EndSelect=1; MenuNum=0;
MovetoArea(20,122);
}
}
}while(EndSelect==0);
return(RMain);
}

void AbstracMenu(void)
{
register int  buttons;
int  x=220,y=100;
```

```
char far *Abstrac;


MouseHide();
BackWin(x,y,x+320,y+256,1,7,"Histoery of the GPIB");
outtextxy(x+8,10 + y ,"  The original GPIB was designed by");
outtextxy(x+8,30 + y ,"Hewlett-Packard(where it is called the HP-");
outtextxy(x+8,50 + y ,"IB) to connect and control programmable");
outtextxy(x+8,70 + y ,"instruments manufactured by ");
outtextxy(x+8,90 + y ,"Hewlett-Packard Because of its hight data");
outtextxy(x+8,110+ y ,"tansfer rates of up to 1 megabyte/sec,the");
outtextxy(x+8,130+ y ,"GPIB quickly  gained popularity in other");
outtextxy(x+8,150+ y ,"applications  such as intercomputer");
outtextxy(x+8,170+ y ,"communication  and peripheral control.");
outtextxy(x+8,190+ y ,"  It was later accepted the name General");
outtextxy(x+8,210+ y ,"Purpose Interface Bus.");
MouseShow();


do{
regs.x.ax=3;
int86(0x33,&regs,&regs);                    /*-- Mouse information --*/
buttons=regs.x.bx&3;
x=regs.x.cx;
y=regs.x.dx;
/*Position(x,y);*/
if(x>=523 && x<=539 && y>=75 && y<=95
&& buttons==1){
MouseHide();
Block(523,75,539,95,1,8,7,15);
do{
regs.x.ax=3;
```

```
int86(0x33,&regs,&regs);   /*-- Mouse information --*/
buttons=regs.x.bx&3;
}while(buttons!=0);
buttons=3;
MouseShow();

}
}while(buttons!=3);
MouseHide();
setfillstyle(1,7);
bar(214,70,547,362);

MouseShow();
}

void HowtouseMenu(void)
{
printf("");
}

int WaitLoop(int Lx,int Rx,int Ty,int Ly)
{

register int   x,y,buttons=5;
int   Press=0;
int   Ht=3,ButtonRe;

settextstyle(2,0,5);
MouseHide();
size=imagesize(Lx,Ty,Rx,Ly);
```

```
Video=(char *)malloc(size);
getimage(Lx,Ty,Rx,Ly,Video);


setcolor(14); setfillstyle(1,14);
bar(Lx,Ty,Rx,Ly);
setcolor(0);
outtextxy(MenuS[MenuNum].iCol,MenuS[MenuNum].iRow[MenuItem],
MenuS[MenuNum].cItem[MenuItem].Choice);
MouseShow();
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);              /*-- Mouse information --*/
buttons=regs.x.bx&3;
x=regs.x.cx;
y=regs.x.dx;
}while(buttons==0 && x>=Lx-Ht && x<=Rx+Ht
&& y>=(Ty-Ht) && y<=Ly+Ht);
MouseHide();
/*-- Cancle buttons to protect erro --*/
if(buttons==1)
do{
Press++;
if(Press==30) Press=2;
regs.x.ax=3;
int86(0x33,&regs,&regs);              /*-- Mouse information --*/
buttons=regs.x.bx&3;
if(Press==1) ButtonRe=buttons;
}while(buttons!=0);
/*-------------------------*/
```

```
putimage(Lx,Ty,Video,COPY_PUT);
MouseShow();
free((void *)Video);
return(ButtonRe);
}

void Dis_Play(int x,int y)
{

MouseHide();

Block(x+360,y+10,x+560,y+80,1,15,7,8);
setcolor(0);
setlinestyle(0,0,2);
rectangle(x+358,y+8,x+562,y+82);
setlinestyle(0,0,1);

MouseShow();

ButtonPause(x, y,15,7,8);
ButtonPlay(x , y,15,7,8);
ButtonSTOP(x , y,15,7,8);
ButtonsSFF(x , y,15,7,8);
ButtonREC(x  , y,15,7,8);

}

void Block(int Xtop,int Ytop,int Xdown,int Ydown,int Cwid,
int iCleft,int iCcenter,int iCright)
{
```

```
int Up[14];
int Dn[14];

Up[0]=Xtop      ;    Up[1]=Ydown      ;
Up[2]=Xtop-Cwid ;    Up[3]=Ydown+Cwid ;
Up[4]=Xtop-Cwid ;    Up[5]=Ytop-Cwid  ;
Up[6]=Xdown+Cwid ;   Up[7]=Ytop-Cwid  ;
Up[8]=Xdown     ;    Up[9]=Ytop       ;
Up[10]=Xtop     ;    Up[11]=Ytop      ;
Up[12]=Xtop     ;    Up[13]=Ydown     ;

Dn[0]=Xdown     ;    Dn[1]=Ytop       ;
Dn[2]=Xdown+Cwid ;   Dn[3]=Ytop-Cwid  ;
Dn[4]=Xdown+Cwid ;   Dn[5]=Ydown+Cwid ;
Dn[6]=Xtop-Cwid ;    Dn[7]=Ydown+Cwid ;
Dn[8]=Xtop      ;    Dn[9]=Ydown      ;
Dn[10]=Xdown    ;    Dn[11]=Ydown     ;
Dn[12]=Xdown    ;    Dn[13]=Ytop      ;

setcolor(iCcenter);
setfillstyle(1,iCcenter);
bar(Xtop,Ytop,Xdown,Ydown);

setcolor(iCleft);
setfillstyle(1,iCleft);
fillpoly(7,Up);

setcolor(iCright);
setfillstyle(1,iCright);
fillpoly(7,Dn);
```

```
}


int HoldStatus(void)
{
union REGS regs;
register int buttons=5;
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);   /*-- Mouse information --*/
buttons=regs.x.bx&3;
}while(buttons!=0);
return(3);
}

void MouseShow(void)
{
regs.x.ax=1;
int86(0x33,&regs,&regs);
}

void MouseHide(void)
{
regs.x.ax=2;
int86(0x33,&regs,&regs);
}

void MovetoArea(int x,int y)
{
```

```
union REGS regs;

regs.x.ax=4;
regs.x.cx=x;
regs.x.dx=y;
int86(0x33,&regs,&regs);                    /*-- Mouse show   --*/
}

void ButtonREC(int x,int y,int Cleft,int Ccenter,int Cright)
{

MouseHide();
Block(x+530,y+20,x+550,y+50,1,Cleft,Ccenter,Cright);
setcolor(0);
rectangle(x+528,y+18,x+552,y+52);
setcolor(5);
settextstyle(2,0,4);
outtextxy(x+533,y+56,"REC");
setcolor(4);
circle(x+540,y+35,5);
MouseShow();
setlinestyle(0,0,1);

}


void ButtonsSFF(int x,int y,int Cleft,int Ccenter,int Cright)
{
int SFF[8];
```

```
SFF[0]=x+505;          SFF[1]=y+25;
SFF[2]=x+515;          SFF[3]=y+35;
SFF[4]=x+505;          SFF[5]=y+45;
SFF[6]=x+505;          SFF[7]=y+25;

MouseHide();
Block(x+490,y+20,x+520,y+50,1,Cleft,Ccenter,Cright);
setcolor(0); setfillstyle(1,0);
rectangle(x+488,y+18,x+522,y+52);
setcolor(5);
settextstyle(2,0,4);
outtextxy(x+498,y+56,"SFF");
setcolor(0);
fillpoly(4,SFF);
setlinestyle(0,0,3);
line(x+500,y+25,x+500,y+45);
setlinestyle(0,0,1);

MouseShow();
}

void ButtonSTOP(int x,int y,int Cleft,int Ccenter,int Cright)
{

MouseHide();
Block(x+450,y+20,x+480,y+50,1,Cleft,Ccenter,Cright);
setcolor(0); setfillstyle(1,0);
rectangle(x+448,y+18,x+482,y+52);
bar(x+455,y+28,x+475,y+42);
outtextxy(x+455,y+56,"STOP");
```

```
MouseShow();
setlinestyle(0,0,1);
}

void ButtonSREW(int x,int y,int Cleft,int Ccenter,int Cright)
{
int SREW[8];

SREW[0]=x+465;          SREW[1]=y+25;
SREW[2]=x+455;          SREW[3]=y+35;
SREW[4]=x+465;          SREW[5]=y+45;
SREW[6]=x+465;          SREW[7]=y+25;

MouseHide();
Block(x+450,y+20,x+480,y+50,1,Cleft,Ccenter,Cright);
setcolor(0); setfillstyle(1,0);
rectangle(x+448,y+18,x+482,y+52);
fillpoly(4,SREW);
setlinestyle(0,0,3);
line(x+470,y+25,x+470,y+45);
setlinestyle(0,0,1);
setcolor(5);
settextstyle(2,0,4);
outtextxy(x+455,y+56,"REW");
MouseShow();
setlinestyle(0,0,1);
}
```

```
void ButtonPlay(int x,int y,int Cleft,int Ccenter,int Cright)
{
int Play[8];
Play[0]=x+410;          Play[1]=y+25;
Play[2]=x+430;          Play[3]=y+35;
Play[4]=x+410;          Play[5]=y+45;
Play[6]=x+410;          Play[7]=y+25;

MouseHide();
Block(x+400,y+20,x+440,y+50,1,Cleft,Ccenter,Cright);
setcolor(0); setfillstyle(1,0);
rectangle(x+398,y+18,x+442,y+52);
fillpoly(4,Play);
setcolor(5);
settextstyle(2,0,4);
outtextxy(x+410,y+56,"PLAY");
MouseShow();
setlinestyle(0,0,1);
}

void ButtonPause(int x,int y,int Cleft,int Ccenter,int Cright)
{
MouseHide();
Block(x+370,y+20,x+390,y+50,1,Cleft,Ccenter,Cright);
setlinestyle(0,0,2);
rectangle(x+368,y+18,x+392,y+52);
setcolor(0);
setlinestyle(0,0,3);
line(x+377,y+26,x+377,y+44);
line(x+382,y+26,x+382,y+44);
```

```
line(x+500,y+25,x+500,y+45);
line(x+470,y+25,x+470,y+45);
setcolor(5);
setlinestyle(0,0,1);
settextstyle(2,0,4);
outtextxy(x+368,y+56,"STILL");
MouseShow();

}


int WorkingArea(int x,int y,char *Message)
{

register int    buttons;
int     j,i;
int     ReturnV=0;
int     ActivePlay=0;
int     TrActive=1;
int     Mdefaul;
int     xPlay,yPlay;
int     m,n;
FILE     *fp;
char     Mess[20];
int     TemM,TemI;


MouseHide();
```

```
i    =x;          j    =y;
xPlay=x+10;       yPlay=y+250;
MouseShow();
strcpy(Mess,"Input Signal");
BackWin(x,y,x+320,y+256,1,7,Mess);
Screen(x,y);
BackWin(x+40,y+40,x+360,y+296,1,7,Message);
Screen(x+40,y+40);

Dis_Play(xPlay,yPlay);
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);              /*-- Mouse information --*/
buttons=regs.x.bx&3;
x=regs.x.cx;
y=regs.x.dx;
/*Position(x,y);*/

if(x>=14 && x<=51 && y>=42 && y<=60){
TemM=MenuNum; TemI=MenuItem;
MenuNum=0;    MenuItem=0;
ActivePlay=1;
Mdefaul=WaitLoop(14,51,42,60);
if(Mdefaul==1){
MenuNum=TemM; MenuItem=TemI;
Block(409,359,612,433,1,7,7,7);
FileWork();
Dis_Play(xPlay,yPlay);
ActivePlay=0;
}
```

```
}

/*--- Active Menu 0---*/
if(x>=62 && x<=340 && y>=74 && y<=95 && buttons==1 &&
TrActive==1){
HoldStatus();
TrActive=0;
MouseShow();
BackWin(i,j,i+320,j+256,1,7,Mess);
Screen(i,j);
}
/*--- Active Menu 1---*/
if(x>=102 && x<=379 && y>=114 && y<=136 && buttons==1 &&
TrActive==0){
HoldStatus();
TrActive=1;
MouseShow();
BackWin(i+40,j+40,i+360,j+296,1,7,Message);
Screen(i+40,j+40);

}
/*--- Input Signal Active ---*/
if(x>=39 && x<=60 && y>=75 && y<=95 && buttons==1 &&
TrActive==0){

HoldStatus();
SignalSet();
MouseHide();
Screen(i,j);
MouseShow();
```

```
}
if(ActivePlay==0){
/*--- Pause Active ---*/
if(x>=xPlay+372 && x<=xPlay+389 && y>=yPlay+21 && y<=yPlay+49
&& buttons==1){
ButtonPause(xPlay,yPlay,8,7,15);
buttons=HoldStatus();
ButtonPause(xPlay,yPlay,15,7,8);
}
/*--- Play Active ---*/
if(x>=xPlay+401 && x<=xPlay+439 && y>=yPlay+21 && y<=yPlay+49
&& buttons==1){
fp=fopen("BC556","r");
fread(cWave,sizeof cWave,1,fp);
fclose(fp);
ButtonPlay(xPlay,yPlay,8,7,15);
HoldStatus();
ButtonPlay(xPlay,yPlay,15,7,8);
Delay=0;
if(TrActive==0){
PlotWave(1,14,i,j);
PlotWave(2,0,i,j);
}else XYPlot(i+40,j+40,4);
}
/*--- STOP Active ---*/
if(x>=xPlay+450 && x<=xPlay+479 && y>=yPlay+21 && y<=yPlay+49
&& buttons==1){
ButtonSTOP(xPlay,yPlay,8,7,15);
buttons=HoldStatus();
```

```
ButtonSTOP(xPlay,yPlay,15,7,8);
setfillstyle(1,7);
MouseHide();
bar(30,68,620,440);
MouseShow();
}
/*--- SFF Active ---*/
if(x>=xPlay+489 && x<=xPlay+519 && y>=yPlay+21 && y<=yPlay+49
&& buttons==1){
ButtonsSFF(xPlay,yPlay,8,7,15);
HoldStatus();
ButtonsSFF(xPlay,yPlay,15,7,8);
Delay=100;
if(TrActive==0){
PlotWave(1,14,i,j);
PlotWave(2,0,i,j);
}else XYPlot(i+40,j+40,14);
}
/*--- REC Active ---*/
if(x>=xPlay+531 && x<=xPlay+549 && y>=yPlay+21 && y<=yPlay+49
&& buttons==1){
ButtonREC(xPlay,yPlay,8,7,15);
buttons=HoldStatus();
ButtonREC(xPlay,yPlay,15,7,8);
}
}

}while(buttons!=3);
settextstyle(2,0,5);
MovetoArea(75,99);
```

```
return(ReturnV);

}


void SignalSet(void)
{
int   buttons,x,y;
int   Mdefaul;
char *Pch1,*Pch2;
char *PGamp,*PGfrq;
unsigned size;
char TemCh[10];

MouseHide();
Block(90,305,210,328,1,15,7,8);
settextstyle(2,0,6);
outtextxy(140,308,"OK");
Block(222,305,342,328,1,15,7,8);
outtextxy(250,308,"Cancle");
setcolor(11); settextstyle(2,0,5);
outtextxy(90,130,"Power Supply");
outtextxy(122,155,"Ch1");   outtextxy(312,155,"Volt");
outtextxy(122,175,"Ch2");   outtextxy(312,175,"Volt");

size=imagesize(200,154,260,170);
Pch1=(char *)malloc(size);
getimage(200,154,260,170,Pch1);

Pch2=(char *)malloc(size);
```

```
getimage(200,176,260,192,Pch2);

PGamp=(char *)malloc(size);
getimage(200,228,260,244,PGamp);

PGfrq=(char *)malloc(size);
getimage(200,250,260,266,PGfrq);

outtextxy(90,190,"Function Gen");
outtextxy(122,228,"Amp");          outtextxy(312,228,"Volt");
outtextxy(122,253,"Frequency");    outtextxy(312,253,"Hz");

sprintf(TemCh,"%3.2f",GAmp);
outtextxy(220,227,TemCh);
sprintf(TemCh,"%4.2f",Gfrq);
outtextxy(205,252,TemCh);
sprintf(TemCh,"%2.2f",Bch2);
outtextxy(220,175,TemCh);
sprintf(TemCh,"%2.2f",Bch1);
outtextxy(220,153,TemCh);

Block(266,154,290,170,1,15,7,8);    UpA(278,159);
Block(266,176,290,192,1,15,7,8);    UpA(278,181);
Block(266,228,290,244,1,15,7,8);    UpA(278,233);
Block(266,250,290,266,1,15,7,8);    UpA(278,255);

MouseShow();
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);
```

```
buttons=regs.x.bx&3;
x=regs.x.cx;
y=regs.x.dx;
/*Position(x,y);*/


/*-- Power supply up Arrow Ch1--*/
if(x>=266 && x<=290 && y>=156 && y<=170 && buttons==1){
Bch1++; if(Bch1>=30) Bch1=0;
MouseHide();
sprintf(TemCh,"%2.2f",Bch1);
Block(266,154,290,170,1,8,7,15); UpA(278,159);
putimage(200,154,Pch1,COPY_PUT);
outtextxy(220,153,TemCh);
MouseShow();
HoldStatus();
MouseHide();
Block(266,154,290,170,1,15,7,8); UpA(278,159);
MouseShow();
}

/*-- Power supply up Arrow Ch2--*/

if(x>=266 && x<=290 && y>=176 && y<=192 && buttons==1){
Bch2++; if(Bch2==30) Bch2=0;
MouseHide();
sprintf(TemCh,"%2.2f",Bch2);
Block(266,176,290,192,1,8,7,15); UpA(278,181);
putimage(200,176,Pch2,COPY_PUT);
outtextxy(220,175,TemCh);
```

```
MouseShow();
HoldStatus();
MouseHide();
Block(266,176,290,192,1,15,7,8); UpA(278,181);
MouseShow();
}
/*-- Function Gen Amp --*/
if(x>=266 && x<=290 && y>=228 && y<=244 && buttons==1){
GAmp++; if(GAmp==30) GAmp=0;
MouseHide();
sprintf(TemCh,"%2.2f",GAmp);
Block(266,228,290,244,1,8,7,15); UpA(278,233);
putimage(200,228,PGamp,COPY_PUT);
outtextxy(220,227,TemCh);
MouseShow();
HoldStatus();
MouseHide();
Block(266,228,290,244,1,15,7,8); UpA(278,233);
MouseShow();
}
/*-- Function Gen Frequency--*/

if(x>=266 && x<=290 && y>=250 && y<=266 && buttons==1){
Gfrq+=500; if((int)Gfrq==5000) Gfrq=0.0;
MouseHide();
sprintf(TemCh,"%4.2f",Gfrq);
Block(266,250,290,266,1,8,7,15); UpA(278,255);
putimage(200,250,PGfrq,COPY_PUT);
outtextxy(205,252,TemCh);
MouseShow();
```

```
HoldStatus();
MouseHide();
Block(266,250,290,266,1,15,7,8); UpA(278,255);
MouseShow();
}

/*--- OK Avtive ---*/
if(x>=90 && x<=210 && y>=305 && y<=328 && buttons==1){
MouseHide();
Block(90,305,210,328,1,8,7,15);
settextstyle(2,0,6);
outtextxy(140,308,"OK");
MouseShow();
buttons=HoldStatus();
MouseHide();
Block(90,305,210,328,1,15,7,8);
settextstyle(2,0,6);
outtextxy(140,308,"OK");
MouseShow();
settextstyle(2,0,5);
}

/*--- Cancle Avtive ---*/
if(x>=222 && x<=342 && y>=305 && y<=328 && buttons==1){
MouseHide();
Block(222,305,342,328,1,8,7,15);
settextstyle(2,0,6);
outtextxy(250,308,"Cancle");
MouseShow();
buttons=HoldStatus();
```

```
MouseHide();
Block(222,305,342,328,1,15,7,8);
settextstyle(2,0,6);
outtextxy(250,308,"Cancle");
MouseShow();
settextstyle(2,0,5);

}

}while(buttons!=3);

free((void *)Pch1);
free((void *)Pch2);
free((void *)PGamp);
free((void *)PGfrq);
}

void UpA(int x,int y)
{
moveto(x+3,y+5);
lineto(x,y);
lineto(x-3,y+5);
}

void DnA(int x,int y)
{
moveto(x+3,y);
lineto(x,y+5);
lineto(x-3,y);
```

```
}

void FileWork()
{
int buttons,x,y;

MouseHide();
Block(500 ,70 ,600 ,90,1,15,7,8);
Block(500 ,110,600 ,130,1,15,7,8);
Block(500 ,150,600 ,170,1,15,7,8);
outtextxy(520,75 ,"File Open");
outtextxy(520,115,"File Save");
outtextxy(520,155,"< Close >");

MouseShow();
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);
buttons=regs.x.bx&3;
x=regs.x.cx;
y=regs.x.dx;
/*Position(x,y);*/
/*-- Open file --*/
if(x>=500 && x<=600 && y>=70 && y<=90 && buttons==1){
MouseHide();
Block(500 ,70 ,600 ,90,1,8,7,15);
outtextxy(520,75 ,"File Open");
HoldStatus();
OPenFile();
Block(500 ,70 ,600 ,90,1,15,7,8);
```

```
outtextxy(520,75 ,"File Open");
MouseShow();
}
/*-- Save file --*/
if(x>=500 && x<=600 && y>=110 && y<=130 && buttons==1){
MouseHide();
Block(500 ,110,600 ,130,1,8,7,15);
outtextxy(520,115,"File Save");
HoldStatus();
Block(500 ,110,600 ,130,1,15,7,8);
outtextxy(520,115,"File Save");
MouseShow();
}

/*-- Close file --*/
if(x>=500 && x<=600 && y>=150 && y<=170 && buttons==1){
MouseHide();
Block(500 ,150,600 ,170,1,8,7,15);
outtextxy(520,155,"< Close >");
buttons=HoldStatus();
Block(500 ,150,600 ,170,1,15,7,8);
outtextxy(520,155,"< Close >");
MouseShow();
}

}while(buttons!=3);
MouseHide();
Block(499 ,69 ,601 ,171,1,7,7,7);
MouseShow();
}
```

```
void OPenFile(void)
{
unsigned size;
char    *Openf;
char    *Labelf;
char    *File;
int     buttons,x,y;
char    *Item[100];
char    *TempCh[6];
int     n=0,nFile=0;
FILE    *fp;
struct SAVE{
int n;
char fname[10];
char ScWave[4096];
}wave;



size=imagesize(299 ,69 ,451 ,91);
Labelf=(char *)malloc(size);
getimage(299 ,69 ,451 ,91,Labelf);

Block(300 ,70 ,450 ,90,1,15,7,8);
outtextxy(320,75 ,"Select Devive ");

size=imagesize(299 ,119 ,451 ,301);
Openf=(char *)malloc(size);
getimage(299 ,119 ,451 ,301,Openf);
```

```
Block(300 ,120 ,450 ,300,1,15,7,8);
Block(310 ,120 ,430 ,300,1,8,3,15);
Block(434 ,125 ,448 ,150,1,15,7,8); UpA(441,135);
Block(434 ,270 ,448 ,295,1,15,7,8); DnA(441,280);

Item[0] ="TR AC331";
Item[1] ="TR BC331";
Item[2] ="TR CC332";
Item[3] ="TR DC333";
Item[4] ="TR EC334";
Item[5] ="TR FC335";
Item[6] ="TR GC336";
Item[7] ="TR HC337";
Item[8] ="TR IC338";
Item[9] ="TR JC339";
Item[10]="TR KC317";
Item[11]="TR LC327";
Item[12]="TR MC337";
Item[13]="";

size=imagesize(314 ,145 ,405 ,270);
File=(char *)malloc(size);
getimage(314 ,145 ,405 ,270,File);

TempCh[0]=Item[0];
TempCh[1]=Item[1];
TempCh[2]=Item[2];
TempCh[3]=Item[3];
```

```
TempCh[4]=Item[4];
TempCh[5]=Item[5];



outtextxy(320,150,Item[0]);
outtextxy(320,170,Item[1]);
outtextxy(320,190,Item[2]);
outtextxy(320,210,Item[3]);
outtextxy(320,230,Item[4]);
outtextxy(320,250,Item[5]);

MouseShow();
do{
regs.x.ax=3;
int86(0x33,&regs,&regs);
buttons=regs.x.bx&3;
x=regs.x.cx;
y=regs.x.dx;
/*Position(x,y);*/
/*-- Open file --*/
if(x>=500 && x<=600 && y>=70 && y<=90 && buttons==1){
MouseHide();
Block(500 ,70 ,600 ,90,1,15,7,8);
outtextxy(520,75 ,"File Open");
MouseShow();
buttons=HoldStatus();
}
/*-- Up Arrow --*/
if(x>=434 && x<=488 && y>=125 && y<=150 && buttons==1){
MouseHide();
```

```
-;
if(n<0){
nFile--; n=0;
if(nFile<=0) nFile=0;
TempCh[0]=Item[nFile];
TempCh[1]=Item[1+nFile];
TempCh[2]=Item[2+nFile];
TempCh[3]=Item[3+nFile];
TempCh[4]=Item[4+nFile];
TempCh[5]=Item[5+nFile];
}
putimage(314 ,145 ,File,COPY_PUT);
bar(318,146+n*20,400,166+n*20);
outtextxy(320,150,TempCh[0]);
outtextxy(320,170,TempCh[1]);
outtextxy(320,190,TempCh[2]);
outtextxy(320,210,TempCh[3]);
outtextxy(320,230,TempCh[4]);
outtextxy(320,250,TempCh[5]);

Block(434 ,125 ,448 ,150,1,8,7,15); UpA(441,135);
MouseShow(); HoldStatus(); MouseHide();
Block(434 ,125 ,448 ,150,1,15,7,8); UpA(441,135);
MouseShow();
}
/*-- Down Arrow --*/
if(x>=434 && x<=488 && y>=270 && y<=295 && buttons==1){
MouseHide();
+;
if(n>5){
```

```
nFile++; n=5;
if(strlen(Item[5+nFile])<=2) nFile--;
TempCh[0]=Item[nFile];
TempCh[1]=Item[1+nFile];
TempCh[2]=Item[2+nFile];
TempCh[3]=Item[3+nFile];
TempCh[4]=Item[4+nFile];
TempCh[5]=Item[5+nFile];

putimage(314 ,145 ,File,COPY_PUT);
bar(318,146+n*20,400,166+n*20);
outtextxy(320,150,TempCh[0]);
outtextxy(320,170,TempCh[1]);
outtextxy(320,190,TempCh[2]);
outtextxy(320,210,TempCh[3]);
outtextxy(320,230,TempCh[4]);
outtextxy(320,250,TempCh[5]);
}
putimage(314 ,145 ,File,COPY_PUT);
bar(318,146+n*20,400,166+n*20);
outtextxy(320,150,TempCh[0]);
outtextxy(320,170,TempCh[1]);
outtextxy(320,190,TempCh[2]);
outtextxy(320,210,TempCh[3]);
outtextxy(320,230,TempCh[4]);
outtextxy(320,250,TempCh[5]);
Block(434 ,270 ,448 ,295,1,8,7,15); DnA(441,280);
MouseShow(); HoldStatus(); MouseHide();
Block(434 ,270 ,448 ,295,1,15,7,8); DnA(441,280);
MouseShow();
```

```
}

/*-- Ok --*/
/*
if(x>=500 && x<=600 && y>=150 && y<=170 && buttons==1){
MouseHide();
Block(500 ,150,600 ,170,1,8,7,15);
outtextxy(520,155,"<  O K  >");
buttons=HoldStatus();
Block(500 ,150,600 ,170,1,15,7,8);
outtextxy(520,155,"< Close >");

fp=fopen("AAA","r");
while(fread(&wave,sizeof wave,1,fp)==1){
strcpy(cWave,wave.ScWave);
};
MouseShow();
}
*/

}while(buttons!=3);
MouseHide();
putimage(299 ,119 ,Openf ,COPY_PUT);
putimage(299 ,69  ,Labelf,COPY_PUT);
free((void *)Openf);
free((void *)Labelf);
free((void *)File);
}
```

บรรณานุกรม

1. ธันวา ศรีประมง "การเขียนโปรแกรมภาษา C สำหรับวิศวกรรม"

2. กนก เจนจิรพงศ์เวช , วารสารเซมิคอนดัคเตอร์ ฉบับที่ 78 บทความเรื่อง "GPIB (IEEE 488) บัสอินเตอร์เฟส มาตรฐานและการใช้"

3. น.ส. กฤษรา อำพลจันทร์ และ น.ส. นุช จาคจวบสินธ์ "การควบคุมโดยใช้มาตรฐาน IEEE-488 (GPIB)", ปีการศึกษา 2537

4. ธานินทร์ ถาวรศาศนวงศ์ และ ทินกร คุ๊ก "การอินเตอร์เฟส IBM/PC"

5. National Instrument Corporation, "NI-488.2TM Solfware Reference Manual for MS-Dos"

กิตติกรรมประกาศ

ทางผู้จัดทำปริญญานิพนธ์ขอกราบขอบพระคุณ รศ.ดร. มนัส สังวรศิลป์ เป็นอย่างยิ่งซึ่งได้ให้คำแนะนำในการจัดทำโครงงานฉบับนี้จนเสร็จสมบูรณ์ และขอขอบคุณ เพื่อนๆพี่ๆน้องๆทุกคนที่ให้คอยกำลังใจและช่วยเหลือในด้านต่างๆ โดยเฉพาะ พี่โจ้ที่มีคอมฯให้ใช้, น้องสาวยุทธ(เปิ้ล), วัติ,เพ็ญ ที่ช่วยจัดทำในด้านเอกสาร และ ร้านค้าร้านอาหารต่างๆที่มีอาหารในยามราตรีให้รับประทาน สุดท้ายขอทุกๆคนจงมีแต่สิ่งที่หวัง