เครื่องวัด บันทึกสัญญาณและวิเคราะห์แถบความถี่โดยใช้ไมโครคอมพิวเตอร์

DIGITAL STORAGE OSCILLOSCOPE AND THE SPECTRUM ANALYZER BY MICROCOMPUTER OR COMPATIBLE

โดย

นาย สุริยนต์ พรหมห์วงค์

นาย สุทธิไกร เทียงดาห์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขา เทคโนโลยีอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

หัวข้อปริญญานิพนธ์ DIGITAL STORAGE OSCILLOSCOPE AND THE SPECTRUM ANALYZER BY MICROCOMPUTER OR COMPATIBLE

โดยนาย นาย สุริยนต์ พรหมห์วงค์
นาย สุทธิไกร เทียงตาห์
ภาควิชา เทคนิคอุตสาหกรรม
อาจารย์ที่ปรึกษา อ. อรรถสิทธิ์ หล่าสกุล
อ. ไพศาล สิทธิโยภาสกุล

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์

คณะกรรมการสอบปริญญานิพนธ์

.....อาจารย์ที่ปรึกษา
(.....)

.....กรรมการ
(.....)

.....กรรมการ
(.....)

.....กรรมการ
(.....)

.....กรรมการ
(.....)

เครื่องวัด บันทึกสัญญาณและวิเคราะห์แถบความถี่โดยใช้ไมโครคอมพิวเตอร์

DIGITAL STORAGE OSCILLOSCOPE AND THE SPECTRUM ANALYZER BY MICROCOMPUTER OR COMPATIBLE

โดย นาย สุริยนต์ พรหมห์วงค์

นาย สุทธิไกร เทียงดาห์

อาจารย์ที่ปรึกษา อ. อรรถสิทธิ์ หล่าสกุล

อ. ไพศาล สิทธิโยภาสกุล

บทคัดย่อ

รายงานฉบับนี้ได้นำเสนอการออกแบบและพัฒนา DIGITAL STORAGE OSCILLOSCOPEที่ใช้งานบนเครื่อง PC (IBM หรือ COMPATIBLE) ทั่วๆ ไป โดยจะมี 2 ส่วนคือส่วนที่เป็น HARD WARE เสียบที่สลอตของเครื่อง PC และส่วน SOFTWARE ที่ใช้ควบคุมการทำงานของ CARD โดยใช้ ADC แบบ FLASH วัดสัญญาณแรงดันต่ำไม่เกิน±5 โวลท์ ความถี่ไม่เกิน 1เม็กกะเฮิร์ท การแสดงผล MONOCHROME,VGA,EGA สามารถแสดงรูปสัญญาน และสเปกตรัมของสัญญาณพร้อมทั้งเก็บข้อมูลลงบนแผ่นดิสค์ และพิมพ์รูปสัญญาณออกทางเครื่องพิมพ์

ABSTRACT

THIS THESIS PRESENTS DESIGN AND DEVELOPMENT OF THE DIGITAL STORAGE OSILLOSCOPE AND THE SPECTRUM ANALYZER BY IBM PC OR COMPATIBLEWHICH IS USED WITH PC (IBM OR COMPATIBLE) IN GENERAL. IT CONSISTS OF TWO PARTS : HARDWARE AND SOFT WARE. THE HARDWARE IS INSERTED INTO THE SLOT OF THE PC WHILE THE SOFTWARE CONTROLS THE CARD. THIS SYSTEM CAN PICK UP SIGNAL LOW VOLTAGE BETWEEN ± 5 VOLTS AND FREQUENCY NOT MORE THAN 1 MHz, DISPLAY ON VGA,MONOCHOME,OR EGA SCREEN, IT CAN DISPLAY THE SIGNAL AND THE SPECTRUM OF THE SIGNAL AND ALSO STORE DATA INTO DISK AND ALSO DISPLAY THE SIGNAL TO PRINTER.

สารบัญ

บทที่ 1
บทนำ

ที่มาของโครงงาน

เริ่มจากที่เครื่องออสซิลโลสะโคป SCILLOSCOPEที่เราใช้กันอยู่ในปัจจุบันนี้จะบันทึกข้อมูลหรือสัญญาณไม่ได้ OSCILLOSCOPE รุ่นใหม่ที่จะเก็บสัญญาณไว้หรือบันทึกข้อมูลไว้ได้ก็มีราคาสูงมากจึงได้มีแนวคิดว่าเราน่าจะสร้างเครื่อง STORAGE OSCILLOSCOPE ที่สามารถใช้งานร่วมกับเครื่อง MICROCOMPUTER แบบ PC ได้ เพื่อที่เราจะสามารถแสดงผลหรือรูปร่างของสัญญาณที่เป็นแบบ ANALOG บนอภาพ ของเครื่อง MICROCOMPUTER แบบ PCและบันทึกสัญญาณบนแผ่น DISK และเรียกออกมาใช้ได้เมื่อต้องการ ซึ่งจะดีกว่า OSCILLOSCOPE รุ่นก่อนๆ ทำให้เกิดความสะดวกเพราะในปัจจุบันนี้ COMPUTER แบบ PC มีใช้อยู่ทั่วไปเราจึงสามารถสร้าง HARDWARE มาใช้ร่วมกับเครื่องCOMPUTERที่อยู่ได้นอกจากนั้นเรายังสามารถเคราะห์และแสดงแถบความถี่ของรูปคลื่นได้ (SPECTRUM ANALYZER) และยังสามารถวัดสัญญาณความถี่สูงสุดของสัญญาณได้ถึง 1 MHz จากความต้องการที่กล่าวมาทั้งหมดได้ก่อให้เกิดแนวความคิดว่าเราจะทำโครงงานนี้ขึ้นมาโดยมีข้อแม้อยู่ว่าควรจะมีราคาถูกใช้งานได้ครอบคลุมอย่างกว้างขวางและมีประสิทธิภาพทัดเทียมหรือดีกว่าที่มีขายอยู่ในท้องตลาดและต้องสามารถบันทึกข้อมูลเก็บไว้ในแผ่น DISKETTE ได้ในงานบางอย่างเรามีความจำเป็นที่จะต้องบันทึกข้อมูล ในลักษณะที่เป็นสัญญาณในรูปแบบต่างๆ เช่น สัญญาณ SINE WAVE, SQUARE WAVE, SAW TOOTH หรือ สัญญาณ RANDOW ซึ่งเครื่องมือประเภทหนึ่งที่ทำหน้าที่วัดและแสดงผลข้อมูลที่เป็น ANALOG ทางจอภาพที่เราเรียกว่าOSCILLOSCOPE นั้นมีข้อจำกัดในการใช้งานบางอย่าง คือ ไม่สามารถจัดเก็บรูปสัญญาณ เพื่อตรวจสอบรายละเอียดได้ไม่สามารถเก็บบันทึกข้อมูลเพื่อเป็นหลักฐานไว้ได้ และยังไม่สามารถวิเคราะห์แถบความถี่ของรูปคลื่นสัญญาณได้ในปัจจุบันนี้ ได้มีการสร้างเครื่อง STORAGE OSCILLOSCOPE ขึ้นมาซึ่งสามารถที่จะแก้ไขปัญหาที่กล่าวมาข้างต้นได้แต่มีราคาที่สูงมากเกินกำลังของผู้ที่มีงบประมาณจำกัดจะสามารถซื้อได้

ฉะนั้น จึงได้มีการสร้างเครื่องมือชนิดหนึ่งขึ้นมา เรียกว่า เครื่อง DIGITAL STORAGE OSCILLOSCOPE AND FFT SPECTRUM ANALYZER BY IBM PC OR COMPATIBLE เพื่อแก้ไขปัญหาในจุดนี้ และมีข้อดีคือใช้งานร่วมกับเครื่องคอมพิวเตอร์ทั่วไปที่มีใช้กันอย่างแพร่หลายอยู่แล้ว เพียงแต่นำ CARD ของ HARDWARE เข้าไปเสียบที่ SLOT ของเครื่องคอมพิวเตอร์ และมี SOFTWARE เป็นตัวควบคุมให้คอมพิวเตอร์ทำหน้าที่เป็น STORAGE SCOPE และ SPECTRUM ANALYZER ได้ โดยแสดงผลออกทางจอภาพและเครื่องพิมพ์เช่นเดียวกับเครื่อง STORAGE OSCILLOSCOPE ที่มีจำหน่ายอยู่ทั่วไป

จากที่กล่าวมานี้ จึงได้รวบรวมข้อมูลรายละเอียดทั้งทฤษฎี วงจร และเทคนิคต่างๆ ในการสร้างเครื่อง DIGITAL STORAGE OSCILLOSCOPE & THE SPECTRUM ANALYZER โดยจะแบ่งออกเป็นส่วนต่างๆ ดังนี้

บทที่ 2 - ทฤษฎีและหลักการ

- รายละเอียดของอุปกรณ์
- การ INTERFACE วงจรเข้ากับไมโครคอมพิวเตอร์
- รายละเอียดเกี่ยวกับ SLOT ต่างๆ บน MAIN BOARD XT/AT

บทที่ 3 เป็นการออกแบบวงจรในแต่ละส่วน

บทที่ 4 เป็นส่วนของ SOFTWARE ซึ่งจะใช้ภาษา C เขียนโปรแกรมเพราะภาษามีข้อดีในด้านความเร็วในการทำงาน

บทที่ 5 จะเป็นส่วนของผลการทดลอง ,สรุปผลการทดลอง, ปัญหาที่พบในการทำงาน, การแก้ปัญหาและแนวทางในการพัฒนาโครงงาน

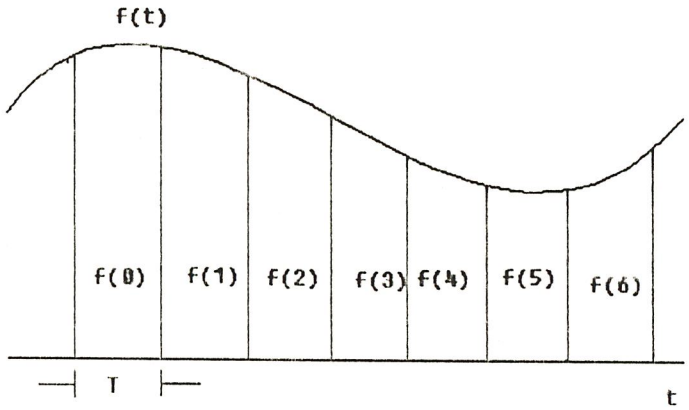
บทที่ 2
ทฤษฎีและหลักการ

2.1 ทฤษฎีการแซมปลิ้ง (SAMPLING)

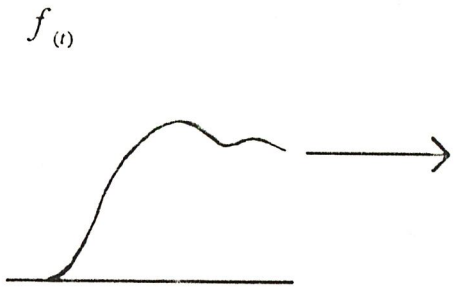
การแซมปลิ้งมีความสำคัญในการติดต่อสื่อสาร ทฤษฎีการแซมปลิ้ง กล่าวว่าสัญญาณในช่วงที่จำกัด (BAND LIMITED) $f_{(t)}$ ที่ไม่มี (SPECTRAL COMPONENT) เกินกว่าความถี่ f_m จะกำหนดโดยค่าของสัญญาณนั้นที่เกิดขึ้นที่เวลาห่างกันเป็นช่วงโดยแต่ละช่วงห่างเท่าๆ กัน ซึ่งแต่ละช่วงเวลานั้นน้อยกว่า $1/2f_m$ วินาที

ทฤษฎี การนี้รู้จักกันในชื่อ UNIFORM SAMPLING THEOREM เพราะว่ามันเกี่ยวข้องกับรายละเอียดของสัญญาณที่กำหนดโดย SAMPLING ของมันที่ช่วงเวลาเท่ากัน คือ $1/2f_m$ วินาที หมายความว่าถ้า FOURIER SPECTRUM ของ $f_{(t)}$ เป็นศูนย์เมื่อความถี่มีค่าสูงกว่า ความถี่~$\omega_m = 2f_m$ แล้วการแซมปลิ้งของมันจะมี INFORMATION ที่สมบูรณ์ของ $f_{(t)}$ อยู่ในตัวซึ่งอยู่ห่างเป็นระยะเวลาที่เท่ากัน ซึ่งน้อยกว่า $1/2f_m$ วินาที ดังแสดงในรูป 2.4 $1/2f_m$ นั้น จะถูกแซมปลิ้งทุกๆ T วินาที โดย T < $1/2f_m$ หรือมีความถี่มากกว่าหรือ เท่ากับ $2f_m$ เมื่อสัญญาณถูกสุ่มออกมาอย่างต่อเนื่อง $(f_0, f_1, \ldots, f_n)$ จากทฤษฎีการ SAMPLING จะเห็นว่าตัวอย่างที่สุ่มมาได้เหล่านี้จะมี INFORMATION ของ $f_{(t)}$ ที่ทุกค่าของ T อย่างไรก็ตามค่า SAMPLING RATE อย่างน้อยที่สุดต้องเป็น 2 เท่าหรือมากกว่าความถี่สูงสุดของ f_m ซึ่งมีอยู่ใน SPECTRUM ของสัญญาณ f_m หรือพูดอีกอย่างหนึ่งว่า สัญญาณจะถูก SAMPLING อย่างน้อย 2 ครั้ง ระหว่างแต่ละคาบเวลาหรือวงรอบของความถี่สูงสุด

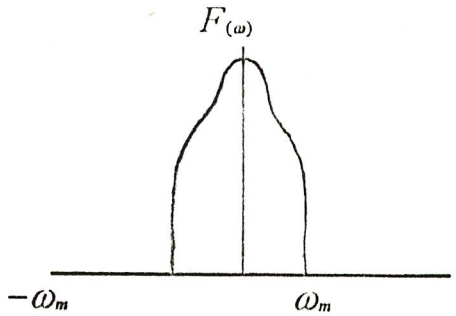
ทฤษฎีการ SAMPLING สามารถพิสูจน์ได้โดยใช้ทฤษฎี FREQUENCY CONVOLUTION เมื่อพิจารณาสัญญาณในช่วงความถี่ที่จำกัด $f_{(t)}$ ซึ่งไม่มี SPECTRAL COMPONENT ที่เกินกว่า f_m หมายความว่า $f_{(\omega)}$ ซึ่งเป็น FOURIER SPECTRUM ของ $f_{(t)}$ มีค่าเป็นศูนย์ เมื่อ $\omega > \omega_m$,$(\omega_m = 2\Pi f_m)$ ถ้าเราคูณฟังก์ชัน $f_{(t)}$ ด้วยฟังก์ชัน PERIODIC IMPULES δ_T (T) ในรูปที่ 2.4-3 เราจะได้ผลลัพธ์ของฟังก์ชันเป็น IMPULSE TRAIN ที่มีระยะห่างเท่ากันกับ T และมีขนาดของ IMPULSE เท่ากับค่าของ $f_{(nt)}$ $(n = \pm 1, \pm 2, \pm 3, \ldots)$ ผลคูณของ $f_{(t)}\,\delta_T$ (T) แทนฟังก์ชัน $f_{(t)}$ ที่ถูก SAMPLING ที่ช่วงเวลาห่างกับ T เราจะแสดงฟังก์ชันที่ถูก SAMPLING นี้โดย $f_{s(t)}$ ดังรูปที่ 2.4-5



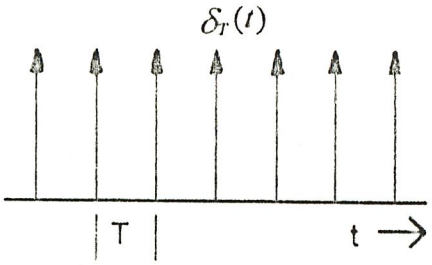
รูปที่ 2.4



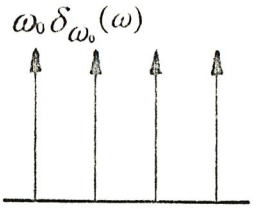
รูปที่ 2.4-1



รูปที่ 2.4-2

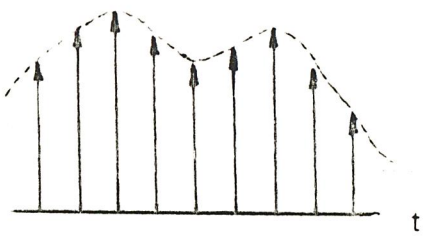


รูปที่ 2.4-3

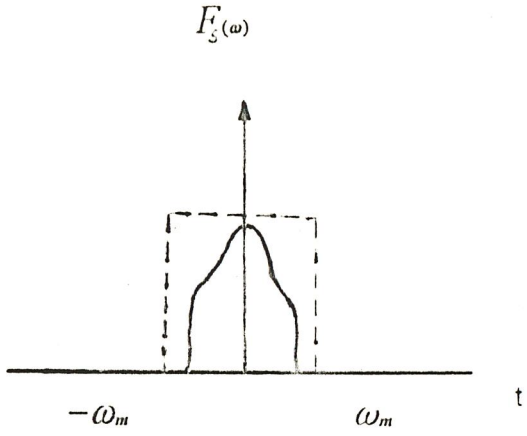


$$\left|\omega_0 = \frac{2\Pi}{T}\right|$$

รูปที่ 2.4-4

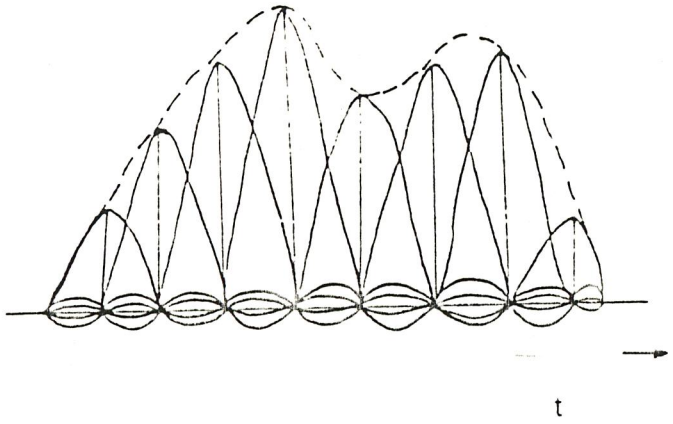


รูปที่ 2.4-5



รูปที่ 2.4-6

$$\left|\omega_0 = \frac{2\Pi}{T}\right|$$



รูปที่ 2.4-7

$$f_{s(t)} = f_{(t)}\delta_{T(t)}$$

ในที่นี้ $\delta_{(x)} = \sum_{n=-\infty}^{\infty}(X - nx)$ ตามคำจำกัดความของ PERIODIC IMPULSE เมื่อให้ FREQUENCY SPECTRUM ของ $f_{(t)}$ คือ $f_{(\omega)}$ FOURIER TRANSFORM ของฟังก์ชั่น IMPULSE ที่ต่อเนื่องกันอย่างสม่ำเสมอคือ $\delta_{T(t)}$ จะมีลักษณะเป็นฟังก์ชั่น IMPULSE คือ $\omega_0\,\delta_{\omega 0}(\omega)$ ที่ต่อเนื่องสม่ำเสมอด้วยดังรูปที่ 2.4-4 IMPULSE จะอยู่ห่างกันเป็นช่วง ความถี่ $\omega_0 = 2/\Pi T$ ซึ่งเท่ากับ

$\delta_{T(t)} = \omega_0 \delta_{\omega 0}(\omega)$

FOURURIER TRANSFORM ของ $f_{(t)}\omega_{T(t)}$ ตามทฤษฎี FREQUENCY CONVOLUTION จะถูกกำหนดโดย CONVOLUTION ของ $f_{(\omega)}$ กับ $\delta_{\omega 0}(\omega)$

$$f_{s(t)} = \frac{1}{2\Pi}[F_\omega * \omega_0 \delta_{\omega 0}(\omega)]$$

แทนค่า $\omega_0 = 2\Pi / T$

$$f_{s(t)} = \frac{1}{T}[F_\omega * \delta_{\omega 0}(\omega)] \qquad (2.1)$$

จากสมการข้างบนจะเห็นได้ชัดว่า SPECTRUM ของสัญญาณ SANMPLING $f_{s(t)}$ ถูกกำหนดโดย CONVOLUTION, ของ $f_{(\omega)}$) กับ IMPULSE $\delta_{\omega 0}(\omega)$ ดังแสดงในรูปที่ 2.4-2 และ รูปที่ 2.4-4 ตามลำดับ เรา สามารถหา CONVOLUTION โดยใช้รูปในการที่จะหา CONVOLUTION โดยใช้รูปนี้ทำได้โดยการพับกลับฟังก์ชั่น $\delta_{\omega 0}(\omega)$ ที่แกน $\omega = 0$ เพราะว่า $\delta_{\omega 0}(\omega)$ เป็นฟังก์ชั่นคู่ที่ถูกพับจะเหมือนกับฟังก์ชั่นเดิมคือ $\delta_{\omega 0}(\omega)$ ใน การที่จะทำตามวิธีการของ CONVOLUTION เราจะเลื่อน IMPULSE ที่ต่อเนื่องกันทั้งหมด $\delta_{\omega 0}(\omega)$ ในทิศ ทาง $+\omega$ ขณะที่แต่ละ IMPUSLE ผ่าน $F_{(\omega)}$ ออกมาวิธีการของ CONVOLUTION ยอมให้ $F_{(\omega)}$ เกิดซ้ำตัวเอง ทุกๆ ω_0 นี้คือฟังก์ชั่น $F_{s(\omega)}$
โดย $F_{(\omega)}$ จะซ้ำตัวเองเป็นระยะโดยปราศจากการซ้อนทับกันเท่าที่ $\omega_0 > 2\omega_m$ หรือ

$$\frac{2\Pi}{T} = 2 * (2\Pi f_m)$$

นั่นคือ

$$T < \frac{f_m}{2} \qquad (2.2)$$

ดังนั้นตราบเท่าที่เรา SAMPLING $f_{(t)}$ ที่ช่วงเวลาเท่าๆ กัน ซึ่งน้อยกว่า $f_m/2$ วินาที $F_{s(\omega)}$ ซึ่ง เป็น SPECTRAL DENSITY FUNCTION ของ $f_{s(T)}$ จะมีการซ้ำเป็นระยะของ $F_{(\omega)}$ และจะมี INFORMATION ทั้ง หมดของ $f_{(t)}$ เราสามารถนำ $F_{(\omega)}$ กลับคืนมาจาก $F_{s(\omega)}$ ได้โดยการให้สัญญาณที่ได้จากการ SAMPLINGที่ผ่านวงจรกรองความถี่ต่ำผ่านซึ่งจะยอมให้ส่วนประกอบที่มีความถี่ต่ำกว่า f_m ผ่าน และจะลด ทอนสัญญาณที่มี

ความถี่มากกว่า f_m คุณสมบัติของ FILTER ในทางอุดมคติดังแสดงในรูปที่ 2.4-6 สังเกตได้ว่าถ้าในช่วงการ SAMPLING T มากกว่า 1/2 fm แล้ว CONVOLUTION ของ $F_{(\omega)}$ กับ $\delta_{\omega 0}\ (\omega)$ จะทำให้เกิดผล $F_{(\omega)}$ เป็นระยะแต่จะเกิดการซ้อนกันเป็นระยะอย่างต่อเนื่องและ $F_{(\omega)}$ ไม่สามารถนำกลับคืนมาจาก $F_{s(\omega)}$ ดังนั้นถ้าช่วงเวลาการ SAMPLING มากเกินไปแล้วข้อมูลบางส่วนจะสูญเสียไปและสัญญาณ $f_{(t)}$ ไม่สามารถนำกลับคืนมาจากสัญญาณ SAMPLING $f_{s(t)}$ ช่องของการสุ่มตัวอย่างมากที่สุด $T = 1/2f_m$
ถูกเรียกว่า NYQUIST INTERVAL
$F_{(\omega)} * \delta_{\omega 0}\ (\omega)$ หาได้โดยรูปแล้ว ยังสามารถหาได้จากการวิเคราะห์

$$\begin{aligned}\delta_{\omega 0}\ (\omega) &= \delta_{\omega} + \delta_{(\omega-\omega_0)} + \cdots + \delta_{(\omega-n\omega_0)} + \cdots \delta_{(\omega+\omega_0)} + \cdots \delta_{(\omega+n\omega_0)} \\ &= \sum_{n=-\infty}^{\infty} (\omega - n\omega_0)\end{aligned}$$

จากสมการ 2.1 จะได้

$$\begin{aligned}F_{s(\omega)} &= \frac{1}{T}\left[F_{(\omega)} * \delta_{\omega 0}\ (\omega)\right] \\ &= \frac{1}{T}\left[F_{(\omega)} * \sum_{n=-\infty}^{\infty} \delta_{(\omega-n\omega_0)}\right]\end{aligned}$$

จาก $f_{(t)} * \delta_{(t-T)} = f_{(t-T)}$
จะได้

$$F_{s(\omega)} = \frac{1}{T} \sum_{n=-\infty}^{\infty} F_{(\omega-n\omega_0)} \qquad (2.3)$$

ด้านขวามือของสมการข้างบนแทนฟังก์ชั่น $F_{(\omega)}$ ซึ่งซ้ำตัวเองทุกๆ ω_0 เรเดียนต่อวินาที วิธีการนี้มีความเที่ยงตรงใกล้กับที่ได้จากรูป

การนำ $f_{(t)}$ กลับคืนมาจาก SAMPLE

ดังได้กล่าวมาแล้วว่าฟังก์ชั่นเริ่มต้นสามารถนำกลับคืนมาโดยการนำ SAMPLE ฟังก์ชั่นผ่านวงจรกรองความถี่ต่ำ ที่มีความถี่ CUT OFF ω_m ซึ่งเป็นการทำงานที่ปรากฏชัดได้ในโดเมนความถี่ ที่มีการทำงานเทียบเท่ากับในโดเมนเวลา ที่จะนำ $f_{(t)}$ กลับคืนมาจาก SAMPLE ของมันต่อไปนี้เราพิจารณาความเป็นไปได้ของมัน พิจารณาสัญญาณ $f_{(t)}$ SAMPLE ด้วยอัตรา $2f_m$ ในกรณีนี้

$$T = \frac{1}{2} f_m \text{ ,} \omega_0 = 4\Pi f_m = 2\omega_m$$

ดังนั้นสมการ 2.3 จะเป็น

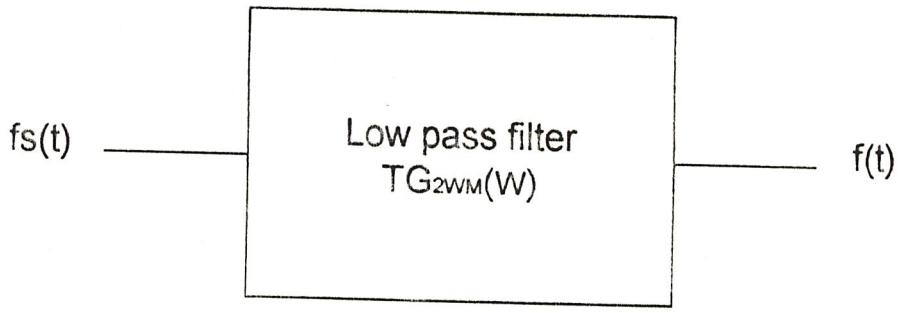
$$F_{s(\omega)} = \frac{1}{T}\sum_{n=-\infty}^{\infty} F_{(\omega-2n\omega_m)} \qquad (2.4)$$

ได้สังเกตมาแล้วว่า SPECTRUM $F_{(\omega)}$ เกิดขึ้นได้โดยผ่าน $F_{s(\omega)}$ เข้าวงจรกรองความถี่ต่ำซึ่งมีความถี่คัท ออฟเท่ากับ ω_m เป็นที่ทราบกันดีแล้วว่าการทำงานของวงจร FILTER เทียบเท่าการคูณ $F_{s(\omega)}$ ด้วย$G_{2\omega_m}(\omega)$ ดังนั้นจากสมการ 2.4 จะได้

$$F_{s(\omega)}G_{2\omega_m}(\omega) = \frac{1}{T}F_{(\omega)}$$

ดังนั้น

$$F_{(\omega)} = TF_{s(\omega)}G_{2\omega_m}(\omega) \qquad (2.5)$$



รูปที่ 2.5 เมื่อป้อน $f_{s(t)}$ ผ่านวงจรกรองความถี่ต่ำจะได้ $f_{(t)}$

ดังนั้นการส่งสัญญาณ $f_{s(t)}$ ผ่านวงจรกรองความถี่ต่ำจะได้สัญญาณ $f_{(t)}$ ออกมาฟิลเตอร์นี้มีความถี่คัท ออฟ ω_m และ $T = 1/2f_m$ ทรานเฟอร์ฟังก์ชั่น $H_{(\omega)}$ ของวงจรฟิลเตอร์ ดังรูปที่ 2.3 สามารถแสดงได้ดังนี้

$$H_{(\omega)} = TG_{2\omega_m}(\omega)$$
$$= \frac{1}{2}f_m G_{2\omega_m}(\omega)$$

โดยการประยุกต์ทฤษฎี TIME CONVOLUTION ในสมการ 2.5 จะได้

$$f_{(t)} = Tf_{s(t)} * \frac{\omega_m}{\Pi} S_a(\omega)_{mt} * f_{s(t)} * S_a(\omega)_{mt} \qquad (2.6)$$

$f_{s(t)}$ ถูกกำหนดโดย

$$f_{s(t)} = \sum_n f_{n(t-nt)} * S_a(\omega)_{mt} \qquad (2.7)$$

ซึ่ง f_n คือ SAMPLE ตัวที่ n ของ $f_{(t)}$ ดังนั้น

$$\begin{aligned} f_{(t)} &= \sum_n f_{n(n-nt)} * S_a(\omega)_{mt} \\ &= \sum_n f_n S_a[\omega_{m(t-nt)}] \qquad (2.7\text{-}1) \\ &= \sum_n f_n S_a \omega_{m(t-nt)} \qquad (2.7\text{-}2) \end{aligned}$$

จะเห็นได้ว่า $f_{(t)}$ สามารถสร้างได้ในโดเมนเวลาจาก SAMPLE ของมันตามสมการ 2.7 สำหรับรูปนั้นแต่ละ SAMPLE ฟังก์ชั่นอีกอันหนึ่งและผลลัพธ์ของสัญญาณทั้งหมดจะถูกรวมกันและได้ $f_{(t)}$ ออกมาดังรูปที่ 2.4-7ในทางปฏิบัติ SPECTRUL DENSITY FUNCTION จะลดลงเมื่อความถี่สูงขึ้นพลังงานส่วนใหญ่จะอยู่ใน COMPONENT ที่อยู่ในช่วงความถี่หนึ่ง สำหรับวัตถุประสงค์ในการใช้ประโยชน์ทั้งหมด

ทฤษฎีการ SAMPLING นั้นเป็นหลักการสำคัญอันหนึ่งโดยอันดับของ SAMPLE ที่แยกเป็นส่วนๆ โดยไม่มีการสูญเสีย INFORMATION ของสัญญาณอินพุทก่อน SAMPLING ซึ่งเทียบกับส่วนที่เป็น DISCRETE ของ INFORMATION เพราะหลักการของ SAMPLING กำหนดจำนวนน้อยที่สุดของค่า DISCRETE ที่จะเป็นในการสร้างสัญญาณ CONTINOUS กลับคืนมา

จากทฤษฎีที่กล่าวมาทั้งหมดสามารถที่จะสรุปได้ว่า ความถี่ที่จะนำมา SAMPLING นั้น จะต้องมีความถี่มากกว่าหรือเท่ากับ 2 เท่า ของความถี่สูงสุดของสัญญาณข้อมูล ดังนั้น ในส่วนนี้จึงจำเป็นสิ่งที่ต้องคำนึงถึงอย่างมากในการการทำโครงงานนี้ว่า SAMPLING แล้วจะมีช่วงเวลาเป็นเท่าใด มีเวลาเหลือพอที่จะให้วงจรอื่นๆ ทำงานได้ทันหรือไม่ด้วย

2.2 The Fast Fourier Transform and Spectral Analysis

FFT

คุณสมบัติของการแสดงผลทางจอภาพนั้นมีอยู่หลายวิธีกับธรรมชาติของระยะเวลาซึ่งจะมีสิ่งรบกวนต่างๆ ปนอยู่ด้วยทำให้เกิดการแปรผันขึ้นๆ ลงๆ อย่างไรก็ตามยังสามารถจะแยกออกจากกันได้ โดยเข้าใจจาก

ฐานของระยะเวลาที่มีอยู่ก่อนในขบวนการนั้นๆ ฟูเรียร์ทรานสฟอร์มได้ถูกพัฒนาให้เป็นวิธีการทางสเปคตรัมทั่วๆ ไปพร้อมกับวิธีการอื่นๆ

สิ่งที่จำเป็นของขวนการนี้ก็คือ การสันณิษฐานของรูปลักษณะที่เกิดขึ้น (ซึ่งจะเกิดขึ้นเป็นอิสระได้บ่อยครั้งในการเปลี่ยนแปลงทางเวลา) และการทรานสฟอร์มของขบวนการนี้จะทำให้เกิดผลของสเปคตรัมร่วมกับการเปลี่ยนแปลงของความถี่อย่างอิสระ

ในที่นี้ความถี่ที่ใช้จะมีองค์ประกอบและจะถูกส่งไปอย่างอิสระเปลี่ยนแปลงไปตามรูปลักษณะของขบวนการสำหรับการจะมีขอบเขตเฉพาะในการส่งเวลาที่อนุกรมกัน (เมื่อเวลาไม่อยู่บนฐานเดียวกัน) การเป็นอิสระของการเปลี่ยนแปลงจะปรากฏเหมือนองค์ประกอบในทรานฟอร์ม คือ ความถี่จะวัดได้ชุดหนึ่งกับเวลา ในส่วนที่ตามกันไป เราสามารถตัดสินใจโดยใช้ Fourier transform และ Discrete Fourier transform (DFT)

The Discrete Fourier Transform (DFT)

เมื่อมีขบวนการของรูปสัญาณที่เป็น aperiodic ฟูเรียนร์ทรานส์ฟอร์มจะมีขอบเขตโดยการใช้อินอิกัลตามรูปแบบคือ

$$X_{(f)} = \frac{1}{\sqrt{2\Pi}} f_{x(t)}^{-} 2\Pi f, dt \qquad (2.8)$$

ขนาดของ $X_{(f)}$ คือ power spectral และ Complex phasor satistics

$$e^{-j2\Pi f,} = \cos(2\Pi f_{t}) - j\sin(2\Pi f_{t}) \qquad (2.9)$$

สำหรับขบวนการของระยะเวลาที่แสดงจะเป็นช่วงเวลา T จะได้ดังนี้

$$x_{(t)} = x_{(t-nT)} \qquad (2.10)$$

แทนสมการ (2.10) ลงในสมการที่ (2.19)

$$e^{-j2\Pi f,} = 1 \qquad (2.11)$$

หรือ จะมีความสำพันธ์ที่เป็นไปได้ในการที่จะลดสมการที่ (2.8)

$$X_{(f)} = \frac{1}{\sqrt{2\Pi}} \sum_{n=-\infty}^{\infty} \int_{-nT/2}^{nT/2} x_{(t)} e^{-j2\Pi f,} dt \qquad (2.12)$$

จะเห็นว่าการคำนวณในสมการที่ (2.12) จะเป็นสมการไม่มีที่สิ้นสุดต้องจากหลังความแน่นอนแต่ละ class ของ function ในทางปฏิบัติ function ของ time domain $x_{(t)}$ ค่าที่เราไม่รู้ในช่วงอื่นๆ หรือค่าจำนวนเล็กน้อยที่หายไปของบางช่วง (ถ้ามีค่ามากเราต้องให้ความสนใจ) เราจะลดทอนโดยต้องพิจารณาใน function นี้โดยใช้ $x_{(t)}$ คูณด้วย $\omega_{(t)}$ ซึ่งเป็นฟังก์ชั่นที่มีน้ำหนัก ซึ่งก็คือ 0 ของแต่ละช่วงของความยาว T ที่จุดศูนย์กลางคือ

$$X_{(f)} = \frac{1}{\sqrt{2\Pi}} \sum_{n=-\infty}^{\infty} \int_{-nT/2}^{nT/2} x_{(t)} e^{-j2\Pi f_{(t)}} dt \qquad (2.13)$$
$$= \frac{1}{\sqrt{2\Pi}} \int_{t_0-t/2}^{nT/2} x_{(t)}\, \omega_{(t)} e^{-j2\Pi f_t}$$

มีหลายตัวอย่างของ fucntion ที่มีแกนน้ำหนักโดยจะมีคุณสมบัติที่เด่นชัดเจนร่วมกับตัวอ้างอิงที่มีผลกระทบของพารมิเตอร์ในการที่จะเกิดขึ้นอีกและลดขนาดของ side lope เราจะใช้ Hanning Weighting (2) ซึ่งได้รูปแบบคือ

$$\omega_{H(t)} = \begin{cases} \frac{1}{2}\frac{1-\cos 2\Pi_{(t_0-T)}}{T'}, & \frac{(t_0-T')}{2} \le t \le \frac{(t_0+T')}{2} \\ 0, elsewhere \end{cases} \qquad (2.14)$$

สมการที่ (2.13) เราจะได้เป็น

$$X_{(f)} = \int_{to-T'/2}^{to-T'/2} x_{h(t+t_0)} e^{-j2\Pi f_{to}} dt \qquad (2.15)$$

เมื่อ $U_{H(t)} = x_{(t)}\omega_{H(t)} / \sqrt{2\Pi}$ ไดผลเป็น

$$X_{(f)} = e^{-j2\Pi f_t} \int_{-T'/2}^{T'/2} x_{h(t+t_0)} e^{-j2\Pi f_t} dt \qquad (2.16)$$

เมื่อจำนวนของ $\ell^{-j2\Pi f_{to}}$ คือเฟสเฟคเตอร์ที่เกิดขึ้นเองสำหรับขนาดหนึ่งๆ เส้นกลางของ X_H ที่ t_0และมีขอบเขต $t = n^{\Delta t}$ เมื่อ Δt คือรูปบางรูปในช่องต่างๆ มีค่าประมาณ จากสมการ 9 โดยที่

$$X_{(f)} = \sum_{n-N_2}^{N_2} X_H(n\Delta t)\, \ell^{-j2\Pi f_n \Delta t} \qquad (2.17)$$

ในที่นี้การเพิ่มทาง phasor จะมีการลดลงมา (เราจะคำนวณขนาดของ $x_{(f)}$ และ phasor นี้จะไม่ตรงประเด็น จำนวน N จะมีขอบเขตโดยที่

Key factor จำเป็นจะต้อง link กับเวลาและความถี่ domain คือมันจะไม่มีความแน่นอนผลที่ได้ก็คือ

$$\Delta f = \frac{1}{N\Delta t} \tag{2.18}$$

ในสมการที่ 11 ข้างซ้ายจะเป็นจำนวนที่ประกอบขึ้นด้วย resolution ใน domain ของความถี่สมการที่ 2.18 จะคล้ายกับ

$$\Delta f \Delta t > 1 \tag{2.19}$$

การทดสอบใน 2 สมการนี้จะพิจารณาจากความถี่ควอนไดซ์ ใช้ $f = m\Delta f$ สมการที่ 2.17 กลายเป็น

$$X_{(m\Delta f)} = \sum_{n=N/2}^{N/2} X_H(n\Delta t)\, \ell^{-j2\Pi mn/N} \tag{2.20}$$

การลดทอนจะต้องขึ้นอยู่กับ Δt และ Δf

$$X_{(m)} = \sum_{n=N/2}^{N/2} X_H(n\Delta t)\, \ell^{-j2\Pi mn/N} \tag{2.21}$$

สมการ (2.21) คือสมการจาก discreat fourier transform (DFT) N เป็นค่าสำหรับ $X_{(m)}$ m = -N/2, -N/2+1, -N/2+2 ,N/2 ซึ่งแต่ละเทอมมีขอบเขตโดยมีผลรวมของ N เพิ่มขึ้น ซึ่งการเพิ่ม phasor จะคำนวนจากสมการ (2.21) ตามต้องการ

$$(N-1)^2 \tag{2.22a}$$

การเพิ่ม (เชิงซ้อน) เราคาดคะเนจากลำดับการที่แยกออกไปของ N/2 โดยที่จำนวนที่ต้องการเพิ่มมีค่าประมาณ

$$\left[\frac{N}{2}\right]^2 2 = \frac{N^2}{2} \tag{2.22b}$$

จำนวนที่ลดของการเพิ่มในการคำนวน ทรานสฟอร์มโดยแฟคเตอร์ 2 ซึ่งวิธีนี้จะเป็นได้ในการทรานสฟอร์มจำนวนเล็กมากของการคูณ ซึ่งเรียกว่าโดยวิธี DFT ซึ่งแบบนี้ทราบจาก Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT)

FFTคือวิธีหรือแบบของการคำนวนในโดเมนของความถี่เปลี่ยนไปเป็นไทม์โดเมนโดยใช้เลขจำนวนเล็กน้อยในการทำซึ่งมากกว่าจำนวนธรรมดาร่วมกับ DFT

เขียนสมการ (2.22b) ใหม่เป็น

$$X_{(m)} = \sum_{n-N/2}^{N/2} X_H(n)\,\omega^{mn} \tag{2.23}$$

ตาม $\omega_n = \ell^{-j2\Pi/N}$ ไม่ขัดเจนกับ lower care ω มีความสัมพันธ์กับ weighting function ตามสมการ (2) ก็คือ ระยะเวลาของ ω และ

$$\omega_N^{(m+m'N)(n+n'N)} = \omega_N^{mn} \tag{2.24}$$

สำหรับจุดประสงค์สำหรับความเข้าใจของโครง FFT มันจะมีประโยชน์อย่างมากในการประยุกต์และการรวมให้เกิดขึ้นในสมการที่ (2.23) เขียนใหม่ได้เป็น

$$X_{(m)} = \sum_{n=0}^{N-1} X_H(n)\,\omega^{mn} \tag{2.25}$$

สมการที่ (2.24) จะเหมือนกับ (2.23) ภายใต้การสมมติ $u_H(n)$ สามารถขยายระยะเวลากับส่วนประกอบ Phaseer facter เราจะให้ความสนใจในขนาด squared ของค่าสำหรับ $x_{(m)}$

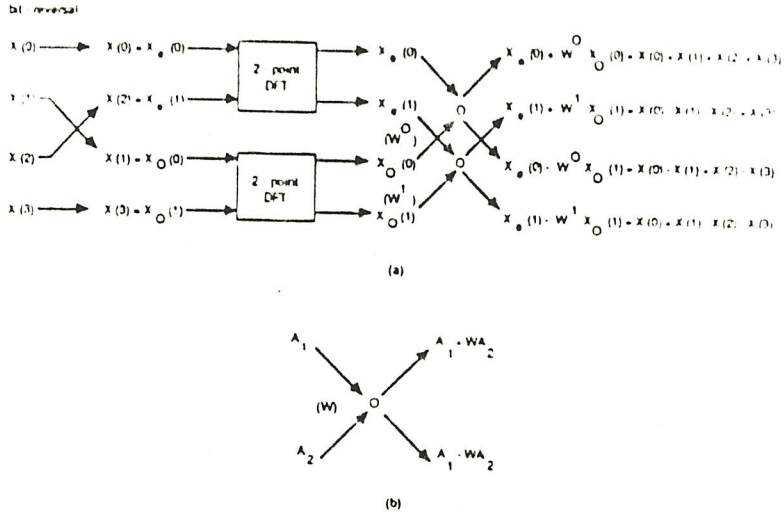
การแปลงผลรวมต่างๆ จะได้ตามสมการ (2.25) ในส่วนประกอบที่เป็นฟังก์ชั่นคู่ และฟังก์ชั่นคี่

$$X_{(m)} = \sum_{n=0}^{N/2-1} X_0(2n)\,\omega_n^{(2n)m} + \omega_N^m \sum_{n=0}^{n/2-1} X_0(2n+1)\,\omega_n^{(2n)m} \tag{2.27a}$$

ที่นี้ $X_0(2n) = \{X(n): n = 0,2,4,..N/2-1\}$

และ $X_0(2n+1) = \{X(n): n = 1,3,5,\ldots,n/2+1\}$

$$X_{(m)} = \left(m - \frac{N}{2}\right) + \omega_N^{m} x_0^{(m-N/2)};\ (\frac{n}{2} \le m \le N-1) \tag{2.27b}$$



Steps needed to evaluate the fast Fourier transform for (a) N = 4 and (b) the multiplying "butterfly."

รูปที่ 2.6 แสดงค่าลำดับขั้นที่เราต้องการในการประเมินค่า สมการ 17a และ 17b สำหรับ N =4 มันสามารถจะแก้ปัญหาได้ดีขึ้น สามารถที่จะลดประสิทธิภาพของการเพิ่มขึ้นได้จริงโดยที่ ตั้งค่าสำหรับจุดสองจุดDFTs และมีการรวมตัวกันอย่างเหมาะสมสำหรับประสิทธิ์นี้ เราเรียกว่า "Butterflies"

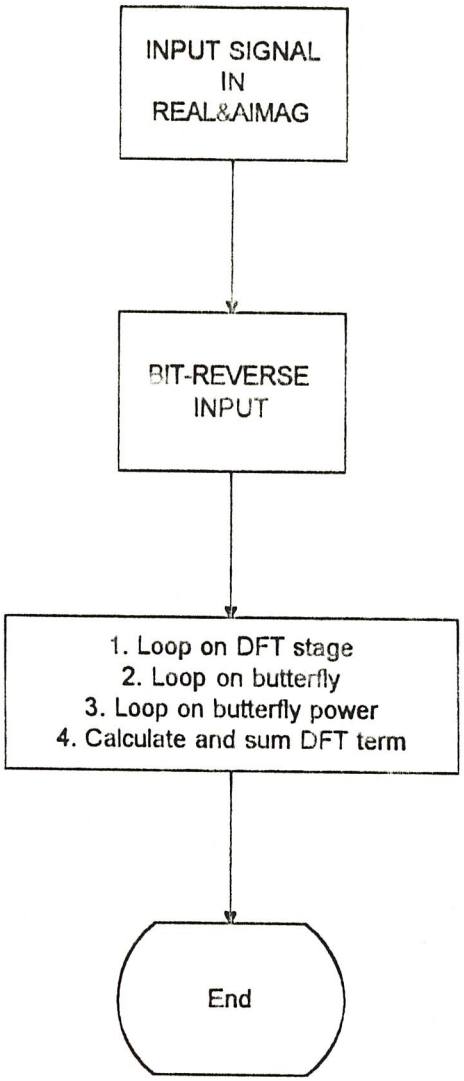
ตามปกติ FFT จะมีประโยชน์สำหรับใช้กลับบิทและเลี่ยงข้อมูลที่ต้องการทางด้านอินพุท ขั้นตอนนี้จะเป็นขั้นที่ง่ายถ้า M บิท เราจะใช้เป็นตัวชี้การนับของ FFT (M = อยู่ภายใต้ FFT) ซึ่งแต่ละอินพุทจะต้องเข้าไปในบิทที่กลับภายใต้ของ ตัว ชี้ธรรมดาภายใต้เอ้าท์พุทสำหรับตัวอย่างถ้า M = 4 (4 บิท) จะชี้จำนวน 10 (ตามแบบของเลข binary)

1010

กลับบิทเป็นตรงข้ามจะได้เป็น

0101

ซึ่งก็คือ 5 ในที่นี้ตำแหน่งที่ชี้ 10 จะต้องเป็นด้านอินพุทของ DFT ที่จะผ่านขั้นตอนไปชี้เป็น 5 เราจะมั่นใจได้ในการกลับบิท แสดงให้เห็นแบบดังรูปที่ 1 โดยเป็น FFTs



Flow chart for a function FFT,which calculate the fast Fourier transform

รูปที่ 2.7 แสดง Flow chart สำหรับฟังก์ชั่น c ที่มีการคำนวนทั่วๆ ไป (มีขอบเขตโดยใช้รูปพารามิเตอร์ m)FFT สำหรับจำนวนเงินเชิงซ้อนทางอินพุท

2.3 POWER SPECTRUM

เมื่อสัญญาณเวลาเป็นช่วง $X_{(n)}$ ถูกแปลงโดยใช้ FFT จะได้ $X_{(m)}$ ซึ่งโดยปกติหมายถึงจะหมายถึง linear spectrum ในการนำไปประยุกต์ใช้งานบ่อยครั้งที่เราน่าสนใจ ขนาด (magnitude) ยกกำลังสอง และเนื่องจากขนาดยกกำลังสอง จะเป็นสัดส่วนกับกำลัง (power) เราจึงใช้คำว่า power spectrum สำหรับฟังค์ชั่นนี้ และกำหนดโดย

$$S_{xx}(m) = \frac{\left|X(m)\right|^2}{N} = \frac{X_{(m)}X_{(m)}}{N} = \frac{X_r^2(m) + X_s^2(m)}{N} \tag{2.28}$$

จาก Parseval theorem จะได้

$$\sum_{n=0}^{n-1} X_{(n)}^2 = \sum_{m=0}^{n-1} S_{xx}(m) \tag{2.29}$$

พิจารณาสัญญาณที่เป็นปริมาณจริง $x_{(t)}$ ที่มีค่าตลอดช่วงเวลาถึงอนันต์ ในกรณีนี้สัญญาณจะมีพลังงานเป็นอนันต์ และกำลังเฉลี่ย (average power)ไม่เป็นศูนย์ ดังนั้นเราสนใจที่จะหาลักษณะที่กำลังกระจายตลอดช่วงความถี่ เรานิยาม power density spectrum (PDS) ของ $x_{(t)}$ ว่า

$$G_{(x)}(\omega) = \frac{1}{2\Pi}\lim_{T\to\infty}\frac{1}{T}\left|X(\omega)\right|^2 \tag{2.30}$$

$X_{(\omega)}$ คือ Fourier transform ของส่วน (segment) ของ $x_{(t)}$ ที่มีความยาว T วินาที ขณะนี้ กำลังเฉลี่ยของ $x_{(t)}$ กำหนดโดย

$$P_{av} = \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} \tag{2.31}$$

PDS ที่เราเพิ่งจะนิยาม จะมีคุณสมบัติ

$$P_{av} = \int_{-\infty}^{\infty} G_x(\omega)d\omega \tag{2.32}$$

ซึ่งหมายถึงพื้นที่ทั้งหมดภายใต้ $G_x(\omega)$ จะเท่ากับกำลังเฉลี่ย ดังนั้น $G_x(\omega)$ จึงมีชื่อว่า power density spectrum เราจะได้ว่ากำลังในช่วงแบนวิดท์ที่กำหนด (ω_1, ω_2) ถูกกำหนดโดย

$$P^{*}(\omega_1,\omega_2)=\int_{-\omega_2}^{-\omega_1}G_{(x)}(\omega)d\omega+\int_{\omega_1}^{\omega_2}G_x(\omega)d\omega=2\int_{\omega_1}^{\omega_2}G_x(\omega)d\omega \qquad (2.33)$$

เนื่องจาก $|X_{(\omega)}|^2$ เป็นฟังก์ชั่นคู่รอบ $\omega=0$ แนวคิดเรื่อง PDS จะมีประโยชน์มากเมื่อเราต้องศึกษาเกี่ยวกับสัญญาณrandom เนื่องจากทำให้เราสามารถศึกษาการกระจายของกำลังเฉลี่ย (distribution of average power) ของสัญญาณ คำว่า spectral estimation จะหมายถึงขบวนการการประมาณค่า SDS ในที่นี้ เราจะกล่าวถึงเพียงการคำนวณเพื่อหาค่า PDS โดยใช้ FFT

พิจารณาซีเควนซ์ที่มีความยาวเป็นอนันต์และเราต้องการหา PDS ในทางปฏิบัติเราทำได้เพียงในส่วนหนึ่งของซีเควนซ์ $X_{(n)}$ และประมาณค่า PDS จริงๆ ของซีเควนซ์ โดยใช้ $X_{(n)}$ ถ้า $G_x(\omega)=G(\ell^{j\omega t})$ เป็น PDS จริง $G_x(\omega)$ จะแทนค่าประมาณของมัน

เราจะคำนวณ $G_x(\omega)$ โดยเริ่มจากแบ่ง $X_{(m)}$ เป็นซีเควนซ์ย่อย $X_l(m)$ ซึ่งโดย $D=M/2$ ขณะนี้

$$X_l(m)=X[m+(l-1)D];1\le l\le K \qquad (2.34)$$

เมื่อ K คือ จำนวนซีเควนซ์ทั้งหมดที่เรานำมาหาค่า PDS
ต่อไปคูณแต่ละ $X_l(m)$ ด้วยซีเควนซ์ window $\omega_{(m)}$ ได้

$$X_l'(m)=X_l(m)\,\omega_{(m)};0\le m\le M-1 \qquad (2.35)$$

สำหรับ $1\le l\le K$ ทำการคำนวณ DFT

เมื่อ $\omega_{(m)}=\ell^{-j2\Pi/M}$ และ $X_l'(n)$ คือ สัมประสิทธิ์ที่ n ของ DFT ของซีเควนซ์ย่อย 1
เรานิยาม periodogram $Q_l(f_n)$

$$Q_l(f_n)=\frac{1}{MU}|X_l'(n)|^2;0\le n\le M-1 \qquad (2.36)$$

เมื่อ $f_n=\frac{n}{MT};0\le n\le M-1$

$$U = \frac{1}{M}\sum_{m=0}^{M-1}\omega_{(m)}^{2} \quad \text{;กำลังเฉลี่ยใน window sequence} \tag{2.37}$$

จะคำนวณ PDS ที่ต้องการโดย

$$\overline{G}(f_n) = \frac{1}{K}\sum_{I=0}^{K} Q_1(f_n) = \frac{1}{KMU}\sum_{I=1}^{K}\left|X'_I(n)\right|^2 ; 0 \le n \le M-1 \tag{2.38}$$

ตัวอย่างในรูปเป็นสัญญาณเวลา เป็นช่วงที่ได้จากการ sample ด้วยอัตรา 256 sps เพื่อให้ได้เควนซ์ 2048 จุด ขณะนี้ซีเควนซ์มูลถูกโพรเซสโดย 50% เหลี่ยมกันแต่ละซีเควนซ์ย่อย ประกอบด้วย 256 จุด (M = 256) และใช้ร่วมกับ hanning window sequence (M - 256) เนื่องจากซีเควนซ์ข้อมูลเป็นปริมาณจริง ดังนั้น เราคำนวณ 256 จุด FFT โดยหาค่าเพียง 129 จุด

$$f_n = \frac{n}{MT} = n; 0 \le n \le 128 \tag{2.39}$$

นั่นคือ จุดของ PDS จะห่างกัน 1Hz (frequency resolution ของ PDS)

2.4 พื้นฐานการแปลงสัญญาณ A/D แบบขนาน (แฟลช)

หลักการอย่างง่ายที่สุดของการแปลงสัญญาณอนาล็อกเป็นดิจิตอลแบบขนานแสดงดังรูปที่ 3 ใช้ตัวเปรียบเทียบ 3 ตัว ต่อในลักษณะ "ขนานกัน" มีตัวต้านทานต่อแบ่งแรงดันจากแรงดันอ้างอิงไว้กำหนดค่าแรงดันต่ำสุดที่ตัวเปรียบเทียบทั้ง 3 ตัว ยังสามารถทำงานได้ แรงดันอ้างอิงนี้อาจจะมีค่าเท่ากับค่าแรงดังสูงสุดของสัญญาณอินพุทที่เป็นอะนาลอกก็ได้

จากตารางที่ 1 จะเห็นว่าตัวเปรียบเทียบแต่ละตัวจะให้เอ้าท์พุทเป็น "1" ก็ต่อเมื่อแรงดันอินพุทมีค่าสูงกว่าแรงดันอ้างอิงของตัวเปรียบเทียบแต่ละตัว ซึ่งมีค่าแตกต่างกัน และถ้าแรงดันอินพุทมีค่าอยู่ในช่วง 3-4 โวลท์ (แรงดันอ้างอิง +4 โวลท์) จะทำให้ตัวเปรียบเทียบทั้ง 3 ตัว ให้เอ้าท์พุทเป็น "1" หมด เอ้าท์พุทจากตัวเปรียบเทียบทั้งหมดส่งเข้าไปที่วงจรเข้ารหัสเพื่อทำให้เป็นสัญญาณดิจิตอลในระบบเลขฐานสองต่อไป

รูปวงจรที่ 3 นี้ ตอบสนองต่อแรงดันอินพุท (อะนาลอก) 4 ระดับแลแต่ละระดับมีความแตกต่าง 1 โวลท์ ดังนั้นความละเอียด (resolution) ของวงจรนี้มีขนาด 2 บิต เราสามารถหาความละเอียดของวงจรได้จากจำนวนของตัวเปรียบเทียบนั่นคือ

จำนวนเปรียบเทียบ = ความละเอียด

เช่น ต้องการความละเอียดขนาด 8 บิท จะต้องใช้ตัวเปรียบเทียบถึง 255 ตัว (แทนค่า n = 8 ในสูตร)

จากลักษณะการต่อตัวเปรียบเทียบให้ขนานกัน เพื่อให้รับสัญญาณอินพุทได้พร้อมๆ กัน เราจึงเรียกวงจรนี้ว่า วงจรแปลง สัญญาณอะนาล็อกเป็นดิจิตอลแบบขนาน (parallel A/D converter) และเนื่องจากมันสามารถตอบสนองต่อสัญญาณอินพุทเป็นอะนาลอก และแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอลได้อย่างรวดเร็วมาก เราจึงเรียกได้อีกอย่างว่า วงจรแปลงสัญญาณอะนาล็อกเป็นดิจิตอลแบบแฟลช (flash A/D converter)

2.5 CA3318 แฟลช A/D CONVERTER

โดยทั่วๆไป เมื่อกล่าวถึง A/Dconverter แล้วจะหมายถึง ตัวแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอล สัญญาณอะนาล็อกก็คือสัญญาณที่มีการเปลี่ยนแปลงอย่างต่อเนื่องกันไปที่รู้จักคุ้นเคยกันดี ได้แก่ สัญญาณรูปซายน์, สัญญาณสามเหลี่ยมและสัญญาณเสียง เป็นต้น

CA3318 เป็นไอซี ทำหน้าที่แปลงสัญญาณอะนาล็อกเป็นดิจิตอลแบบแฟลช ขนาด 8 บิท มีความเร็วในการแปลงสัญญาณสูงมาก มีขนาด 24 ขา ตัวถังเป็นแบบ DIP

สำหรับคำว่า "แฟลช" ที่อยู่หน้า A/D คอนเวอร์เตอร์ เป็นรูปแบบการแปลงสัญญาณอะนาล็อกเป็นดิจิตอลอีกรูปแบบหนึ่งในอีกหลายๆ แบบ ซึ่งแบบแฟลชนี้มีความเร็วในการแปลงสัญญาณสูงกว่าแบบอื่นๆ

แต่หัวใจสำคัญของการแปลงสัญญาณ (ไม่ว่าจะเป็น A/D หรือ D/A ก็ตาม) ก็คือ ค่าความถูกต้องของการแปลงสัญญาณก็ขึ้นอยู่กับความละเอียด (resolution) ของการแปลงสัญญาณและความเร็วในการแปลงสัญญาณ (อย่าลืมว่าการแปลงสัญญาณเป็นเพียงการแปลงสัญญาณที่อยู่ในรูปแบบหนึ่งโดยเนื้อหายังต้องคงเดิมอยู่ มิใช่แปลงรูปแบบไปแล้วเปลี่ยนเนื้อหาไปด้วย)

2.6 โครงสร้างและการทำงานของ CA3318

โครงสร้างภายในและการกระทำของ CA3318 แสดงดัง รูปที่ 2.17 ดังนี้

ชุดสวิตช์อิเล็กทรอนิกส์ (ส่วนที่เห็นเป็นวงกลมมีกากบาทอยู่ภายใน) ทำการสุ่มสัญญาณเข้ามาสู่ชุดตัวเปรียบเทียบจำนวน 256 ชุด ตัวเปรียบเทียบนี้ทำหน้าที่เปรียบเทียบสัญญาณอินพุทที่เป็นอนาล็อกกับแรงดันอ้างอิงของตัวเปรียบเทียบทั้ง 256 ชุด ที่ได้กำหนดไว้แล้ว

ข้อมูลทั้งหมดจากตัวเปรียบเทียบ (เป็น "0" หรือ "1") ส่งเข้า D ฟลิปฟลอปทั้ง 256 ชุด โดยตรง เป็นไปในลักษณะตัวเปรียบเทียบชุดที่ 1 ส่งข้อมูลเข้า D ฟลิปฟลอปชุดที่ 1 คือส่งเข้าชุดใครชุดมัน D ฟลิปฟลอป ทำหน้าที่เป็น shift register ทำงานในโหมดสัญญาณนาฬิกา (ตอบสนองต่อสัญญาณนาฬิกาเฉพาะช่วงขอบขาขึ้นและขอบขาลงพัลซ์เท่านั้น) ทำการแลตช์ (latch) ข้อมูลไว้ชั่วขณะ จนกว่าจะมีข้อมูลใหม่เข้ามาจึงจะเลื่อน (shift) ข้อมูลนั้นส่งเข้าชุดเข้ารหัส (encoder logic array) เพื่อแปลงข้อมูลทั้ง 256 ค่า ออกมาเป็นข้อมูลดิจิตอลขนาด 9 บิท (รวมบิตส่วนเกินด้วย) ส่งต่อไปยังเอ้าท์พุทรีจิสเตอร์ ซึ่งใช้ D ฟลิปฟลอบ ทำหน้าที่นี้อีก เช่นเคยก่อนส่งไปยังตัวขับ 3 สถานะเป็นเอ้าท์พุทต่อไป เอ้าท์พุทนี้สามารถควบคุมได้ด้วย CE_1 และ CE_2

การทำงานทั้งหมดนี้เราสามารถควบคุมได้ที่ขาควบคุมเฟส (ขา 19)

2.7 คุณสมบัติของ CA3318

- คุณสมบัติแบบคร่าวๆ ของ CA3318 มีดังนี้
- ใช้เทคโนโลยี CMOS/SOS
- ใช้เทคนิคการแปลงข้อมูลแบบขนาน
- ให้เอ้าท์พุทขนาด 8 บิท
- ใช้แหล่งจ่ายไฟชุดเดียว 4 โวลท์ ถึง 6.5 โวลท์
- แยกระบบกราวด์ของอะนาลอกกับดิจิตอลออกจากกันโดยเด็ดขาด
- กำลังงานสูญเสีย 200 มิลลิวัตต์
- แรงดันอินพุทอยู่ในช่วง 0 - 6.4 โวลต์
- สัญญาณนาฬิกา 20 เมกกะเฮิรตซ์

2.8 สัญญาณนาฬิกากับเฟส

CA3318 ใช้เทคนิคการแปลงข้อมูลแบบขนานเป็นลำดับ (sequential parallel technique) โดยอาศัยการจัดระดับลอจิกของสัญญาณนาฬิกาไปควบคุมจังหวะในการทำงานของส่วนต่างๆ ให้สอดคล้องกัน ซึ่งจุดประสงค์จริงๆ ก็คือ ความเร็วในการแปลงสัญญาณต้องเป็นแบบ "แฟลช" นั่นเอง

จากรูปแบบแสดงโครงสร้างภายใน (รูปที่ 1) ขาป้อนสัญญาณนาฬิกา (ขา 18) และขาควบคุมเฟส (ขา19) ต่อกับวงจรลอจิกเกตชุดหนึ่ง ซึ่งวงจรนี้ทำหน้าที่จัดสัญญาณนาฬิกาให้แบ่งเป็นเฟส2เฟสคือ $\varnothing_1$ (auto balance) และ $\varnothing_2$ (sample unknown) เฟสทั้ง 2 เฟสนี้จะถูกจัดให้อยู่ในช่วงลอจิก "0" หรือ "1" ของสัญญาณนาฬิกา (ใน 1 คาบเวลา) เราควบคุมได้โดยใช้ขาควบคุมเฟส

จากการจัด $\varnothing_1$ และ $\varnothing_2$ ให้อยู่คนละช่วงของสัญญาณนาฬิกาด้วยขาควบคุมเฟสนี้ ทำให้เราสามารถควบคุมความเร็วในการแปลงสัญญาณ (ข้อมูล) ของ CA3318 ให้เปลี่ยนแปลงไปตามสัญญาณนาฬิกาได้

2.9 การควบคุม CA3318

เราสามารถใช้ขาควบคุมเฟส (ขา 19) ควบคุมความเร็วการแปลงสัญญาณของ CA3318 ได้ 2 วิธีคือ

วิธีแรก โดยการป้อนลอจิก "0"เข้าที่ขาควบคุมเฟส แสดงดังรูปที่ 2 (ก) $\varnothing_1$ จะถูกจัดอยู่ในลอจิก "1" และ $\varnothing_2$ ถูกจัดอยู่ให้อยู่ในลอจิก "0" ของสัญญาณนาฬิกา ข้อมูลจากตัวเปรียบเทียบ (อะนาลอก) จะถูกแลตช์ไว้ที่ขอบขาขึ้นของพัลซ์ $\varnothing_1$ และเมื่อถึงช่วงขอบขาลงของพัลซ์ $\varnothing_1$ แล้ว ก็จะทำการเลื่อนข้อมูลนั้นส่งเข้าเอ้าท์พุทรีจิสเตอร์ต่อไป ซึ่งวิธีนี้เป็นการแปลงข้อมูลเสร็จสิ้นภายในครึ่งคาบเวลาของสัญญาณนาฬิกาเท่านั้น

วิธีที่สอง โดยการป้อนลอจิก "1" เข้าที่ขาควบคุมเฟสเช่นกัน แสดงดังรูปที่ 2(ข) $\varnothing_1$ ถูกจัดให้อยู่ในลอจิก "0" และ $\varnothing_2$ ถูกจัดให้อยู่ในลอจิก "1" ของสัญญาณนาฬิกาเมื่อถึงช่วงขอบขาลงของพัลซ์ $\varnothing_2$ ข้อมูลจากตัวเปรียบเทียบ (อะนาล็อก) ถูกแลตช์ไว้จนกว่าจะถึงช่วงขอบขาขึ้นของพัลซ์ $\varnothing_2$ ถูกต่อมาจึงจะทำการเลื่อนข้อมูลส่งเข้า

เอ้าท์พุทรีจิสเตอร์วิธีนี้จะแปลงสัญญาณเสร็จสิ้นภายใน 1 คาบเวลาของสัญญาณนาฬิกา

จะเห็นว่า วิธีแรกใช้เวลาในการแปลงสัญญาณน้อยกว่าวิธีที่สองและสำหรับงานที่ต้องการความเร็วในการแปลงสัญญาณควรเลือกควบคุม CA3318 ด้วยวิธีแรกจะเหมาะสมกว่า

2.10 การประยุกต์ใช้งาน

ส่วนใหญ่แล้ว CA3318 นำไปใช้ในระบบที่ต้องการประมวลผลด้วยความเร็วสูงมากๆ อย่างเช่น

- การวิเคราะห์สัญญาณเรดาห์
- การวิเคราะห์สัญญาณทรานเซ้ยนต์
- การวิเคราะห์อาการเคลื่อนของวัตถุ
- ใช้ในอุปกรณ์แปลงสัญญาณภาพระบบดิจิตอลในเครื่องรับโทรทัศน์

CA3318 ออกแบบภาคเอาท์พุท โดยใช้ตัวขับ 3 สถานะ (3-state deivers) เพื่อเชื่อมโยงกับไมโครโปรเซสเซอร์ ขนาด 8 บิท โดยเฉพาะ

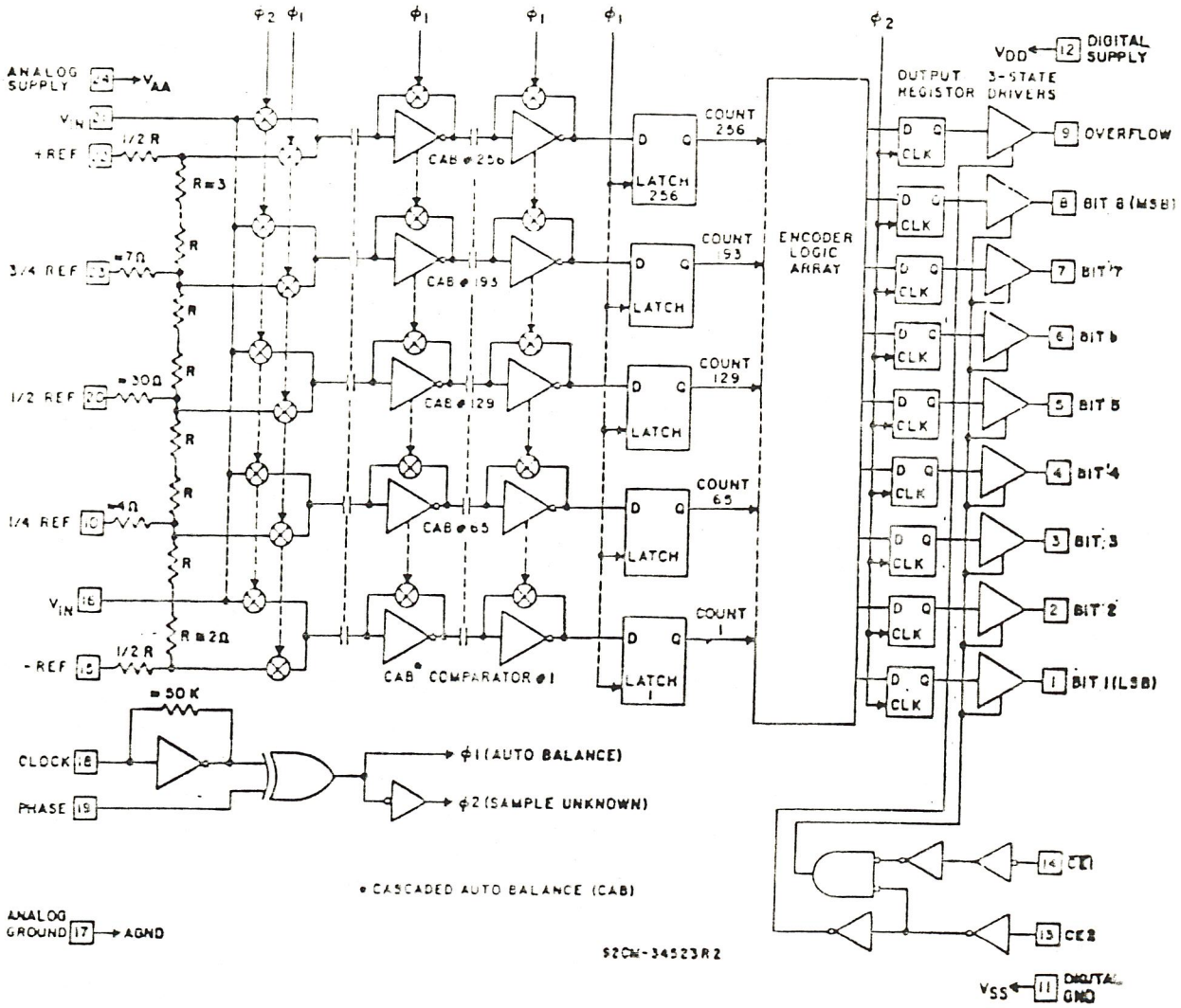
แต่โดยทั่วไปแล้ว ขั้นตอนหลักของการเชื่อมโยงตัวแปลงสัญญาณอะนาล็อกเป็นดิจิตอลเข้ากับไมโครโปรเซสเซอร์ มี 3 ขั้นตอนใหญ่ๆ คือ

ขั้นแรก ป้อนคำสั่ง (command) สู่ไมโครโปรเซสเซอร์ที่จะให้ตัวแปลงสัญญาณอะนาล็อก เป็นดิจิตอลเริ่มต้นการแปลงสัญญาณ

ขั้นสอง ที่จะต้องมีการตรวจสอบสถานะข้อมูลอินพุท (data ready) จนกระทั่งการแปลงสัญญาณเสร็จสิ้นสมบูรณ์

และขั้นตอนที่ 3 ทำการอ่านข้อมูลที่เป็นดิจิตอลเข้าสู่ตัวไมโครโปรเซสเซอร์เอง

นอกเหนือจาก CA3318 แล้วยังมีไอซีอีกเบอร์ซึ่งอยู่ในอนุกรมเดียวกันทำหน้าที่อย่างเดียวกันคือเบอร์ CA3318C ปรับปรุงมาจาก C3318 ให้มีความเร็วในการสุ่มสัญญาณสูงขึ้นจาก 15 MSPS (Million Sampling Per Second) สัญญาณนาฬิกา 20 เมกกะเฮิรตซ์ ที่ 5 โวลท์ (CA3318) เป็น 20 MSPS สัญญาณนาฬิกา 15 เมกะเฮิรตซ์ ที่ 5 โวลท์ แต่กำลังสูญเสียลดลงเหลือเพียง 150 มิลลิวัตต์ ส่วนคุณสมบัติอย่างอื่นและวงจรภายในเหมือนกับ CA 3318 ทุกประการ



รูปที่ 2.7 แสดงการต่อวงจรภายในของ CA3318

2.11 ช่องเสียบสัญญาณบนเมนบอร์ด (SLOT)

ในปัจจุบันคอมพิวเตอร์มีราคาถูกลงมาก มีการนำไปใช้งานอย่างกว้างขวาง ไม่ว่าจะเป็นในด้านที่เกี่ยวกับเอกสารหรืองานทั่วไป เช่น การทำเสียงเอฟเฟคใช้ประกอบภาพยนต์หรือในการทำในรูปโฆษณา เป็นต้น การใช้งานด้านต่างๆ เหล่านี้ มักจะมีอุปกรณ์การทำงาน ซึ่งเป็นตัวเชื่อมต่อระหว่างผู้ใช้งานและคอมพิวเตอร์ ทำให้การใช้งานเป็นไปอย่างง่ายดายขึ้นเช่นจอยสติก เม้าส์ ปากกาเขียนจอ เป็นต้น อุปกรณ์เหล่านี้จะติดต่อกับคอมพิวเตอร์โดยมีรูปแบบการติดต่อที่เป็นมาตรฐานที่นิยมใช้กันอยู่ทั่วไป คือ มาตราฐานการสื่อสารแบบอนุกรม (RS-232C) และมาตราฐานการสื่อสารแบบขนาน (GPIB) ที่จำเป็นต้องมีมาตราฐานเดียวกันก็เพื่อทำให้อุปกรณ์นั้น สามารถใช้งานได้อย่างกว้างขวาง ไม่เฉพาะเจาะจงใช้กับเครื่องคอมพิวเตอร์ยี่ห้อใด ดังนั้น เครื่องคอมพิวเตอร์ไม่ว่าจะเป็นเครื่องรุ่นใดก็ตาม สามารถจะใช้อุปกรณ์เหล่านี้ได้แต่อย่างไรก็ตามมาตราฐานเหล่านี้ก็ยังมีข้อจำกัดอยู่บ้าง ดังนั้นในที่นี้จะขออธิบายการเชื่อมต่อ (การอินเตอร์เฟส) กับ IBM PC ซึ่งการเชื่อมต่ออุปกรณ์โดยตรงกับระบบบัสของคอมพิวเตอร์ เป็นการเพิ่มขีดความสามารถในการติดต่อและเพิ่มประสิทธิภาพในการทำงานของอุปกรณ์ I/O ที่ต่ออยู่กับระบบได้ดีขึ้น

2.12 AT SLOT & XT SLOT

เนื่องจาก AT ใช้ CPU เบอร์ 80286 ซึ่งเป็น CPU ขนาด 16 บิทแท้ คือ มีการประมวลผลภายในเป็นแบบ 16 บิท และมีดาต้าบัสขนาด 16 บิท ในขณะที่ XT ใช้ CPU เบอร์ 8088 ซึ่งเป็น 16 บิทเทียม คือ มีการประมวลผลภายในเป็นแบบ 16 บิท แต่มีดาต้าบัสเพียง 8 บิท ดังนั้นการจัดการเกี่ยวกับข้อมูลขนาด 16 บิท จึงต้องทำสองครั้งๆ ละ 8 บิท จะเห็นว่าระบบบัสของ XT เป็นซับเซตของระบบบัสของ AT

บนเมนบอร์ดของ AT จะมี slot อยู่ 2 ชนิด คือ สล็อตสั้นและสล็อตยาวซึ่งสล็อตยาวจะมีจำนวนขาสัญญาณ และตำแหน่งของขาสัญญาณบนสล็อตเหมือนกับ XT ส่วนขาสัญญาณที่อยู่สล็อตสั้นที่เพิ่มขึ้นจะเป็นสัญญาณที่มีแต่เฉพาะบน AT เท่านั้น ประกอบด้วยข้อมูลคร่ึงบน 8 บิท แอดเดรสที่ขึ้นมาอีก 4 บิท และขาสัญญาณควบคุมที่เพิ่มขึ้นจาก XT

ขนาดและรูปร่างของการ์ดที่เสียบลงบนสล็อตของ XT จะคล้ายกับของ AT และ เนื่องจากสัญญาณบนสล็อตจะมีตำแหน่งตรงกันด้วย ดังนั้นการ์ดที่ใช้บน XT ทุกการ์ดสามารถนำมาใช้ได้บน AT โดยเสียบไว้บนสล็อตยาว อย่างไรก็ตามจะต้องคำนึงอัตราการรับส่งข้อมูลของการ์ดนั้นด้วย เพราะความเร็วในการทำงานของ AT จะเร็วกว่า XT ดังนั้นเมื่อการ์ดของ XT มาใช้กับ AT อาจจะช้าลงก็ได้ แต่ถ้าการทำงานของการ์ดนั้นไม่เกี่ยวข้องกับหน่วยความจำ หรือการทำ DMA เราก็ไม่จำเป็นต้องสนใจกับอัตราการรับส่งข้อมูลของมัน

2.13 รายละเอียดของสัญญาณต่างๆ บนสล็อต

(I), (O) และ (I), (O) หมายถึง ทิศทางของขาสัญญาณเมื่อเทียบกับเมนบอร์ด
โดยที่

- (I) หมายถึง ขาสัญญาณอินพุท
- (O) หมายถึง ขาสัญญาณเอาท์พุท
- (I,O) หมายถึง ขาสัญญาณที่เป็นได้ทั้งอินพุทและเอาท์พุท
- (*I, O) หมายถึง ในช่วงการทำงานปกติจะเป็นขาสัญญาณเอาท์พุทแต่จะเป็นอินพุทในช่วงที่เกิดขบวนการ DMA

สำหรับขาสัญญาณที่มีเครื่องหมายลบนำหน้านั้น จะหมายถึงขาสัญญาณที่แอคทีฟที่ลอจิก "0" และขาสัญญาณที่ไม่มีหรือมีเครื่องหมายบวกนำหน้านั้นจะหมายถึงขาสัญญาณที่แอคทีฟที่ลอจิก "1" สัญญาณที่ต่ออยู่บนสล็อตนี้สามารถขับไอซี TTL ชนิดโลว์เพาเวอร์ ได้สองตัว โดยไม่ทำให้เกิดการโหลดหรือการเพี้ยนของสัญญาณต่างๆ บนสล็อตของ XT และ AT สามารถแบ่งออกเป็นกลุ่มๆ ได้ดังนี้

เพาเวอร์ซัพพลาย

- GROUND ขาสัญญาณนี้ต่ออยู่กับกราว์ดของระบบเรคกูเลเตอร์
- + 5 V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรคกูเลเตอร์ + 5 V
- - 5 V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรคกูเลเตอร์ - 5 V
- +12V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรคกูเลเตอร์ + 12 V
- -12V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรคกูเลเตอร์ - 12 V

แอดเดรสบัสและสัญญาณต่างๆ ที่เกี่ยวข้อง

- SAO-SA19 เป็นแอดเดรสบิทที่ 0 ถึง 19 โดยที่ SAO มีนัยสำคัญต่ำที่สุดขาสัญญาณนี้จะแอคทีฟ เมื่อขาสัญญาณ BALE มีสถานะเป็น "1" และถูกแลทช์ไว้ตอนขอบขาลงของสัญญาณ BALE แอคเดรสทั้ง 20 บิท นี้สามารถอ้างหน่วยความจำได้ถึง 1 เมกกะไบท์XT และสำหรับ AT เมื่อใช้ร่วมกับ LA17-LA23 จะอ้างได้ถึง 16 เมกกะไบท์
- LA17-LA23 (เฉพาะรุ่น AT ขาสัญญาณนี้จะแอคทีฟ
- (*I, O) เมื่อขาสัญญาณ BALE มีสถานะเป็นลอจิก "1" แต่จะไม่มีการแลทช์ไว้ตอนขอบขาลงของสัญญาณ BALE ดังนั้นถ้าอุปกรณ์ I/O ไม่มีการแอคเดรสเกิน 1 เมกกะไบท์ ขาสัญญาณนี้ก็ไม่จำเป็นต้องใช้ แต่ถ้ามี การอ้างแอคเดรสเกิน อุปกรณ์ I/O จะต้องทำการแลทช์สัญญาณนี้ โดยใช้ขอบขาลงของสัญญาณ BALE ร่วมกับขาสัญญาณ -MEMW และ -MEMR
- AEN (Address Enable) ขาสัญญาณนี้จะแอคทีฟ
- (O) เมื่อตัวควบคุม DMA ได้ทำการควบคุมบัสต่างๆ ของระบบแล้วดังนั้น การอ้างพอร์ทของอุปกรณ์ I/O จะต้องใช้สัญญาณนี้ในการตีโดดด้วยเพื่อที่จะไม่ทำให้เกิดการติดต่อระหว่างระบบกับอุปกรณ์ I/O ตัวอื่น ยกเว้นตัวที่ทำขบวนการ DMA อยู่

- BALE (Address Latch Enable) ขาสัญญาณนี้ใช้ใน
- (O) การแสดงการเริ่มต้นของขบวนการต่างๆ ที่มีการติดต่อกับหน่วยความจำ โดยจะแอคทีฟเมื่อมีค่าแอดเดรสที่ CPU ต้องการติดต่อด้วยอยู่บนแอดเดรสเรียบร้อยแล้วตามปกติขอบขาลงของสัญญาณนี้ จะทำให้เกิดการแลทช์สัญญาณ SAO-SA19 และถ้ามีการแอดเดรสเกิน 1 เมกกะไบท์ใน AT จะใช้ขอบขาลงของสัญญาณนี้ในการแลทช์สัญญาณ LA17-LA23 ด้วยเช่นกันแต่สำหรับในขบวนการ DMA สัญญาณนี้จะมีค่าเป็น "1" ตลอด
- SBHE (เฉพาะรุ่น AT) (Bus High Enable) เป็นขา
- (*I/O) สัญญาณที่ใช้แสดงว่ามีการรับส่งข้อมูลในบิตที่ SDB-SD15

ดาต้าบัส

- SDO-SD7 สำหรับรุ่น AT จะมี SDO-SD15 เพิ่มขึ้นมา
- (I/O) ด้วยคือ ดาต้าบัส 0 ถึง 7 สำหรับรุ่น XT และสำหรับรุ่น AT คือดาต้าบิท 0 ถึง 15 โดยที่ SDO มีนัยสำคัญต่ำที่สุด สำหรับ AT ถ้ามีการติดต่อกับบิทที่ SD8-SD15 สามารถตรวจสอบได้จากขาสัญญาณ SBHE

สัญญาณอินเตอร์รัพต์

- IRQ2-IRQ7 (Interrupt Request) (สำหรับรุ่น AT จะเป็น IRQ3-7, 9-12, 14-15)
- (I) เป็นค่าสัญญาณอินเตอร์รัพต์ CPU สำหรับ AT ลำดับความสำคัญของ

สัญญาณ IRQ เป็นดังนี้ คือ 9, 10, 11, 12, 14, 15, 3, 4, 5, 6 และ 7 โดย IRQ9 มีลำดับความสำคัญมากที่สุดและ IRQ7 มีความสำคัญน้อยที่สุด สำหรับ XT IRQ2 จะมีลำดับความสำคัญมากที่สุดรองๆ ลงไป คือ IRQ3, 4, 5, 6, 7

โดยปกติสัญญาณนี้จะมีสถานะเป็น "0" เสมอถ้าต้องการอินเตอร์รัพต์ CPU ให้ส่งพัลส์ที่เป็น LOGIC "1" ให้กับมันโดยไม่จำเป็นต้องคำนึงถึงคาบเวลาของพัลส์ทั้งนี้เพราะระบบของ IBM ต้องอินเตอร์รัพต์คอลโทรลเลอร์ (8259 Interrupt Controller) จะถูกโปรแกรมให้ทำการตรวจสอบสัญญาณอินเตอร์รัพส์โดยใช้ขอบล่างของสัญญาณนี้

- -I/O CH CK (I/O Channel Check) เป็นค่าสัญญาณที่บอกถึงความผิดพลาดในการรับส่งข้อมูลซึ่งตรวจสอบจากพาริตี้บิต ถ้า พาริตี้บิตที่อ่านจากหน่วยความจำกับพาริตี้ที่สร้างขึ้นจากขบวนการรับส่งข้อมูลมีค่าไม่เท่ากัน แสดงว่าเกิดการผิดพลาดในการรับส่งข้อมูลสัญญาณนี้จะทำให้เกิดการอินเตอร์รัพส์ CPU แบบ NMI เพื่อบอกให้ CPU ทราบว่าเกิด Parity Error ขึ้น CPU จะแสดงข้อความบอกความผิดพลาดขึ้นและหยุดการทำงาน (Halt) เพื่อให้ผู้ใช้ตรวจสอบหาสาเหตุของการผิดพลาด

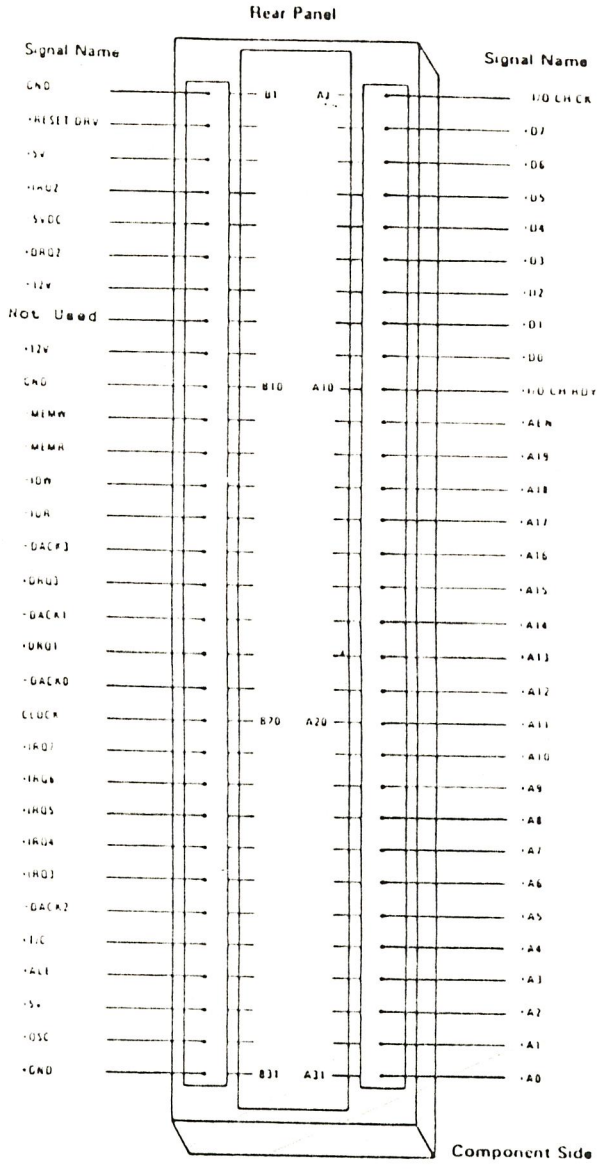
2.14 สัญญาณควบคุมต่างๆ

- MEMR (*I/O) (Memory Read) (สำหรับรุ่น AT คือ ขาสัญญาณ-SMEMR ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำส่งข้อมูลออกมาที่ ดาต้าบัส แต่สำหรับ AT สัญญาณ-SMEMR จะแอคทีฟเมื่อเกิดการอ่านข้อมูล ที่อยู่ในหน่วยความจำ 1 เมกกะไบท์แรกเท่านั้น

- MEMR (O) (เฉพาะรุ่น AT) ขาสัญญาณนี้จะไม่เหมือนกับ MEMR ใน XT มันจะเอคทีฟในทุกๆ ขบวนการอ่านที่เกิดขึ้นไม่ว่าจะอยู่ในหน่วยความจำ 1 เมกกะไบท์แรกหรือไม่
- MEMW (*I/O) (Memory Write) (สำหรับรุ่น AT คือขาสัญญาณ -SMEMW ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำเก็บข้อมูลจากดาต้าบัส สำหรับ AT ขาสัญญาณนี้จะแอคทีฟเมื่อเกิดการเก็บข้อมูลจากหน่วยความจำที่อยู่ใน 1 เมกกะไบท์แรกเท่านั้น
- MEMW(O) (เฉพาะรุ่นAT) ขาสัญญาณนี้ไม่ใช่ขาสัญญาณเดียวกับขา MEMW ใน XT โดยขาสัญญาณนี้จะแอคทีฟในทุกๆ ขบวนการเก็บข้อมูลที่เกิดขึ้น ไม่ว่าจะอยู่ในช่วง 1 เมกกะไบท์แรกหรือไม่
- -IOR (*I/O) (I/O READ) เป็นสัญญาณที่บอกให้อุปกรณ์ภายนอกที่ต่ออยู่ทำการส่งข้อมูลลงมาที่ดาต้าบัส
- RESET DRV (O) (Reset Driver) เป็นขาสัญญาณที่แอคทีฟตอนช่วงที่เราเริ่มจ่ายไฟให้กับระบบเพื่อใช้ในการรีเซ็ท CPU และอุปกรณ์ต่างๆ ในระบบคอมพิวเตอร์ รวมทั้งอุปกรณ์ I/O ที่ต่อยู่ด้วย
- - MEM CS16 (I) (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่บอกให้ระบบทราบว่าต้องการรับส่งข้อมูลกับหน่วยความจำทีละ 16 บิท ถ้าไม่ป้อนสัญญาณที่ขานี้ การรับส่งข้อมูลจะเหมือนกับ XT คือ ทำการรับส่งข้อมูลทีละ 8 บิท สองครั้งเพื่อให้ได้ข้อมูล ขนาด 16 บิท
- - I/O CS16 (I) (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่บอกให้ระบบทราบว่าต้องการรับส่งข้อมูลกับอุปกรณ์ I/O ทีละ 16 บิท ถ้าไม่ป้อนสัญญาณนี้การรับส่งข้อมูลจะทำเหมือนกับ XT คือทีละ 8 บิท สองครั้ง

สัญญาณนาฬิกา

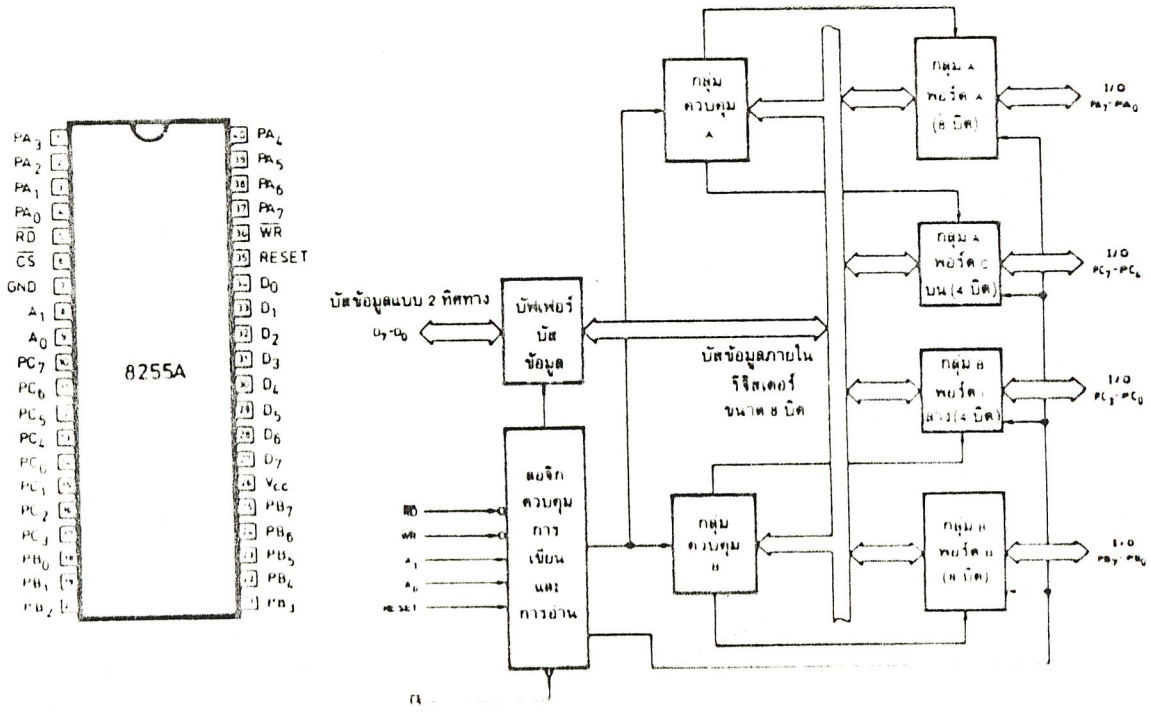
- CLK (O) (System Clock) สำหรับ XT ขาสัญญาณนี้จะมีความถี่ประมาณ 4.47 MH_z หรือมากกว่าสำหรับรุ่นใหม่ๆ และสำหรับ AT ขาสัญญาณนี้จะมีความถี่ประมาณ 6 MH_z หรือในรุ่นใหม่อาจมีความถี่สูงถึง 15 MH_z
- OSC (O) (Oscillater) เป็นขาสัญญาณที่มีความถี่สูง คือ 14.318 MH_z ความถี่ของสัญญาณนี้จะคงที่เสมอ และไม่ซิงโครนัสกับสัญญาณอื่นๆ ในระบบ ดังนั้นจึง ไม่ควรนำสัญญาณนี้ไปใช้เป็นสัญญาณ CLOCK ของอุปกรณ์ I/O ที่ต่ออยู่กับระบบ



รูปที่ 2.8 แสดงตำแหน่งของสัญญาณต่างๆ บน SLOT

2.15 8255A PROGRAMABLE PERIPHERAL INTERFACE

บล็อกกลุ่มแรกที่เราจะพูดถึงนี้ ได้แก่บล็อกจำนวน 4 บล็อกที่อยู่ทางด้านซ้ายของรูปข้าล่างนี้ ซึ่งจะเป็นส่วนเชื่อมต่อกับอุปกรณ์ภายนอกอื่นๆ โดยมีสาย PAO-PA7,PBO-PB7 และ PCO-PC7 เป็นทางผ่านของข้อมูลระหว่างอุปกรณ์ภายนอกกับ 8255 สายสัญญาณเหล่านี้จะถูกแบ่งออกเป็น3I/O Prot ได้แก่ Port A (PA), Port B (PB),Port C (PC) ทั้งสามพอร์ทนี้สามารถเป็นได้ทั้งอินพุทและเอาท์พุทพอร์ท



รูปที่ 2.9 แสดงวงจรภายในและขาของ 8255

2.16 รายละเอียดการจัดเรียงของ 8255

ในส่วนนี้เราจะพิจารณาหน้าที่ของ8255ซึ่งข้อมูลเหล่านี้จะมีประโยชน์ในการเชื่อมต่อเข้ากับระบบบัสของ CPU สำหรับการจัดขาแสดงดังรูปข้างบน รายละเอียดของแต่ละขามีดังนี้ DO-D7: เป็นสายข้อมูลอินพุท/เอาท์ พุทแบบสองทิศทาง จะเป็นทางผ่านของข้อมูลระหว่างพอร์ทต่างๆของ8255กับบัสข้อมูลของ CPU CS(ChipSelect Input): เมื่อขานี้มีสถานะเป็นลอจิก "0" CPU สามารถที่จะอ่านหรือเขียนข้อมูลกับ 8255

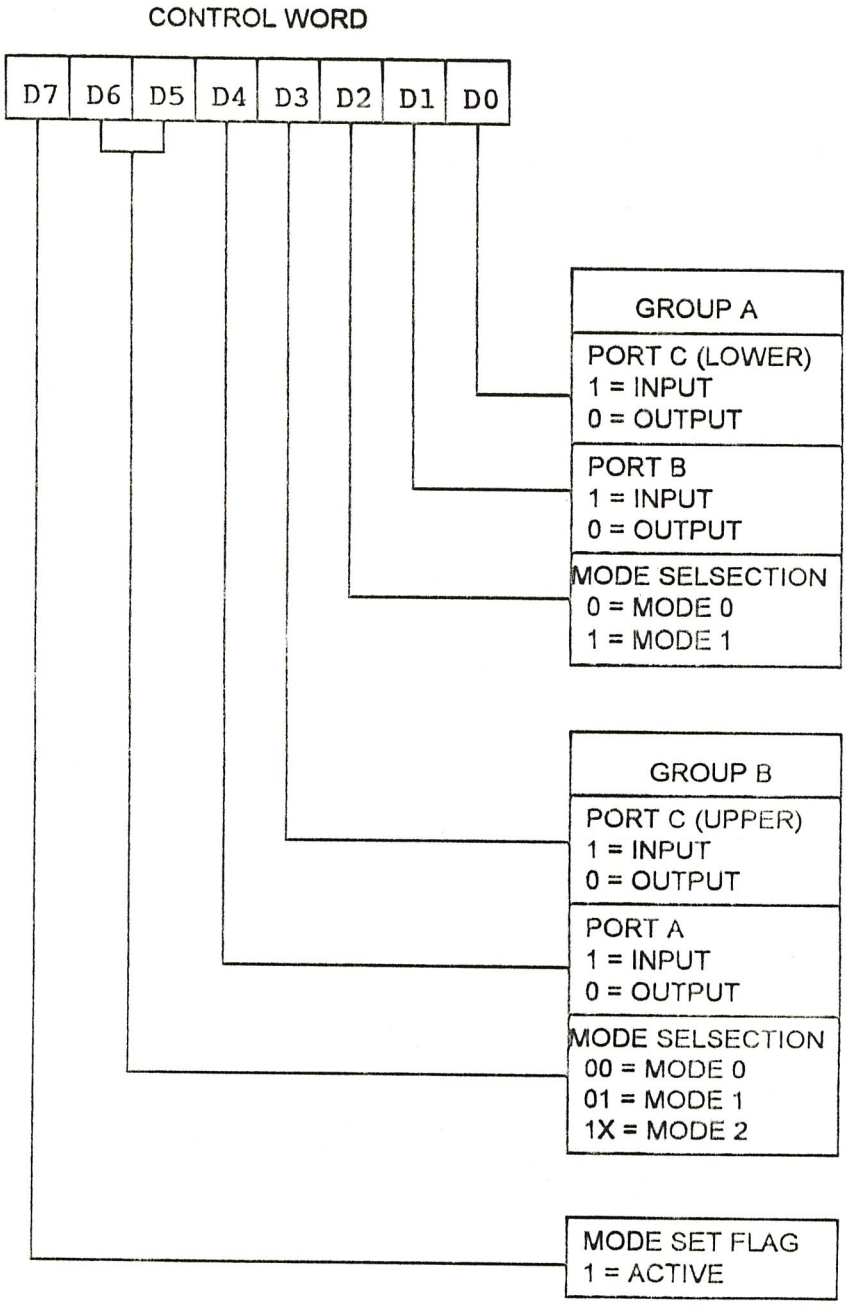
- RD (READ INPUT): เมื่อขานี้มีสถานะเป็น "0" และสัญญาณ CS มีลอจิกเป็น"0" ข้อมูลจาก 8255 จะปรากฏสู่ระบบระบบบัสข้อมูล CPU ก็จะสามารถอ่านข้อมูลออกไปได้ (ในการตั้งชื่อของขาสัญญาณนี้จะถือเอา CPU เป็นหลัก)
- WR (Write Unit): เมื่อขานี้มีสถานะเป็นลอจิก "0" และขาสัญญาณ CS มีลอจิกเป็น "0" ข้อมูลจากระบบบัสข้อมูลจะถูกเขียนเข้าไปใน 8255 ได้
- AO,A1 (Address Unit): จะเป็นตัวกำหนดการเลือกใช้ Reg. ภายในของ 8255 ซึ่งจะกล่าวในภายหลัง
- RESET: เมื่อขานี้มีสถานะเป็น "1" 8255 จะอยู่ในสถานะ RESET ทุกๆ พอร์ทของ 8255 จะถูกเซ็ทให้อยู่ในโหมด อินพุท
- PAO-PA7, PBO-PB7: ขาสัญญาณเหล่านี้จะถูกใช้เป็นพอร์ท I/O ขนาด 8 บิทใช้ต่อเข้ากับอุปกรณ์ภายนอกอื่นๆ
- PCO-PC7: ขาสัญญาณนี้ถูกใช้เป็นพอร์ท I/O ขนาด 8 บิท เช่นเดียวกับ PAO-PA7 และ PBO-PB7 และกลุ่มที่สองใช้ควบคุม PAO-PA7

2.17 8255 READ AND WRITE REGISTER

ขณะนี้เราได้ทำการต่อ 8255 เข้ากับระบบ CPU แล้ว ต่อไปเราจะศึกษาการใช้โปรแกรม 8255 เพื่อที่จะให้ทำงานตามที่เราต้องการได้ โดยจะเริ่มต้นพิจารณาที่ Reg. ภายใน 4 ตัว ของ 8255 สำหรับวงจรถอดรหัสของเรานั้นตำแหน่งของ Reg. จะอยู่ที่แอดเดรส 0500H, 0501H, 0502H, 0503H ซึ่งรายละเอียดของ Reg. มีดังนี้

DEVICE PIN				REGISTER NAME
$\overline{RD}$	$\overline{WR}$	A_1	A_2	
1	0	0	0	wirte port A data
0	1	0	0	read port A data
1	0	0	1	write port B data
0	1	0	1	read port B data
1	0	1	0	write port C data
0	1	1	0	read port C data
1	0	1	1	write control data
0	1	1	1	tlliegal read register

2.18 การโปรแกรม 8255 ให้อยู่ใน Mode ต่าง



รูปที่ 2.10 แสดงลักษณะการเขียน CONTROL WORD

2.19 การทำงานในโหมด 0 (Basic Register I/O)

ในวงจรที่เราจะใช้งานนั้นเรามีข้อกำหนดว่า

- ให้ PORT A เป็นอินพุท อยู่ที่ตำแหน่ง 280H
- ให้ PORT B เป็นเอาท์พุท อยู่ที่ตำแหน่ง 281H
- ให้ PORT C เป็นอินพุท อยู่ที่ตำแหน่ง 282H
- ให้ PORT ควบคุม อยู่ที่ตำแหน่ง 283H

ในการเซ็ท 8255 ให้อยู่ในโหมด 0 นั้น เราจะต้องส่งคำสั่งให้แก่ Reg. ควบก่อน คำสั่งควบคุมนี้ จะกำหนดลักษณะการทำงานแต่ละ PORT ของ 8255 เช่น ในวงจรที่เราออกแบบขึ้นนั้น คำสั่งที่จะสั่งให้ 8255 ทำงานในโหมด 0 คือ

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

รูปที่ 2.11 แสดงตารางการทำงานของละบิท

จากรูป 2.11 เราจะเห็นว่า บิท D7 คือ เป็นตัวบอกให้รู้ว่านี้คือควบคุมคำสั่งควบคุมหรือที่เรียกว่า CONTROL WORD บิท D6-D7 จะกำหนดโหมดการทำงานของ PORT A ถ้าเป็น 0 0 แสดงว่าทำงานโหมด 0

- บิท D4 = 0 จะกำหนดให้ PORT A เป็น PORT OUTPUT
- บิท D3 = 0 จะ SET PORT C ให้ 4 บิท บนเป็น PORT OUTPUT
- บิท D2 = 0 จะ SET MODE ของ PORT B ให้อยู่ในโหมด 0
- บิท D1 = 0 จะ SET PORT B เป็น PORT เอาท์พุท
- บิท D0 = 0 จะ SET PORT C ให้ 4 บิท ล่างเป็นเอาท์พุท

ในวงจรของเรานี้จะใช้ CONTROL WORD เท่ากับ 91H หรือเท่ากับ 1001 0001 ซึ่งจะเซ็ทให้ PORT A เป็นอินพุท PORT B เป็นเอาท์พุท PORT C ล่างเป็นอินพุท PROT C บนเป็นเอ้าท์พุท และทำงานในโหมด 0

บทที่ 3

หลักการออกแบบวงจร (HARD WARE)

3.1 หลักการของ BLOCK DIAGRAM

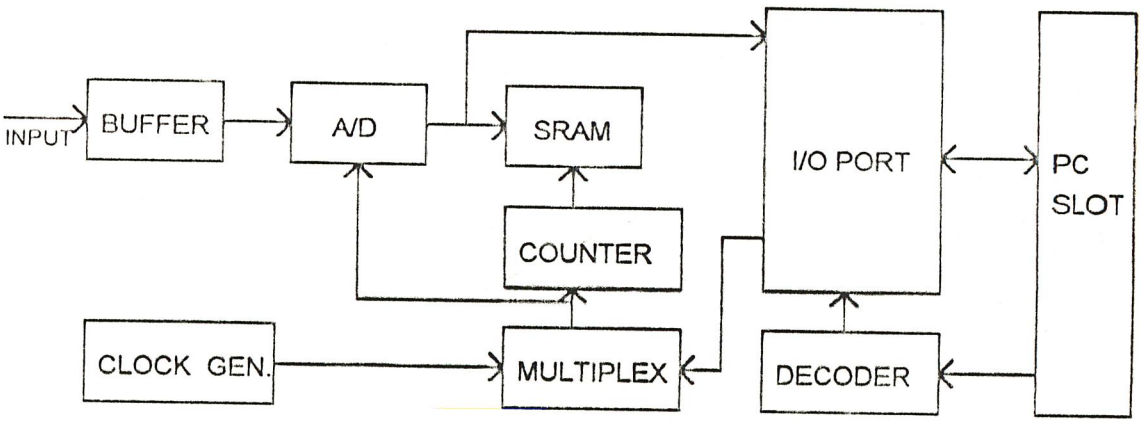
สัญญาณอินพุทที่เป็น ANALOG จะป้อนผ่าน BUFFER เพื่อป้องกัน A/D เสียหายในกรณีที่ อินพุทมี Voltage มากเกินไป BUFFER ตัวนี้ใช้ OP-AMP ต่อเป็นวงจรขยายสัญญาณ 1:1 เพื่อไม่ให้อินพุท Voltage มีค่าเปลี่ยนไป

A/D ที่ใช้เป็นชนิดความเร็วสูง (flash) จะได้ clock sampling มาจาก clock generator มีความถี่ 8 MHz ดังนั้นสัญญาณที่วัดได้สูงสุดที่พอจะดูรูปร่างได้จะประมาณ 1 MHz สัญญาณที่วัดได้จะถูกนำไปเก็บไว้ที่ STATIC RAM ขนาด 32 Kb ต้องเลือกใช้ RAM ชนิด STATIC RAM เพราะการออกแบบวงจรทำได้ง่ายและมีความเร็วในการ ACCRESS ข้อมูลสูงมาก

วงจร COUNTER เป็นตัวนับ ADDRESS ให้กับ RAM โดยค่า COUNTER จะเพิ่มขึ้นทุกครั้งที่มี clock pluse จ่ายวงจรที่ใช้เป็น Synchronous Counter จะเพิ่มขึ้นทุกครั้งที่มี CLOCK PLUSE จ่ายวงจรที่ใช้เป็น Synchronous Counter จะมีการเปลี่ยนแปลงที่ out put ทุกบิทพร้อมกัน ทำให้การเขียนข้อมูลลงที่ RAM มีตำแหน่งที่แน่นอนไม่ผิดพลาด

CLOCK GENERTOR เป็นวงจรสร้าง clock เพื่อจ่ายให้กับวงจร A/D และวงจร COUNTER ใช้ X-tal ความถี่ 30 MHz แล้วหารด้วย 2 จะได้สัญญาณ duty cycle 50% ความถี่ 16 MHz 1 Phase คือ CLK1

PORT จะทำหน้าที่ควบคุม A/D ให้อ่านสัญญาณแล้วเก็บลงที่ RAM พอเสร็จสิ้นขบวนการ CPU จะอ่านข้อมูลจาก RAM เข้ามา Display ที่จอภาพโดยผ่านทาง Port และวนซ้ำไปเรื่อยๆ ดังนั้น PORT จะทำหน้าที่เสมือนตัวเชื่อมโยงระหว่าง CPU และ A/D รวมทั้ง RAM ด้วย การต่อ RAM ในลักษณะนี้มีข้อดีคือ ไม่ทำให้หน่วยความจำของเครื่องลดลงเพราะ ใช้วิธีการต่อผ่านทาง PORT

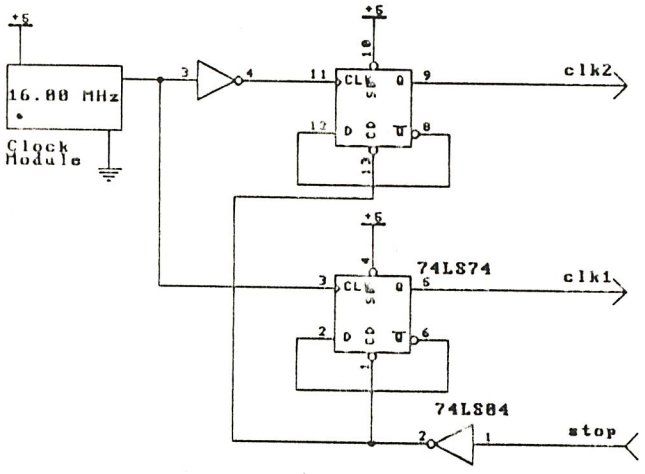


รูปที่ 3.1 แสดงไดอะแกรมของออสซิลโลสโคป (Block digram oscilloscope)

3.2 หลักการออกแบบวงจร clock

ใช้ x-tal ความถี่ 30 MHz ผลิตความถี่ร่วมกันกับ NOT-GATE เบอร์ 7404 ซึ่งจำเป็นต้องใช้ IC TTL เนื่องจากมีความเร็วสูงมากซึ่งสามารถผลิตความถี่ได้สูงกว่า IC ชนิด CMOS

วงจร clock นี้จะนำไปจ่ายให้กับ RAM, Counter และ A/D ซึ่งจะใช้ clock1 phase และ สัญญาณที่ได้จะนำไปหาร ด้วย D - flip flop clock

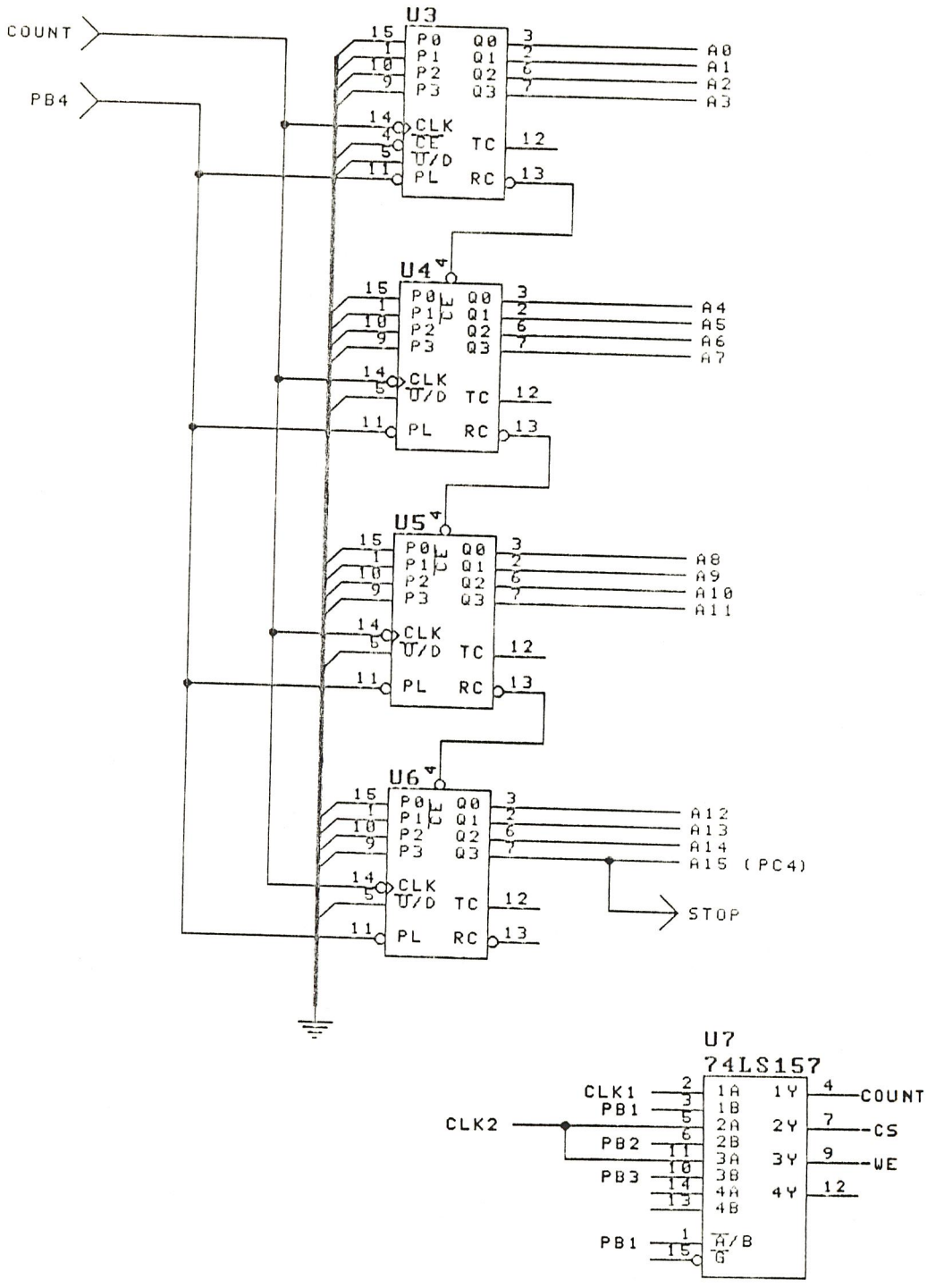


รูปที่ 3.2 แสดงวงจร CLOCK

3.3 การออกแบบวงจร Counter

วงจร counter ที่ใช้ต้องเป็นแบบ synchronous เท่านั้นเพราะถ้าใช้แบบ Asynchronous แล้วจะทำให้การเขียนข้อมูลลงที่ RAM มีปัญหาเนื่องจากจะใช้ Output ของ counter ไปเป็น address จะต้องมีการเปลี่ยนแปลงพร้อมกัน

ในวงจรใช้ Synchronous counter เบอร์ 74193 เป็นชนิด Up/Downในที่นี้จะต่อให้เป็น count up

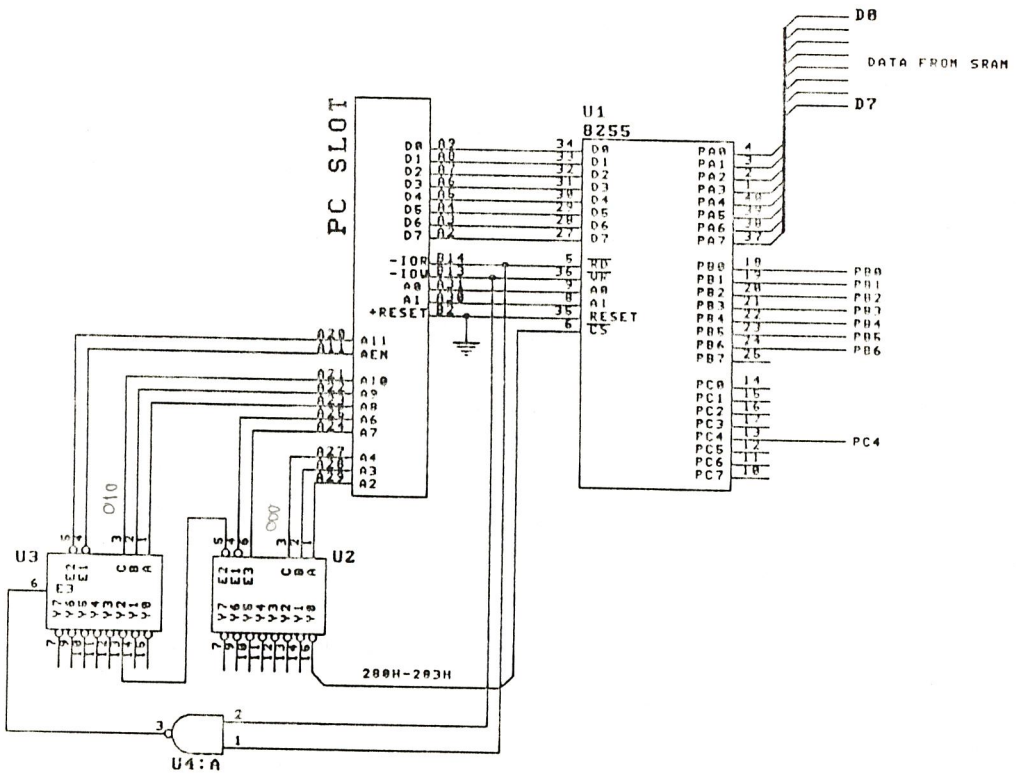


รูป 3.3 แสดงวงจร COUNTER

3.4 การออกแบบวงจร Decode Port

เลือกใช้หมายเลข Port 280H เพราะเป็น Port ว่าง การ Decode ใช้ IC เบอร์ 74138 2 ตัว yo ของ 74138 ที่ 2 ต่อเป็น Chip select ของ 8255 ก็จะได้หมายเลข Port ตั้งแต่ 280H - 283H

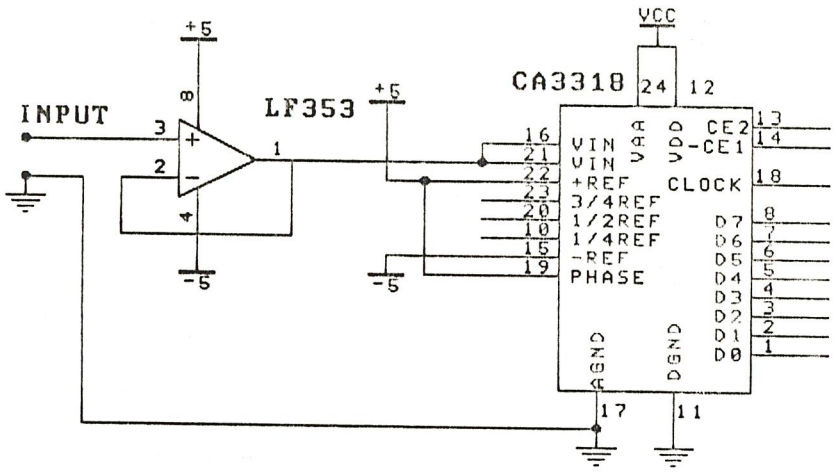
Portเลือกใช้ 8255 มี Port ในตัว 4 Port ใช้งานได้ 3 Port อีก 1 Port เป็นตัว control ดังรูป



รูปที่ 3.4 แสดงวงจร DECODE PORT

3.5 การออกแบบวงจรแปลงสัญญาณ Analog เป็น Digital

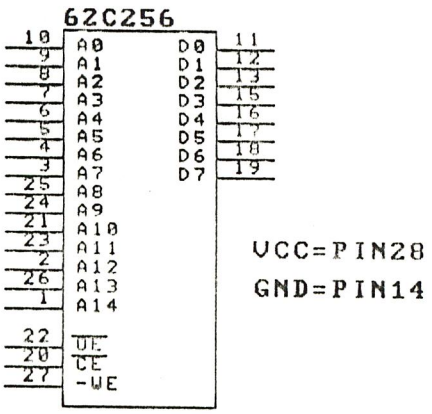
เนื่องจากต้องการให้สามารถวัดความถี่สูงๆ ได้จึงใช้วงจรแปลงสัญญาณ A/D แบบFlashวงจรแปลงสัญญาณ A/D นั้น มีหลายแบบด้วยกัน แต่แบบที่มีความเร็วสูงสุดที่สุดคือ แบบ Flash ในวงจรเลือกใช้เบอร์ GA3318 ของ RCA มีความละเอียดขนาด 8 บิทSampling Rate สูงถึง MHz ที่ Inputของ A/Dเสียหายในกรณีที่สัญญาณInputมีค่าสูงเกินไป



รูปที่ 3.5 แสดงวงจรแปลงสัญญาณ ANALOG TO DIGITAL

3.6 การออกแบบวงจรเก็บข้อมูล RAM

RAM ใช้ขนาด 32 kb เป็น RAM ชนิด Static เหตุที่ต้องใช้เพราะมีความเร็วหรือ Access time ต่ำ ทำให้สามารถที่จะเก็บข้อมูลจาก A/D ได้ทันเลือกใช้เบอร์ 62256 Data ขนาด 8 บิท เพื่อให้เข้ากับ A/D ได้ Address ขนาด 15 เส้น



รูปที่ 3.6 แสดงวงจรเก็บข้อมูล STATIC RAM

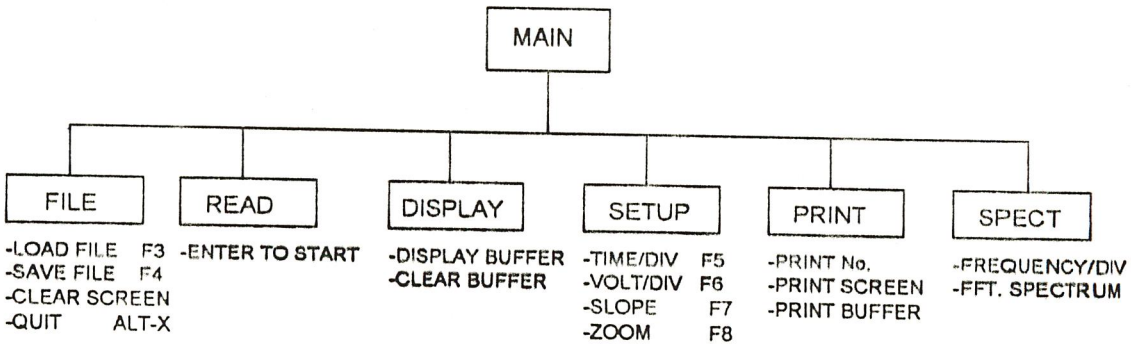
บทที่ 4

การออกแบบ SOFT WARE

SOFTWAREที่ใช้เป็นภาษาCการใช้ภาษาC มีข้อดีมากในการคำนวณทางคณิตศาสตร์ ซึ่งภาษาC ได้จัดเตรียม Function ต่างๆ ไว้ให้มากพออยู่แล้ว ภาษา C ที่ใช้เป็นของ Boland คือ TURBO C เหตุที่ใช้ Turbo C เพราะว่าถูกเล่นทางด้าน Graphics ดีมากมี Front ให้เลือกมาก ศึกษาการใช้คำสั่งได้ง่ายการComplieทำได้ง่ายและเร็วด้วย ทำให้การพัฒนา SOFT WARE เป็นไปได้อย่างรวดเร็ว

จาก Flowchart จะแบ่งออกเป็น 6 ส่วนใหญ่ๆ ดังนี้

- FILE:ในส่วนนี้จะทำหน้าทำจัดการเกี่ยวกับแฟ้มข้อมูลคือSave จัดเก็บรูปร่างของสัญญาณลงบนแผ่น DISK ในทางตรงกันข้ามคือ LOAD จะอ่านข้อมูลจากแผ่น DISK ในทางตรงกันข้ามคือ LOAD จะอ่านข้อมูลจากแผ่น DISK มา Display ที่จอภาพทำให้สามารถเก็บรูปร่างของสัญญาณต่างๆ ไว้ได้มาก พร้อมทั้งยังสามารถเปรียบเทียบเฟสของสัญญาณโดยการ LOAD ข้อมูลซ้อนกันได้ นอกจากนี้ยังมี Function สำหรับ Clear จอภาพด้วย
- READ: ทำการอ่านข้อมูลจาก Flash A/D มา DISPLAY ที่จอภาพผ่าน Port หมายเลข 280H CPU จะส่งคำสั่งไปที่ Card A/D ให้ A/D อ่านข้อมูลแล้วนำไปเก็บที่ RAM บน Card แล้ว CPU จะวนอ่านมาแสดงผลที่จอภาพ ซึ่งสามารถใช้ได้ทั้งจอแบบ EGA/VGA และแบบ Monochrom ความถี่สูงสุดที่วัดได้ 1 MHz ยังสามารถที่จะ Store สัญญาณเพื่อดูรายละเอียดของรูปคลื่นได้
- DISP: เป็นส่วนที่นำเอาข้อมูลจาก Bufferมาแสดงผลบนจอภาพซึ่งข้อมูลนั้น
- ก็คือรูปคลื่นสัญญาณที่ Store จากการอ่าน และยังมี Function พิเศษสำหรับทำการ Clear Buffer เลื่อนดูข้อมูล (รูปคลื่น) ใน Buffer ได้
- SETUP: ทำหน้าที่ Set ค่า Parameter ต่างๆ ของ Storage Scope ได้แก่ Time/Division ซึ่ง Set ได้ถึง 5 ไมโครเซกต่อช่อง (นั่นคือวัดสัญญาณได้สูงถึง 1 MHz) VOLT/DIV เลือกค่าได้ 2 ค่าคือ 2.5 โวลท์ต่อช่อง และ 12.5 โวลท์ต่อช่อง SLOP+ หรือ SLOP-
- PRINT: ในส่วนนี้จะพิมพ์ภาพที่หน้าจอออกไปที่PRINTERโดยจะทำให้สามารถเลือกรูปแบบการพิมพ์ได้ว่าจะพิมพ์หน้าจอหรือพิมพ์ BUFFER การพิมพ์นั้นจะใช้มาตราฐานของเครื่องพิมพ์ EPSON 9 PIN
- SPECT: เป็นส่วนที่ทำหน้าที่ที่นำเอาสัญญาณที่ถูก STORE แล้วจาก BUFFER มาทำการคำนวณโดยใช้หลักการของ FAST FOURIER TRANSFORM จำนวนจุดจะคงที่ไว้ที่ 256 จุด แต่จะให้สามารถป้อน SCALE MAX FREQUENCY (F MAX) ค่าของ F max ก็ไม่ควรเกิน 1 MHz



รูปที่ 4.1 แสดง FLOWCHART SOFTWARE

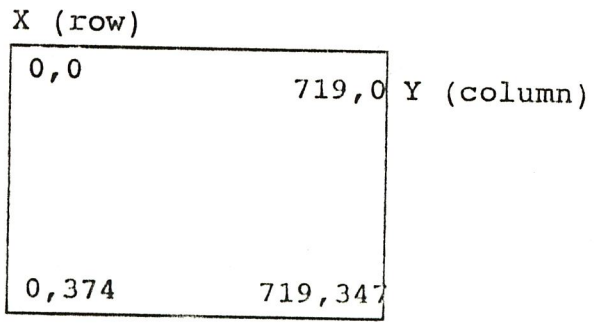
4.1 การแสดงผลออกทางเครื่องพิมพ์

โปรแกรมย่อยแสดงผลออกทางเครื่องพิมพ์ การแสดงผลออกทางเครื่องพิมพ์จะทำการพิมพ์กราฟฟิคจากจอภาพครั้งละหนึ่งจอภาพ ซึ่งจอภาพต้องเป็นชนิด MONOCROM ใช้ HERCULES GRAPHICS CARD เครื่องพิมพ์ที่ใช้พิมพ์เป็นเครื่อง EPSON

โปรแกรมที่เขียนจะมีส่วนสำคัญ 2 ส่วน ได้แก่ ส่วนที่อ่านข้อมูลจาก VIDEO RAM และส่วนที่ส่งข้อมูลออกไปพิมพ์ที่เครื่องพิมพ์

4.2 การอ่านข้อมูลจาก VIDEO RAM

จอภาพชนิด MONOCROM ใช้ HERCULES GRAPHICS CARD จะแสดงผลออกทางจอภาพด้านแนวนอน 720 จุด และแนวตั้ง 348 จุด ดังแสดงในรูปที่ 4.2

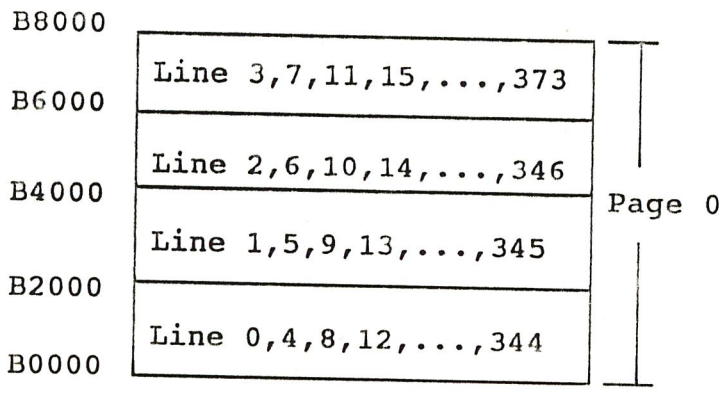


รูปที่ 4.2 GRAPHICS-MODE SCREEN COORDINATES

การเก็บข้อมูลในหน่วยความจำจะเก็บรายละเอียดขนาด 720 จุด ตามแนวนอนและ 348 เส้น ตามแนวตั้ง ตำแหน่งหน่วยความจำเริ่มต้นซึ่งเก็บข้อมูลการแสดงผลบนจอภาพ MONOCROM ใน PAGE 0 จะเริ่มเก็บที่ตำแหน่ง 80000 h ไปจนถึงตำแหน่ง 87FFFh จะครบ 1 จอภาพสำหรับรายละเอียดการเก็บข้อมูลใน VIDEO RAM นั้น
แสดงดังรูป

- LINE0 เริ่มเก็บที่ตำแหน่ง 80000h ไปจนครบ 1 แถว คือ 90 BYTE
- LINE 1 เริ่มเก็บที่ตำแหน่ง 82000h ไปจนครบ 1 แถว คือ 90 BYTE
- LINE 2 เริ่มเก็บที่ตำแหน่ง 84000h ไปจนครบ 1 แถว คือ 90 BYTE
- LINE 3 เริ่มเก็บที่ตำแหน่ง 86000h ไปจนครบ 1 แถว คือ 90 BYTE
- LINE 4 เริ่มเก็บที่ตำแหน่ง 8005Ah ซึ่งเก็บต่อจาก LINE 0 ไปอีก 90 BYTE
- LINE 5 เริ่มเก็บที่ตำแหน่ง 8005Ah ซึ่งเก็บต่อจาก LINE 0 ไปอีก 90 BYTE
- LINE 6 เริ่มเก็บที่ตำแหน่ง 8005Ah ซึ่งเก็บต่อจาก LINE 0 ไปอีก 90 BYTE
- LINE 7 เริ่มเก็บที่ตำแหน่ง 8005Ah ซึ่งเก็บต่อจาก LINE 0 ไปอีก 90 BYTE

การเก็บข้อจะเป็นลักษณะนี้ไปเรื่อยๆ จนครบ 1 จอภาพ ที่หน่วยความจำ 8000h ถึง 81FFFh จะเก็บข้อมูลใน LINE 0, 4, 8, 16,..., 344 ส่วนหน่วยความจำ 80000h, 84000h, 86000h ก็เก็บในลักษณะเดียวกัน



รูปที่ 4.3 GRAPHICS-MODE DISPLAY BUFFER ORGANIZATION

การนำเอาข้อมูลออกมาพิมพ์จะทำการอ่านข้อมูลจาก VIDEO RAM ตำแหน่งที่ชี้จุดต่ำสุดของ COLUMN แรกในจอภาพ (0,348) คือ หน่วยความจำที่ 87E96h เก็บขึ้นมา 4 ไบท์ของ LINE ใน COLUMN แรก จากนั้นบวกเข้าไปอีก 90 ไบท์ เพื่อเก็บใน LINE ถัดมาอีก 4 ไบท์ เก็บไปเรื่อยๆ จนถึง LINE แรกของ COLUMN แรก คือหน่วยความจำที่ 80000h ชี้จุด (0, 0) ในจอภาพดังแสดงในรูป 4.10 แล้วส่งออกไปพิมพ์ที่เครื่องพิมพ์ ส่วน COLUMN ที่เหลือก็จะทำในลักษณะเดียวกันจนครบ 1 จอภาพ

ลือก็จะทำในลักษณะเดียวกันจนครบ 1 จอภาพ

4.3 การนำข้อมูลออกมาพิมพ์ที่เครื่องพิมพ์

ในส่วนนี้จะเป็นการนำข้อมูลที่อ่านได้จาก VIDEO RAM ออกมาพิมพ์การพิมพ์จะนำ 1 COLUMNในหน้าจอภาพมาพิมพ์เป็น1LINEบนเครื่องพิมพ์การส่งข้อมูลออกทางเครื่องพิมพ์จะเรียกใช้บริการอินเตอร์รัพท์เลขที่ 17h ซึ่งข้อมูลที่ส่งให้เครื่องพิมพ์มี 2 ชนิด คือ

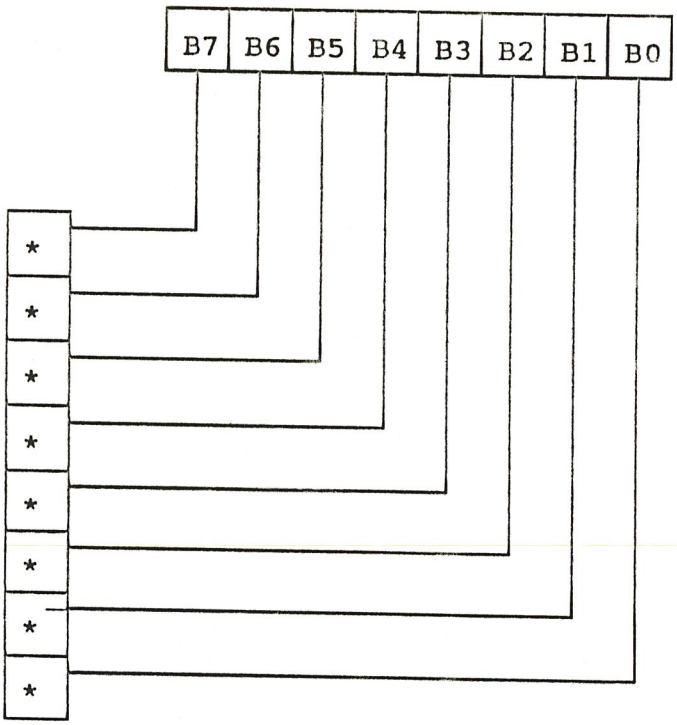
1.CONTROL CODE จะเป็นส่วนที่ไปเซ็ทให้เครื่องพิมพ์พร้อมที่จะพิมพ์ในโหมดกราฟฟิก ซึ่งรายละเอียดของ CONTROL CODE ที่ใช้มีดังนี้คือ

- ESC 3 คือ เลือกเว้นบรรทัดขนาด n/216 นิ้ว
- ESC L คือ เลือกการพิมพ์แบบ DOUBLE-DENSITY GRAPHICS MODE
- LF คือ เลื่อนบรรทัดเมื่อพิมพ์ครบ 1 บรรทัด
- GR คือ ขึ้นบรรทัดใหม่เมื่อพิมพ์ครบ 1 บรรทัด

รายละเอียดของ CONTROL CODE สามารถศึกษาเพิ่มเติมได้จากคู่มือเครื่องพิมพ์แต่ละเครื่อง

2. DATA คือ ส่วนของข้อมูลที่เรานำมาจาก VIDEO RAM เพื่อนำไปพิมพ์ ในการพิมพ์จะพิมพ์แบบ DOUBLE-DENSITY GRAPHICS MODE หัวพิมพ์ของเครื่องพิมพ์จะทำการพิมพ์ได้ครั้งละ 8 จุด (พิมพ์ซ้ำที่เดิม 2 ครั้ง) ตามแนวตั้งจากรูปที่ 4.11 จะเห็นได้ว่าเมื่อทำการส่งค่า 1 ไบท์ เครื่องพิมพ์จะตีความหมายโดยเอาค่าบิทที่ 7 แทนหัวเข็มบนสุดและบิทที่ 0 แทนหัวเข็มล่างสุดของหัวพิมพ์

การพิมพ์จะนำจุดที่มุมล่างซ้ายสุดของจอภาพมาพิมพ์ที่มุมบนซ้ายสุดของเครื่องพิมพ์พิมพ์ไปจนถึงมุมบนขวาสุดของเครื่องพิมพ์ซึ่งก็หมายถึงจุดบนสุดด้านซ้ายของจอภาพ จากนั้นเราจะส่ง CONTROL CODE ให้เครื่องพิมพ์เพื่อให้เครื่องพิมพ์ขึ้นบรรทัดใหม่และเลื่อนบรรทัดจากนั้นจะนำเอา COLUMN ต่อไปของจอภาพมาพิมพ์ เป็นเช่นนี้ไปเรื่อยๆ จนสุด COLUMN หน้าจอด้านขวาหรือ LINE ล่างสุดของเครื่องพิมพ์



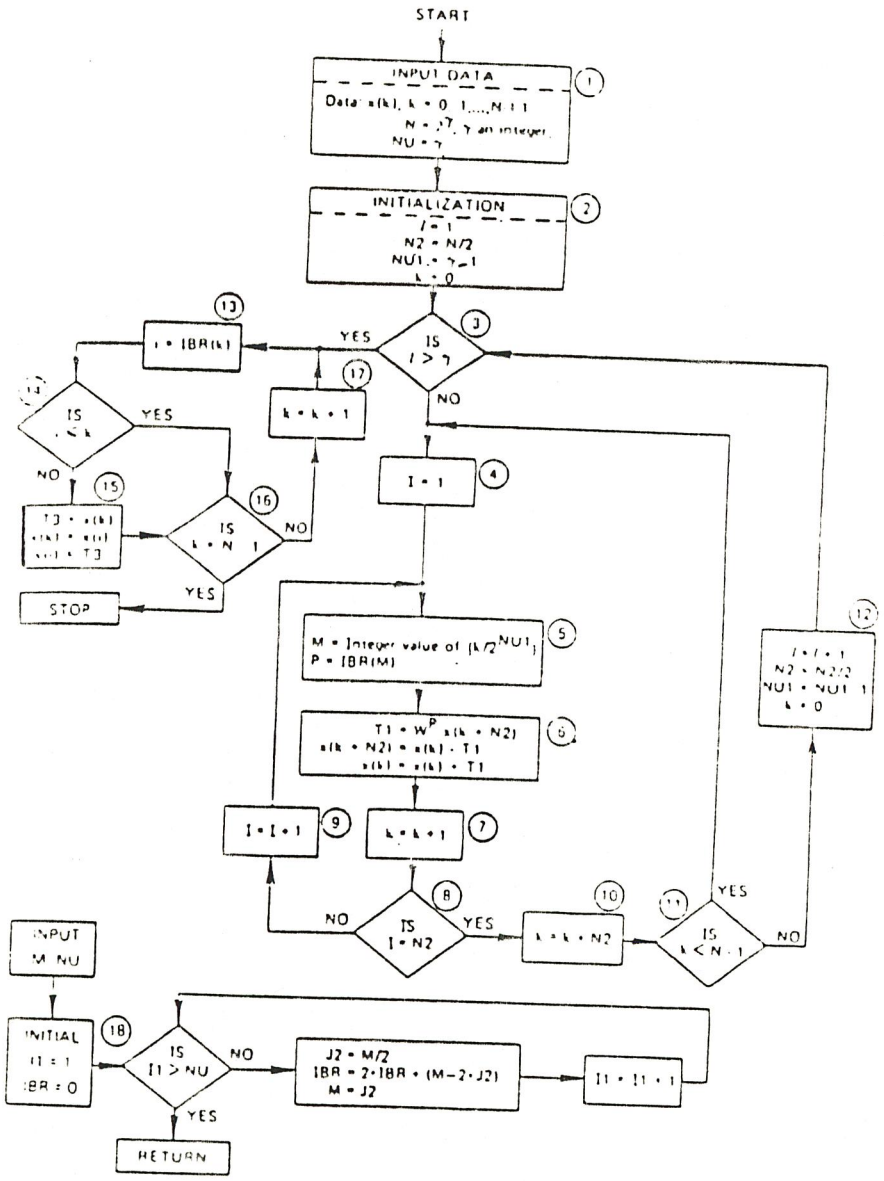
รูปที่ 4.4 ความสัมพันธ์ของหัวเข็มของเครื่องพิมพ์กับตำแหน่งบิท

ผังการทำงานของโปรแกรม

1) จะทำการเก็บค่า Register ต่างๆ จากโปรแกรมหลักไว้ใน Stack
2) เก็บข้อมูลจาก Vedio Ram ใน Column ที่ 1 ไปเก็บไว้ใน Buffer
3) นำข้อมูลจาก Buffer ออกไปพิมพ์
4) ทำซ้ำข้อ 2 และ 3 จนข้อมูลครบ 1 จอภาพและพิมพ์ครบ 1 แผ่น
5) คืนค่าจาก Stack ให้กับ Register เพื่อกลับเข้าสู่โปรแกรมหลัก

4.5 โปรแกรมแปลง Fast Fourier Transform

การแปลง Fast Fourier Transform นั้น สามารถที่จะให้คอมพิวเตอร์แปลงได้โดยใช้ภาษา C ซึ่งสามารถที่จะเขียนเป็น Flow Chart แสดงได้ดังรูปที่ 4



รูปที่ 4.5 แสดง FLOW CHART การแปลง FAST FOURIER TRANSFORM

จาก Block Diagram, Block ที่ (1) และ (2) สัญญาณอินพุทจะถูกเก็บเข้าไปที่ส่วนของ Real Aray ส่วน Imagine จะถูกกำหนดค่าให้เป็นศูนย์ แล้วทำการป้อนค่าของจำนวนจุดที่ต้องการจะคำนวณ (n) และค่าของ ah ของ h

Block ที่ (3) จะทำการ Check Aray ที่จะคำนวณว่ามีค่ามากกว่า r หรือยังถ้ามากกว่าจะกระโดดไปทำงานที่ Block (13)

Block (4) จะ Set ค่าของ Counter ซึ่งค่าของ Counter นี้ จะเป็นจำนวนของ Dual Node Pairs

Block(5) และ (6) จำคำนวณค่า Integer ของ ($K/2^{NU1}$) แล้วนำค่านั้นไปผ่านขบวนการ Bit Reverse แล้วทำการคำนวณค่าของ WP ที่ X(k+N2) แล้วนำเอาผลลัพธ์ไปเก็บไว้ที่ส่วนชั่วคราว ซึ่งผลลัพท์ที่ได้จะมีลักษณะเป็น Dual Node

Block(7)และ (8) เพิ่มค่าของ K แล้ว Check ดูว่า Counter (I) นั้น = N ถ้ายังก็วนลูปไปที่ Block (5) และ (C) ถ้ามากกว่าก็จะให้ค่า K=K+N2 (Block 10) แล้ว Check ว่า K<N-1 หรือยัง ถ้ายังจะเริ่มทำการ Initialzed Data ใหม่แล้ววนกลับไปที่ Blocks ถ้าใช้ก็กระโดดไป Block

Block(13),(14),(15),(16), (7) เป็นขบวนในการทำ Bit Reverse หลังจากผ่านขบวนการนี้แล้วก็จะได้ส่วนของการ Transform เก็บที่ Real และ Imagine Buffer ถ้าต้องการหา Spectrum ก็จะนำเอาค่าของ Real และ Imagine ไปทำการรวมกันแบบ Vector ดังสมการ

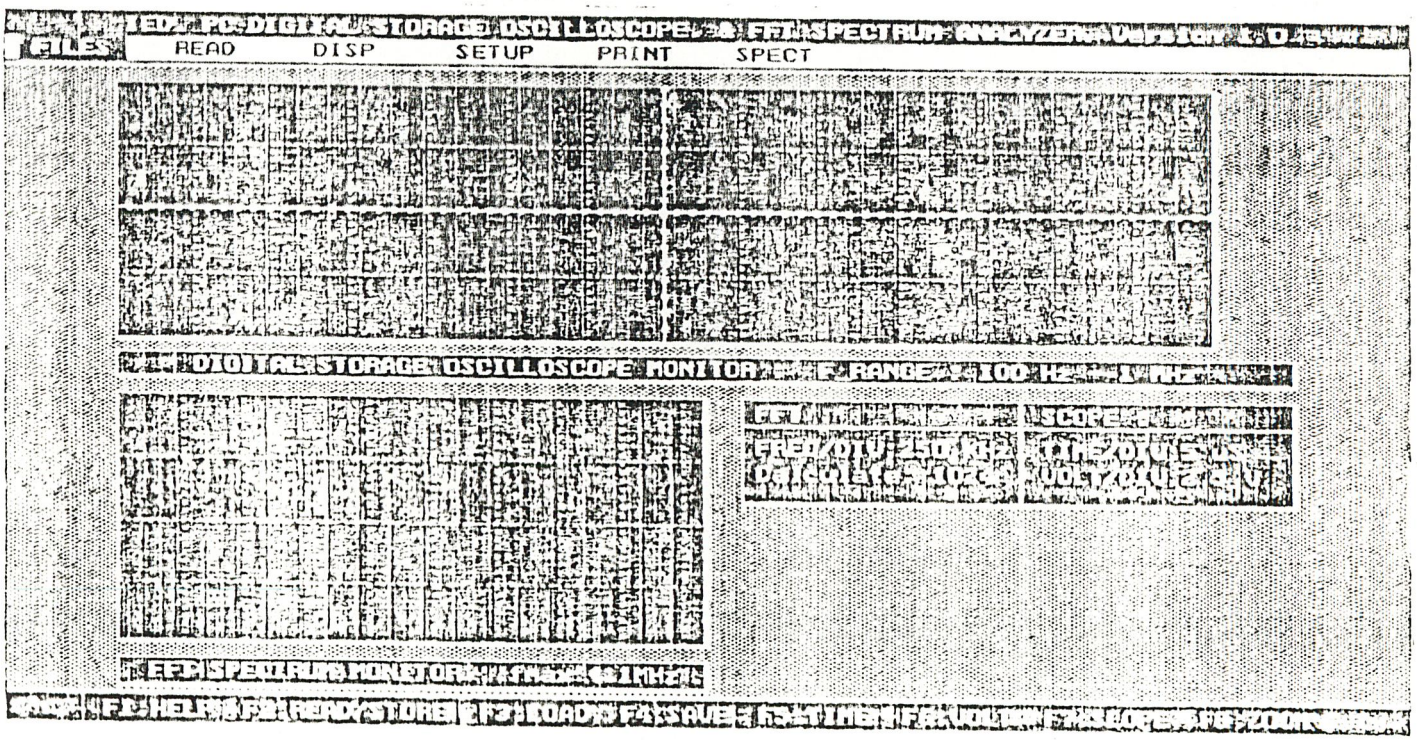
$$Power.spectrum = \sqrt{(\mathrm{Re}al)^2 + (\mathrm{Im}agine)^2}$$

4.6 การใช้งานโปรแกรม

ส่วนประกอบของโปรแกรมจะมีชื่อไฟล์ชื่อ STORE.EXE โปรแกรมนี้ใช้กับเครื่องพีซี (PC) ทั่วไปๆ และ สมารถใช้กับจอภาพแบบ EGA,VGA,MONOCHOME การใช้งานให้ใส่แผ่น DISK และพิมพ์ว่า

A:\STORE (ENTER)

ที่จอภาพจะแสดงดังรูปที่ 4.6



รูปที่ 4.6 แสดงผลของจอภาพและ MENU หลักเมื่อเปิดครั้งแรก

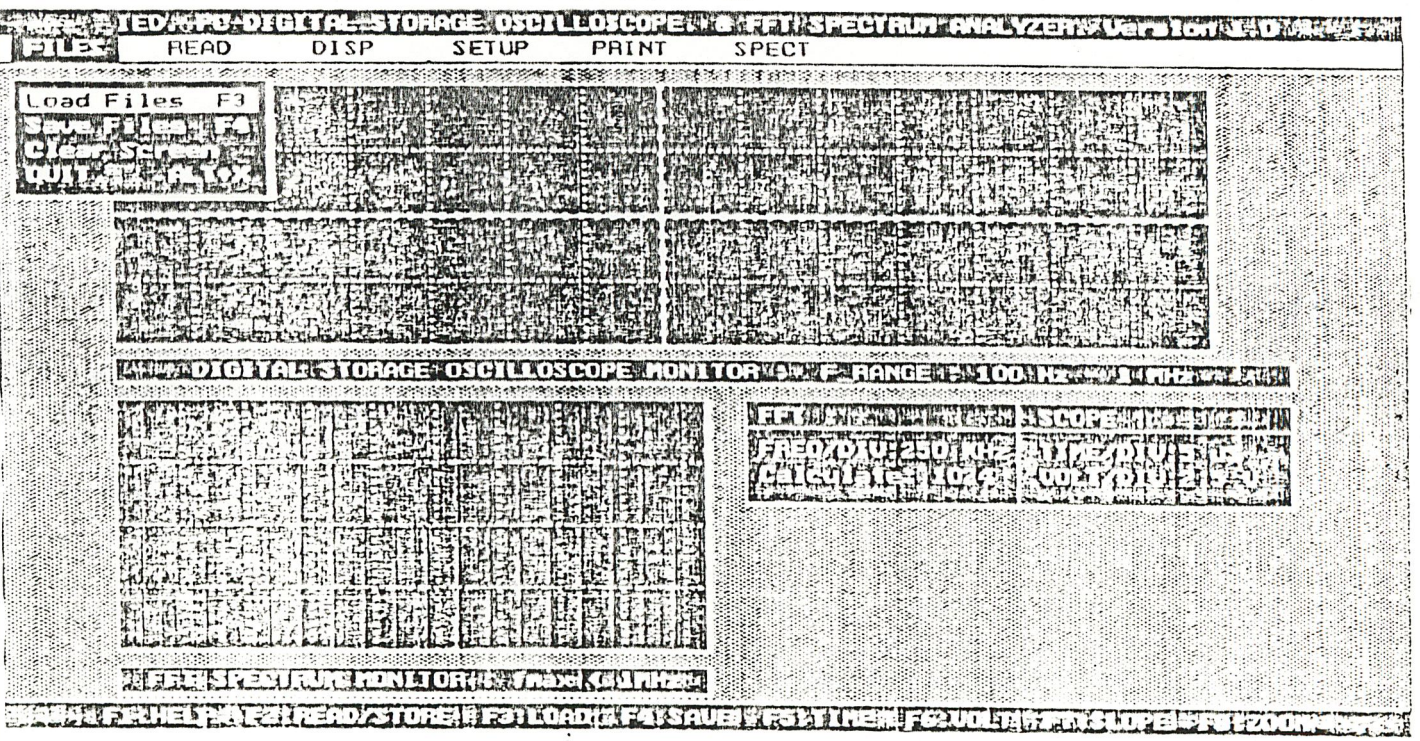
FILE เมนูส่วนนี้จะทำหน้าที่เกี่ยวกับแฟ้มข้อมูลคือ

-Save ใช้เก็บข้อมูลจาก Buffer ลงแผ่น Disk หรือHard Disk ถ้ากดเมนูในส่วนนี้โปรแกรมจะถามชื่อที่จะ Save หลังจากนั้นให้ผู้ใช้ป้อนชื่อแล้วกด Enter เครื่องก็ Save ข้อมูลที่เป็นรูปสัญญาณลงไปที่แผ่น Disk

-Load เมื่อกดเมนูนี้โปรแกรมจะถามชื่อที่ต้องการ Load มาจากแผ่น Disk ให้ป้อนชื่อที่ต้องการ Load แล้วกด Enter โปรแกรมจะอ่านข้อมูลจากแผ่น Disk มาเก็บใน Buffer และแสดงผลออกมาทางจอภาพ การ Load นั้นถ้า Load ใหม่ข้อมูลใหม่จะทับข้อมูลเดิมทันทีแต่ที่หน้าจอจะยังแสดงข้อมูลเก่าอยู่ในลักษณะนี้จะทำให้สามารถเปรียบเทียบเฟสของสัญญาณได้

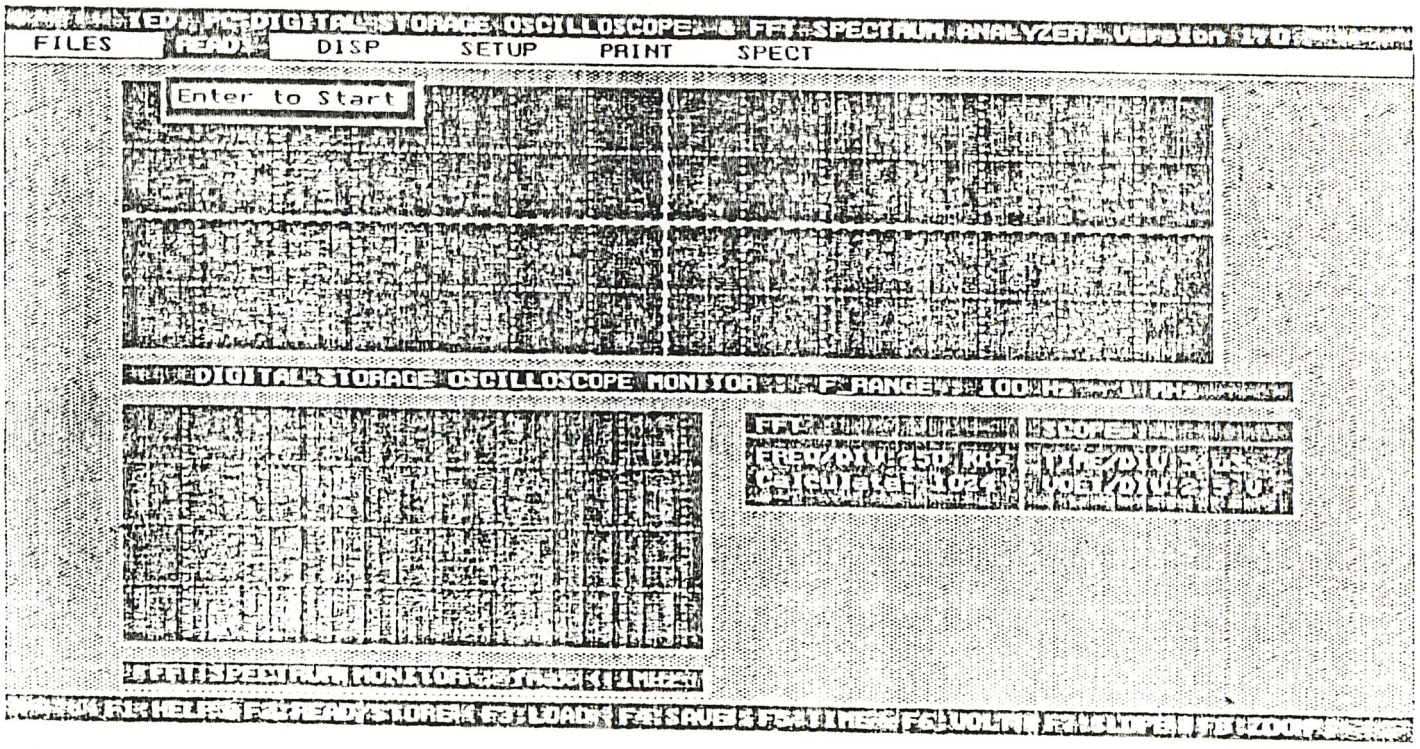
-Clear จะทำการ Clear จอภาพเท่านั้นแต่ข้อมูลใน Buffer จะยังอยู่แต่รูปคลื่นที่หน้าจอจะถูก Clear เท่านั้น

-Quit เป็นการยกเลิกการทำงานทั้งหมดของโปรแกรมคืนการใช้งานให้กับ Dos



รูปที่ 4.7 แสดง Menu ของ File

-Read ใน Menu ส่วนนี้โปรแกรมจะทำการอ่านข้อมูลจาก A/D และนำเอารูปสัญญาณนั้นมา Display ที่จอภาพในลักษณะของสัญญาณต่อเนื่อง การแสดงผลค่อนข้างจะ Real Time อยู่พอสมควรและถ้าต้องการเก็บสัญญาณ (Store) ก็ให้กด F2 ข้อมูลนั้นจะถูกเก็บลงใน Buffer

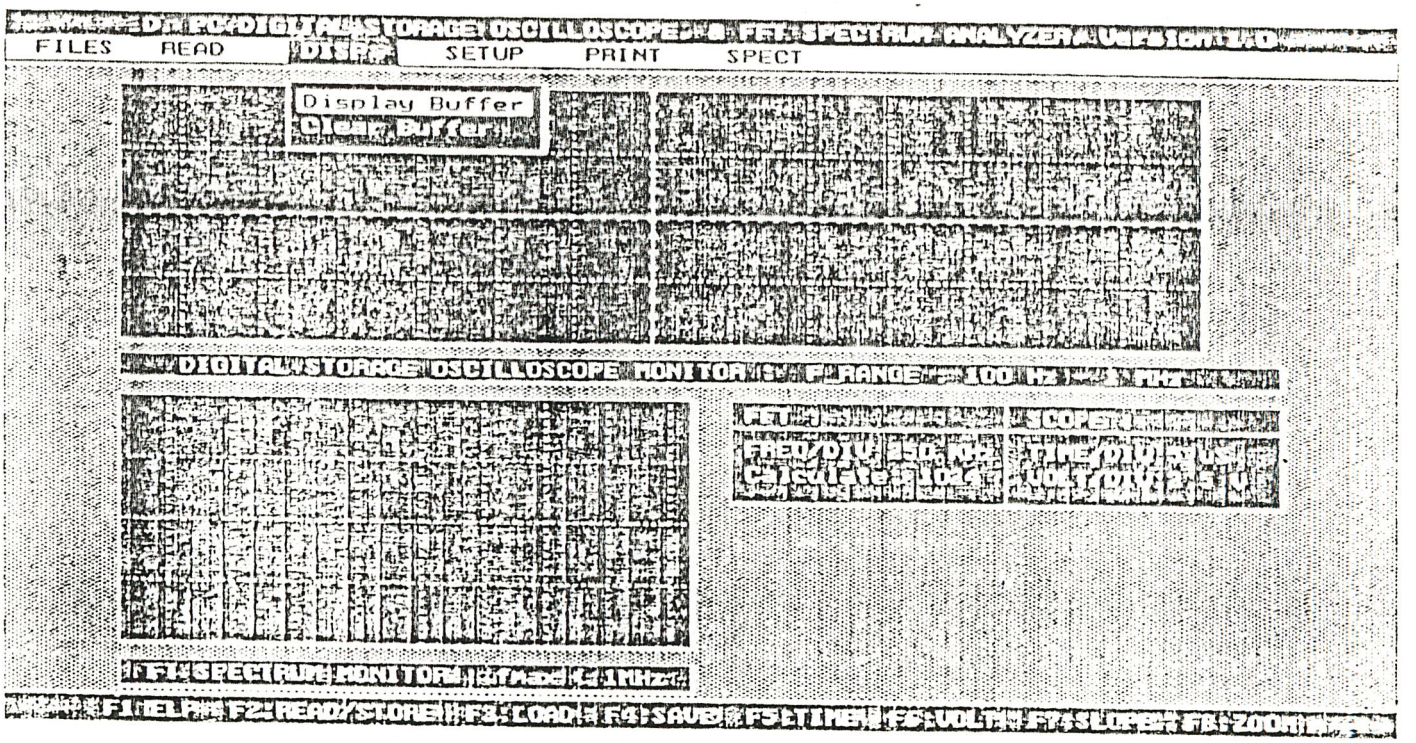


รูปที่ 4.8 แสดง Menu ของ Read

-Disp ใน Mode นี้จะมีอยู่ 2 ส่วนคือ

- Display Buffer จะนำเอาข้อมูลจาก Buffer มาแสดงที่จอภาพสามารถเลื่อนดูภาพได้

-Clear Buffer จะทำการลบข้อมูลใน Buffer ออกและ Clear จอภาพด้วย



รูปที่ 4.9 แสดง Menu ของ Display

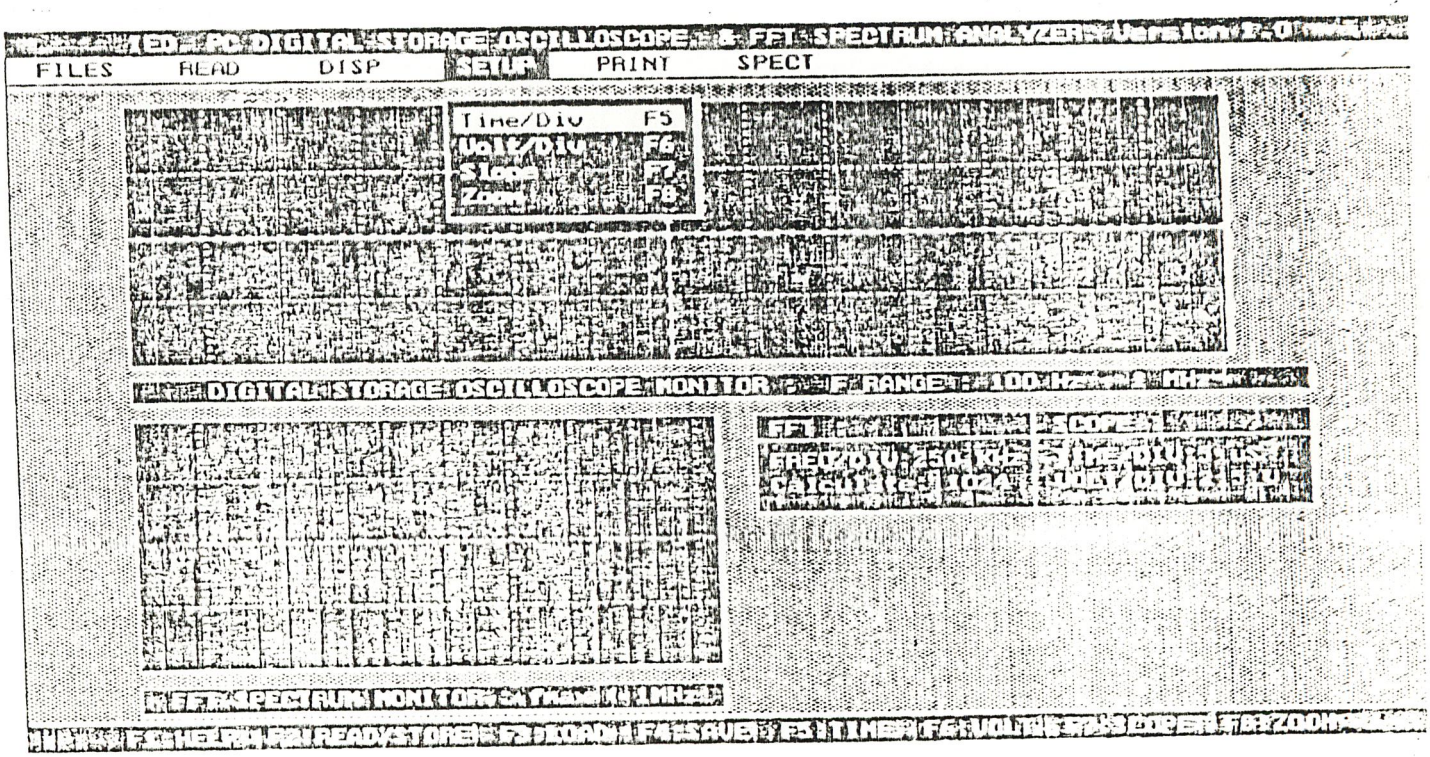
-Setup โปรแกรมในส่วนนี้จะใช้ Set ค่า Parameter ต่าง ๆ ของ Scope เพื่อให้สามารถวัดสัญญาณได้รูปคลื่นที่ถูกต้อง

-Time/Div เมื่อเลือก Menu ส่วนนี้เครื่องจะแสดงค่าของ Time ต่อช่องเลือก 5uS,20uS,0.1mSซึ่งมีค่าดังนี้ ,0.5mS,1mS,5mS,20mS โดยผู้ใช้ต้องเลือกค่าเหล่านี้ ให้เหมาะสมกับความถี่ของสัญญาณที่ป้อนทาง Input ซึ่งเท่ากับโปรแกรมแสดงสัญญาณได้อย่างถูกต้อง

-Volt/Div ใช้ตั้งค่าของ Voltage ต่อช่องสัญญาณ

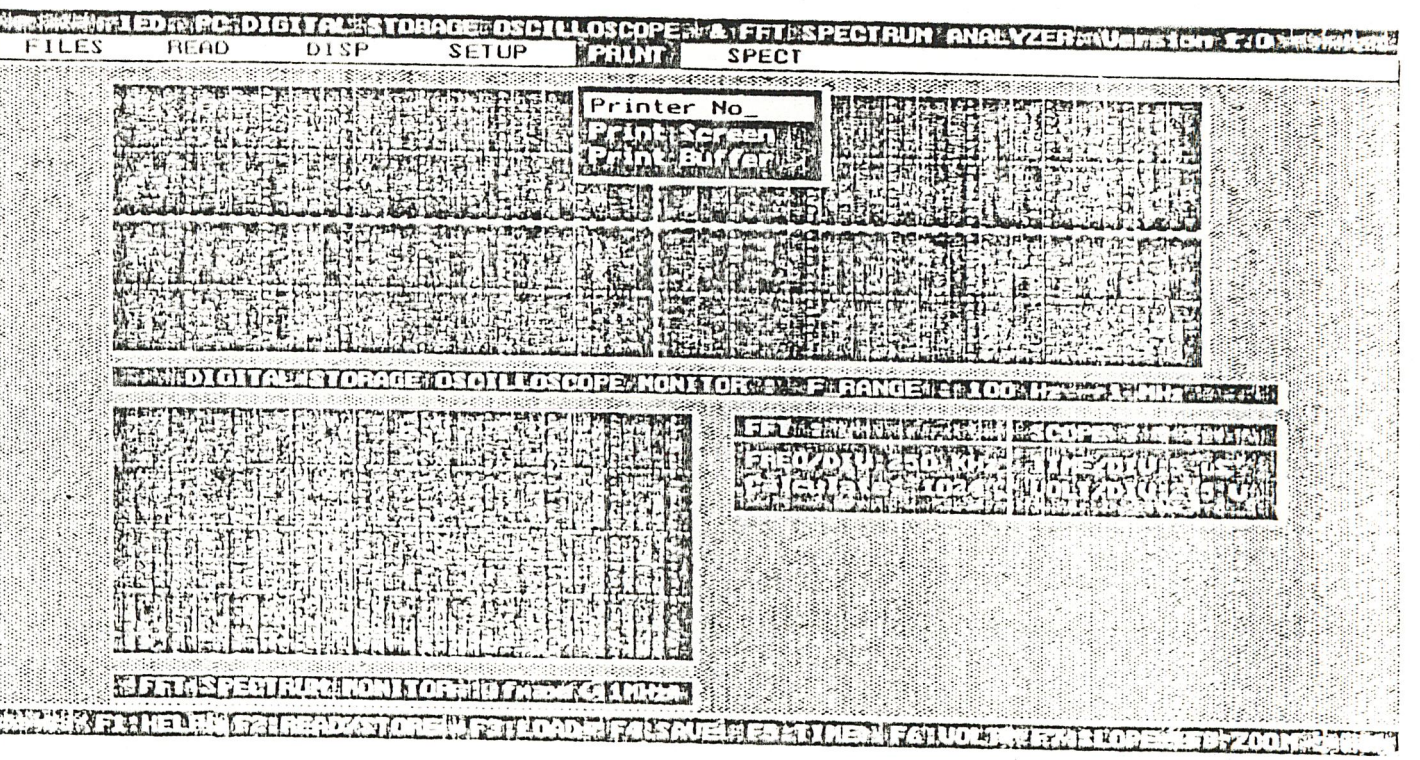
-Slope ใช้สำหลับตั้งค่าสัญญาณของจุดเริ่มต้นว่าจะให้เริ่มที่ขึก บวก หรือขึกลบ

-Zoom ใช้สำหรับยืดหรือขยายหรือย่อรูปร่างของสัญญาณที่ถูก Storeไว้แล้วซึ่งเป็นประโยชน์อย่างมากในการ Transiant ของสัญญาณ



รูปที่ 4.10 แสดง Menu ของ Setup

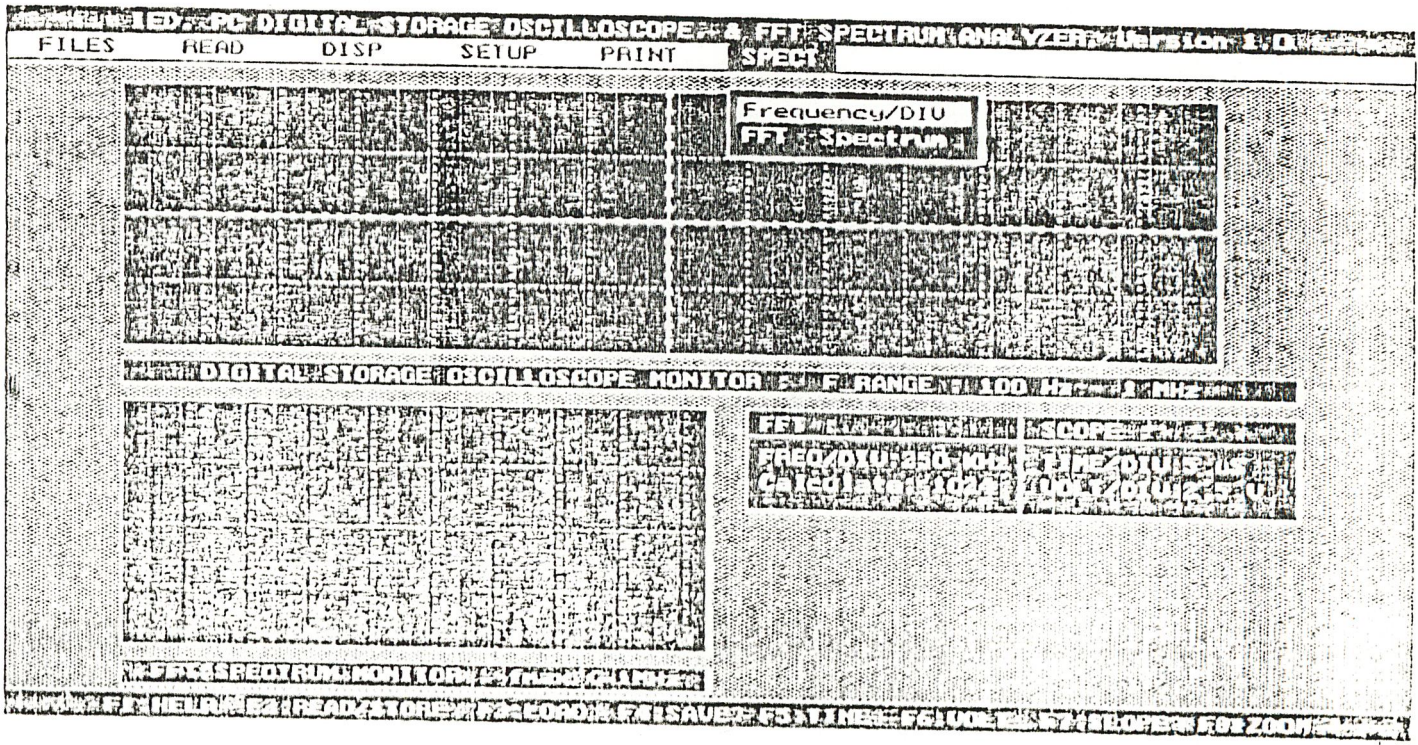
-Print โปรแกรมส่วนนี้ใช้สำหรับพิมพ์รูปภาพที่จอภาพออกทางเครื่องพิมพ์โดยการพิมพ์สามารถเลือกรุ่นของ Printer ได้และยังเลือกอีกว่าจะเลือกพิมพ์ เฉพาะรูปของสัญญาณ Spect โปรแกรมในส่วนนี้จะนำเอาข้อมูลที่ Store อยู่ใน Buffer มาเข้าขบวนการแปลงสัญญาณจาก Time Domain ให้อยู่ในรูปของ Frequency Domain



รูปที่ 4.11 แสดง Menu ของ Print

SPECT โปรแกรมในส่วนนี้จะนำเอาข้อมูลที่ store อยู่ใน buffer มาเข้าขบวนการแปลงสัญญานจาก time domain ให้อยู่ในรูปของ frequency domain

-Freq/Div ในส่วนนี้จะใช้ตั้งค่า Scale ของความถี่ต่อช่องซึ่งมีให้เลือกดังนี้ 256KHz,128KHz,,64KHz,32KHz Plot Spectrum จะทำการคำนวณค่า Fast Fourier Tranform ของรูปคลื่นของสัญญาณแล้วนำเอาค่านั้นมา Plot Graph เพื่อแสดง Spectrum ของสัญญาณ



รูปที่ 4.12 แสดง Menu ของ SPECT

บทที่ 5

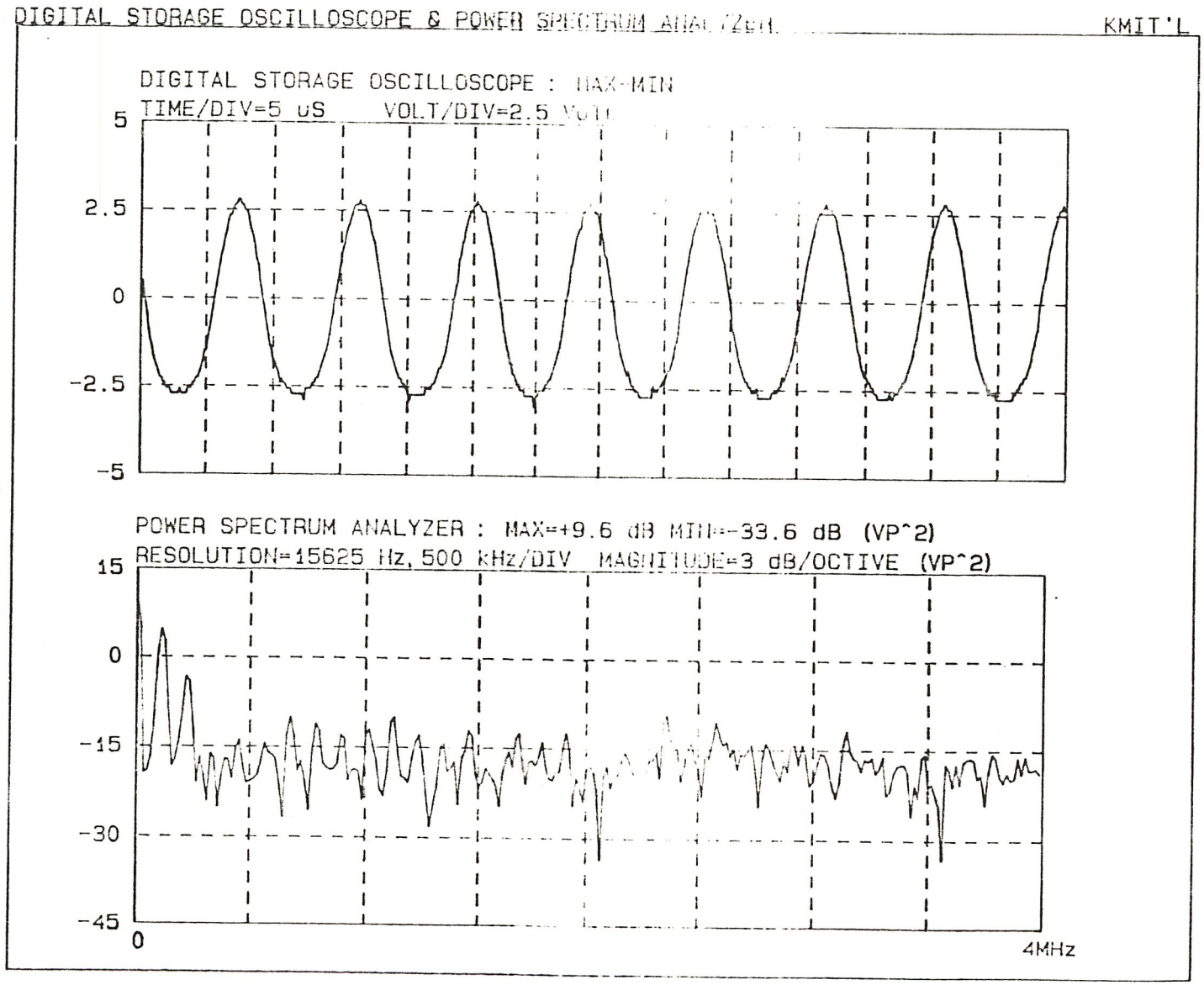
ผลการทดลองและสรุปผลการทดลอง

5.1 ผลการทดลอง

ในส่วนผลการทดลองนั้นใช้การพิมพ์รูปสัญญาณที่ได้จากจอภาพของเครื่องคอมพิวเตอร์จริงที่ได้เขียนโปรแกรมเอาไว้ โดยได้ทดลองวัดสัญญาน Sine , Squar , และ Triangle โดยป้อนสัญญานจาก Signal Generater เข้าที่ Card สัญญานที่ได้จะนำมาแสดงที่จอภาพ

รูปที่ 5a แสดงลักษณะการต่อวัดสัญญาน

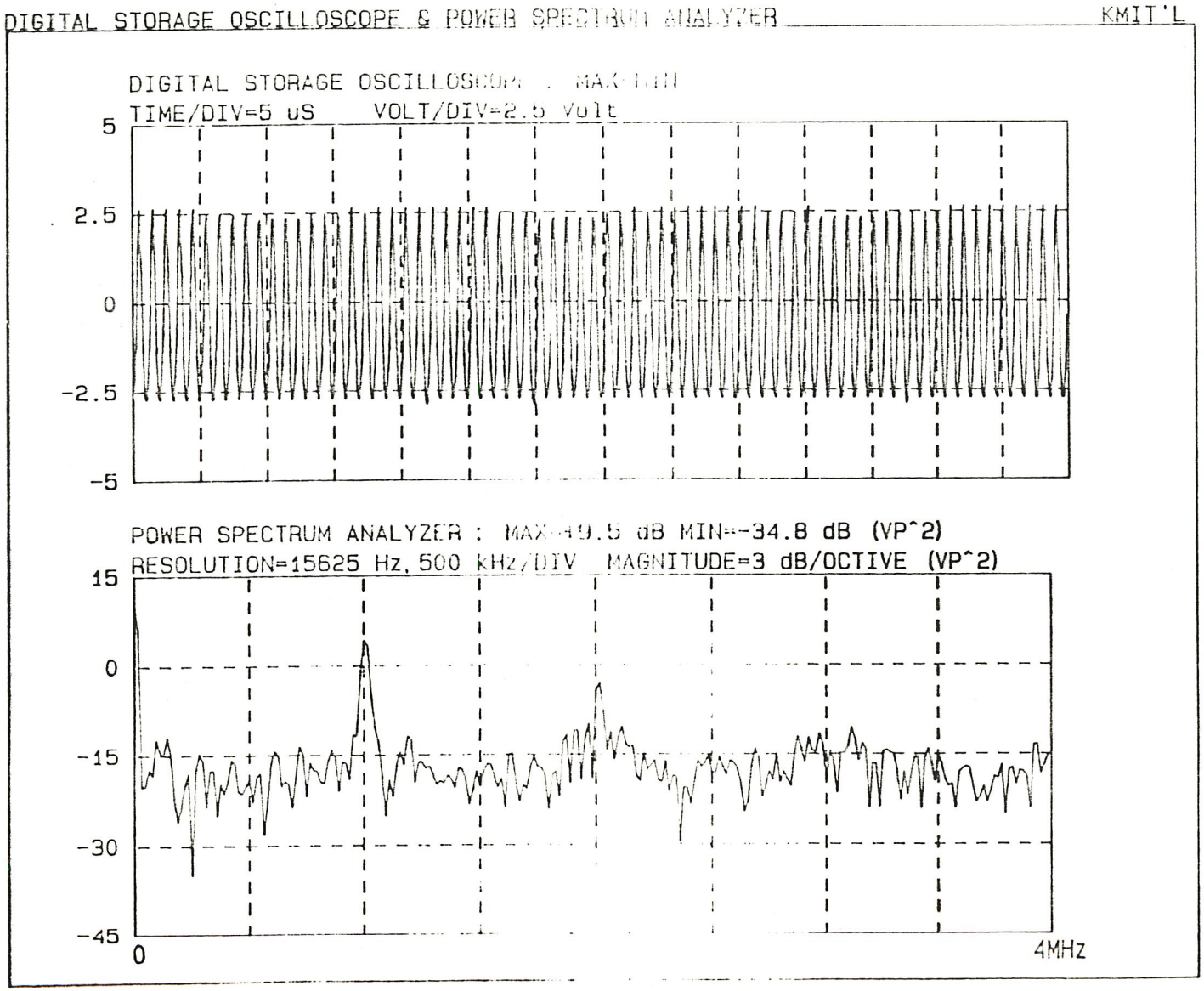
รูปที่จะเป็นผลของการวัดสัญญานที่ความถี่ต่างๆ พร้อมทั้งแสดงสเปคตรัมที่วัดได้ จากผลการทดลองที่ผ่านมาสามารถวัดสัญญานได้จริง มีสเกล (scale) ต่างๆเท่าของจริง โดยสัญญานที่ได้ใกล้เคียงความเป็นจริงมาก แต่ที่ความถี่สูงกว่า 1 MHz จะไม่สามารถมองเห็นรูปสัญญานได้เพราะสัญญานจะถี่ติดกันมากเกินไป ผลการทดลองทั้งหมดแสดงได้ดังรูปต่อไปนี้



รูปที่ 5.1 แสดงรูปสัญญาณที่ได้จากการสุ่มของสัญญาณ Sine ความถี่ 100 KHz

รูปที่ 5.2 แสดงรูปสัญญาณและสเปกตรัมของสัญญาณ Sine คลื่นความถี่ 500 KHz

รูปที่ 5.3 แสดงรูปคลื่นและสเปกตรัมของสัญญาณ Square ความถี่ 500 KHz



รูปที่ 5.4 แสดงรูปสัญญาณและสเปกตรัมของสัญญาณ Sine คลื่นที่ 1 MHz

5.2 บทสรุปและวิจารณ์

สรุปผลการทดลอง

จากผลการทดลองสามารถวัดสัญญานรูปคลื่นได้ ความถี่ตั้งแต่ 100 Hz จนถึง 1 MHz ซึ่งถ้าความถี่เกิน 1 MHz จะดูรูปคลื่นได้ลำบากเพราะรูปคลื่นมีความถี่มาก สัญญานที่วัดได้จะเหมือนกันกับสัญญานจากอินพุททุกประการ (มีความผิดพลาดน้อยมาก) สามารถวัดสัญญานได้ทุกรูปแบบ ที่เป็นอะนาลอก (Analog) การ ทดลองเราสามารถที่จะเก็บข้อมูลใน Buffer เมื่อต้องการนำสัญญานนั้นมาแสดงก็จะเห็นรูปคลื่นของสัญญานในขณะนั้นได้ทันที และจะมองเห็นการเพี้ยนของรูปสัญญานอินพุทได้

ในส่วนของ SPECTRUM ANALYSER ถ้าเราตั้งค่าย่านความถี่ที่ความถี่ต่ำๆ ความถูกต้องของ SPECTRUM ANALYSER จะมีความถูกต้องมากกว่าที่ย่านความถี่สูง เช่น ที่ความถี่ของสัญญาณ SINE WAVE 100 MHz ถ้าเราตั้งย่านความถี่ไว้ 250 KHz SPECTRUM ที่ได้จะอยู่ชิดขอบซ้ายสุดของ SCALE ซึ่งในจุดนี้จะมีความเพี้ยนมาก (ตามคุณสมบัติของ FFT) ถ้าตั้งย่านความถี่ 31 KHz ต่อช่อง SPECTRUM จะอยู่กลาง SCALE ซึ่งค่าของ SPECTRUM ที่ได้จะมีความแน่นอนกว่าเมื่อทำการทดสอบกับ AQUARE WAVE SIGNAL ก็จะได้ SPECTRUM ของ SQUARE WAVE SUGNAL ออกมาแต่จะมี AMPLITUDE ออกมาผิดพลาดเล็กน้อย

เครื่องที่เราสร้างขึ้นเราสามารถที่จะนำไปใช้เป็นเครื่องมือในการเรียนการสอน เป็นประโยชน์อย่างมากในการศึกษา ในเรื่องของ TIME DOMAIN AND FREQUENCY DOMAIN ทำให้ผู้ศึกษามีความเข้าใจในเรื่องของ TIME DOMAIN AND FREQUENCY DOMAIN ซึ่งสิ่งเหล่านี้เป็นพื้นฐานของการสื่อสารของระบบสื่อสารในปัจจุบัน

5.3 ปัญหาที่พบในการทำโครงงาน

1. อุปกรณ์หายาก คือ IC FLASH A/D เบอร์ CA3306 จะต้องสั่งจากต่างประเทศ ทำให้เสียเวลาในการทำงานประมาณเดือนกว่าๆ และ CYSTAL (X-TAL) ขนาด 30 MHZ หายาก

2. เนื่องจากวงจรออกแบบทำให้มีปัญหา ในบางครั้ง ทางด้านผลการทดลองที่ผิดบ้างถูกบ้างกว่าจะได้วงจรที่สมบูรณ์แบบก็ทำให้เสียเวลาไปมาก จึงไม่มีเวลาในการออกแบบวงจรพิมพ์ จึงต้องใช้สาย WIRE ทำให้ดูไม่เรียบร้อยนัก

3. การอ่านค่าจาก A/D ในบางครั้งยังผิดพลาดอยู่ เนื่องจากระบบการเดินสายไม่เป็นระเบียบเกิดการรบกวนกัน

4. การออกแบบวงจรเราใช้แผ่นวงจรเอนกประสงค์ทำให้เกิดการผิดพลาดได้ง่าย

5. เกิดการรบกวนจากสัญญาณภายในเครื่องคอมพิวเตอร์

6. ต้องใช้เวลาในการเขียนโปรแกรมมากเนื่องจากต้องศึกษาการเขียนโปรแกรมภาษา C และภาษาแอสแซมบลี้ เพิ่มเติม

7. การแปลง FAST FOURIER TRANFROM เพื่อแสดงแถบความถี่ ในบางครั้งเกิดความผิดพลาดได้

5.4 การแก้ปัญหา

ในส่วนของ HARDWARE ปัญหาส่วนใหญ่เกิดจากการเดินสายที่ไม่เป็นระเบียบเนื่องจากใช้แผ่นวงจรพิมพ์แบบเอนกประสงค์ ทำให้เกิดการรบกวนกันขึ้น จึงแก้ปัญหาดังนี้

1. วางอุปกรณ์ให้ชิดกันมากที่สุด เพื่อลดระยะทางของสายให้สั้นลง ทำให้การรบกวนลดลงไปได้บ้าง

2. ต่อตัวเก็บประจุคร่อม I.C. ทุกตัวเพื่อกันสัญญาณรบกวนจากแหล่งจ่ายไฟ

3. ในเครื่องคอมพิวเตอร์มีการออสซิลเลท ของอุปกรณ์บางตัวซึ่งจะมารบกวนสัญญาณอนาลอกทางอินพุทของเราได้ ดังนั้นที่จุดต่อสัญญาณอินพุทที่เข้ามายังวงจรควรใช้สาย SHIELD เพื่อที่จะลดสัญญาณรบกวนให้น้อยที่สุดเท่าที่จะทำได้

ในส่วนของ SOFTWARE แก้ปัญหาโดย

1. พยายามค้นคว้าหาตำราหรือเอกสารที่เกี่ยวของทางเทคนิคการเขียนโปรแกรมทั้งภาษา C และภาษาแอสแซมบลี้

2. การคำนวณหา SPECTRUM โดยการใช้ทฤษฎี FAST FOURIER TRANFROM ซึ่งเขียนด้วยภาษา C ในบ้างครั้งจะเกิดการผิดพลาดเนื่องมาจาก การกำหนดตัวแปรฟังค์ชัน FFT ไม่ถูกต้อง ต้องทำการเปลี่ยนแปลงจากตัวแปรแบบ FLOATING POINT มาเป็น DOUBLE FLOATING POINT ซึ่งจะทำให้การทำ FFT ถูกต้องมากขึ้น

3. พยายามแก้ปัญหาหรือเทคนิคต่างๆ ในการเขียนโปรแกรมทั้งการทำ FFT จากผู้รู้

5.5 ประโยชน์ที่ได้รับจากการทำโครงการ

1. ได้เครื่อง DIGITAL STORAGE OSCILLOSCOPE & FFT SPECTRUM ANALYZER ที่มีราคาถูก โดยใช้ต้นทุนในราคา 1,000 บาท ในขณะที่ถ้าเราซื้อที่มีขายอยู่ในท้องตลาดจะมีราคาหลายหมื่นบาท และเครื่องที่เราสร้างขึ้นก็จะมีประสิทธิภาพในการทำงานเช่นเดียวกัน

2. HARDWARE ที่สร้างขึ้นสามารถใช้ร่วมกับคอมพิวเตอร์ PC ที่เรามีใช้อยู่กันอย่างแพร่หลาย ทำให้สะดวกในการใช้งานเป็นอย่างยิ่ง อีกทั้งยังจะทำให้เครื่องคอมพิวเตอร์ใช้งานได้อย่างกว้างขวางขึ้น

3. เครื่องที่เราสร้างขึ้นสามารถใช้งานแทน STORAGE OSCILLOSCOPE และ SPECTRUM ANALYZER ได้

4. เครื่องที่เราสร้างขึ้สามารถที่จะใช้งานได้กับทั้งเครื่องรุ่น XT/AT และยังใช้จอภาพแบบ EGA, VGA และ MONOCROME อีกด้วย

5. เราสามารถนำเครื่องที่เราสร้างขึ้มาใช้ในการเรียนการสอนได้ เช่น ใช้แทน STORAGE OSCILLOSCOPE ซึ่งสามารถวัดสัญญาณได้สูงสุด 1 MHz และเครื่อง SPECTRUM ANALYZER ซึ่งแสดงแถบความถี่ได้สูงถึง 1 MHz เช่นเดียวกัน และยังศึกษาขบวนการแปลง FAST FOURIER TRANFROM ได้ อีกทั้งยังมีข้อดีกว่าที่ว่า สัญญาณที่เราวัดได้จะสามารถแสดงผลทางจอภาพและทางเครื่องพิมพ์ได้ทันที ไม่ต้องเสียเวลาวาด ซึ่ง OSCILLOSCOPE ธรรมดาจะไม่สามารถทำได้

6. ได้รับความรู้กว้างขวางขึ้นอย่างมาก ในการทำโครงการนี้ทั้งในส่วนของ HARDWARE และ SOFTWRAE

7. เครื่องต้นแบบที่เราสร้างขึ้นมาสามารถนำออกแสดงเพื่อสร้าวชื่อเสียงให้กับคณะ และสถาบันได้

8. นักศึกษารุ่นต่อไป จะได้มีโอกาสศึกษาและพัฒนาให้ดีขึ้นต่อไป

5.6 แนวทางในการพัฒนา

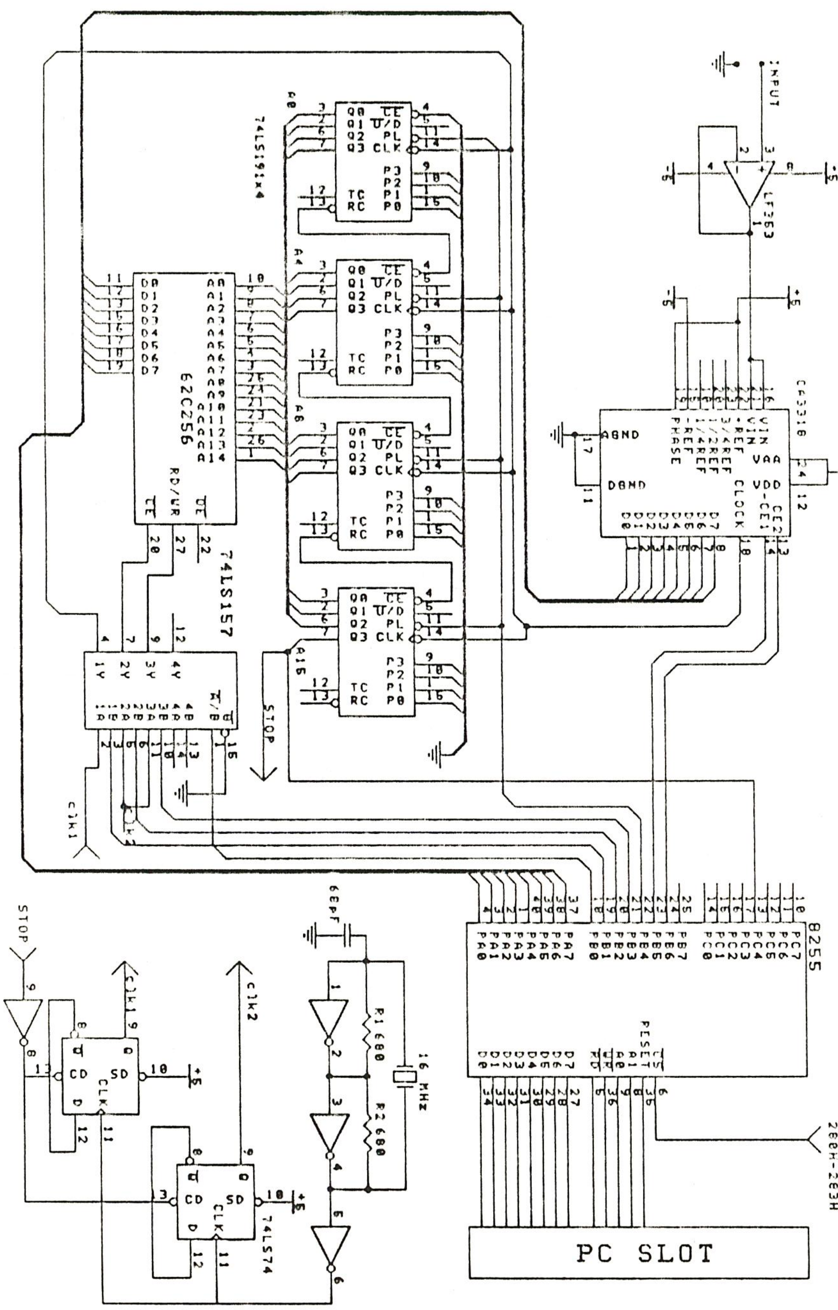
1. พัฒนาในส่วนของ HARDWARE ให้สามารถวัดสัญญาณทางด้าน INPUT ที่มีค่าแรงดัน ได้สูง โดยใช้วงจร DROP แรงดันอินพุท ที่สัมพันะกับส่วนของ SOFTWARE แสดงผล

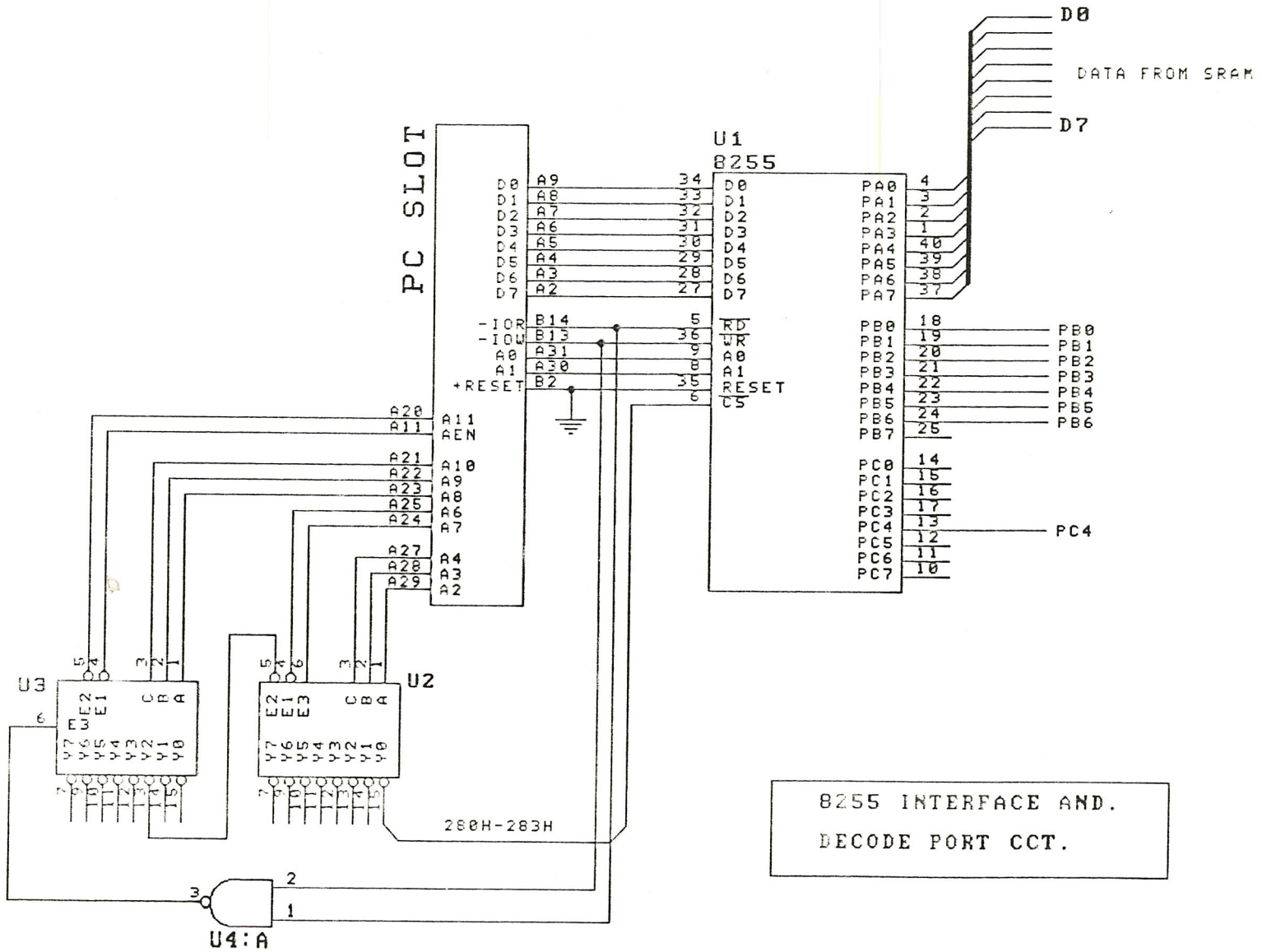
2. พัฒนาให้สามารถวัดความถี่ให้สูงขึ้กว่านี้โดยการเปลี่ยน CLOCK SAMPLING RATE ให้สูงขึ้น 15 MHz ซึ่งจะใช้ความถี่ CYSTAL 30 MHz / 2

3. พัฒนาให้สามารถวัดสัญญาณได้หลายๆ ช่องมากขึ้น โดยการเพิ่ม FLASH A/D และเปลี่ยนแปลง SOFTWARE

4. พัฒนาให้สามารถใช้งานได้กับ PRINTER ทุกแบบไม่ว่าจะเป็นแบบ 24 หัวเข็ม หรือ LQ หรือ 9 หัวเข็ม ก็ตาม โดยการเพิ่มFRONT ให้กับโปรแกรม เพื่อเรียกใช้งาน PRINTER ได้

ภาคผนวก


74LS157
62C256
8255
PC SLOT
74LS74
16 MHz
STOP
INPUT
280H-283H


PC SLOT
U1
8255
D0
DATA FROM SRAM
D7
D0 A9
D1 A8
D2 A7
D3 A6
D4 A5
D5 A4
D6 A3
D7 A2
-IOR B14
-IOW B13
A0 A31
A1 A30
+RESET B2
A20
A11
A21
A22
A23
A25
A24
A27
A28
A29
A11 AEN
A10
A9
A8
A6
A7
A4
A3
A2
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
RESET
CS
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
U3
U2
280H-283H
U4:A
8255 INTERFACE AND.
DECODE PORT CCT.

```
/*
    Turbo C to make the Digital storage oscilloscope & FFT spectrum analyzer.

    This project content of hardware & software.

        - Hardware consis of flash A/D sampling rate 8 MHz.

        - Software writen with TURBO C version 2.

        - Compile with TCC .

        - tcc -mc store graphics.lib herc.obj egavge.obj litt.obj
                                                                    */

#include <dos.h>
#include <math.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <stdarg.h>
#include <graphics.h>
#include <fcntl.h>
#include <io.h>
#include <bios.h>
```

```
void Initg (void);
void Main_Window (void);
void Main_Win1 (void);
void Read_kbd (void);
void Enter (void);
void Read_AD (void);
void Files_Ser (void);
void Get_Name (void);
void Display (void);
void Setup (void);

int       x= 8, y= 13, x1, y1, count= 0, idx= 0, a, acpage=0, vipage=0, res1 ;
char      menuf= 0;
char      count1 []= { 0, 3, 0, 0, 0, 1, 0, 3, 0, 2, 0, 1 } ;
int       GraphDriver, GraphMode  ;
int       maxx, maxy, handle, stept = 1, stept1=0, stepf=1;
char      ascii, scancode ,in;
char      image [400], image0 [1000],  image1 [4800], image2 [2000], image3 [2400] ;
char      image4 [4800], image5 [4000], image6 [2400], image8 [3600],imagek [1000] ;
unsigned  size;
void      far *image9, *image10, *image11, *image7, *image14 ;
char      buffer[10000], xx[500] ;
char      *point []= { image1, image2, image3, image4, image5, image6 } ;
char      fname[]= { "TEST.SCP              " };
float     xreal[1200], ximag[1200], result[1200] ;
int       timediv[]={ 1, 4, 20, 100, 400, 1000, 4000 };
```

```
char        *ctimediv[]= { "5 uS", "20 uS", "0.1 mS", "0.5 mS", "2 mS", "5 mS", "20 mS" };
char        *fdiv[]= { "","250 KHz", "125 KHz","","62 KHz","","","","31 KHz" } ;
int         cfdiv[]= { 1, 2, 4, 8, 16, };

/*
    Main program
                */

void main ( )

{

    outportb (0x283, 0x98);
    Initg ();
    Main_Window ();
    closegraph ();
}

/*
    Auto detect card & Initgraphics mode
                                    */

void Initg (void)

{
    int     a ;
```

```
/*    if  (peekb ( 0x40,0x49 )== 7)
    {   GraphDriver= 7 ;
        GraphMode= 0 ;    } */

       GraphDriver= 3 ;                    /* HERCULIST Driver */
        GraphMode= 1 ;                     /* 2 Color , 720 X 348 ,2 Page */

    if (registerbgidriver(EGAVGA_driver) < 0) exit (1); /* Initialized BGI Driver */
    if (registerbgifont(small_font) < 0) exit (1);

    initgraph (&GraphDriver, &GraphMode, ""); /* Initialized */
    maxx= getmaxx ();                   /* Get MAX_X */
    maxy= getmaxy ();                   /* Get MAX_Y */
    setactivepage (1);                  /* Set Visual Page to 1 */

    settextstyle (SMALL_FONT, HORIZ_DIR, 4);

    setfillstyle (SOLID_FILL,11);
    bar (0, 0, 200, 10);
    getimage (0, 0, 150, 10, imagek);
    size= imagesize (0, 0, 131, 130);
    image7= malloc (size);
    if  (image7== NULL)
        exit (1);
    size= imagesize (0, 0, 300, 200);
```

```
image14= malloc (size);
if  (image14== NULL)
    exit (1);

setfillstyle (SOLID_FILL,14);              /* Set BAR For Inverse MENU */
bar (0, 0, 131, 59);
getimage (0, 0, 131, 59, image7);
setfillstyle (SOLID_FILL,7);               /* Set BAR For Inverse BAR */
bar (0, 0, 130, 11);
getimage (0, 0, 118, 10, image0);
cleardevice ();
setactivepage (1);

setcolor (6);                              /* MENU 1 */
rectangle (0, 0, 129, 57);
rectangle (0, 0, 130, 58);
rectangle (0, 0, 131, 58);
setcolor (6);
outtextxy (0, 5,  " LOAD FILES      F3");
outtextxy (0, 17, " SAVE FILES      F4");
outtextxy (0, 29, " CLEAR SCREEN");
outtextxy (0, 41, " QUIT          ALT+X");
putimage (0, 0, image7, 1);
getimage (0, 0, 131, 58, image1);
cleardevice ();
setactivepage (1);
```

```
setcolor (6);                          /* MENU 2 */
rectangle (0, 0, 129, 21);
rectangle (0, 0, 130, 21);
rectangle (0, 0, 131, 22);
setcolor (6);
outtextxy (0, 5, " ENTER TO START");
putimage (0, 0, image7, 1);
getimage (0, 0, 131, 22, image2);
cleardevice ();
setactivepage (1);

setcolor (6);                          /* MENU 3 */
rectangle (0, 0, 129, 33);
rectangle (0, 0, 130, 33);
rectangle (0, 0, 131, 34);
setcolor (6);
outtextxy (0, 5, " DISPLAY BUFFER");
outtextxy (0, 17, " CLEAR BUFFER");
putimage (0, 0, image7, 1);
getimage (0, 0, 131, 34, image3);
cleardevice ();
setactivepage (1);

setcolor (6);                          /* MENU 4 */
rectangle (0, 0, 129, 57);
```

```
rectangle (0, 0, 130, 57);
rectangle (0, 0, 131, 58);
setcolor (6);
outtextxy (0, 5,  " TIME/DIV        F5");
outtextxy (0, 17, " VOLT/DIV        F6");
outtextxy (0, 29, " SLOPE           F7");
outtextxy (0, 41, " ZOOM            F8");
putimage (0, 0, image7, 1);
getimage (0, 0, 131, 58, image4);
cleardevice ();
setactivepage (1);

setcolor (6);                       /* MENU 5 */
rectangle (0, 0, 129, 45);
rectangle (0, 0, 130, 45);
rectangle (0, 0, 131, 46);
setcolor (6);
outtextxy (0, 5,  " PRINTER No_  ");
outtextxy (0, 17, " PRINT SCREEN ");
outtextxy (0, 29, " PRINT BUFFER ");
putimage (0, 0, image7, 1);
getimage (0, 0, 131, 46, image5);
cleardevice ();
setactivepage (1);

setcolor (6);                       /* MENU 6 */
```

```
rectangle (0, 0, 129, 33);
rectangle (0, 0, 130, 33);
rectangle (0, 0, 131, 34);
setcolor (6);
outtextxy (0, 5,  " FREQUENCY/DIV ");
outtextxy (0, 17, " FFT. SPECTRUM ");
putimage (0, 0, image7, 1);
getimage (0, 0, 131, 34, image6);
cleardevice ();
setactivepage (1);

setcolor (0);                          /* SCOPE BOX MONITOR */
setfillstyle (SOLID_FILL,15);
bar (0,0,560,200);
rectangle (0,0, 560,128);
rectangle (0,0, 561,129);
rectangle (0,0, 562,129);
line (0,32+32, 560, 32+32);
line (280,0, 280,128);
setlinestyle (USERBIT_LINE,0x0303,THICK_WIDTH);
line (0,32+32, 560, 32+32);
line (280,0, 280,128);
setlinestyle (DOTTED_LINE,0,NORM_WIDTH);
x1= 40 ;
for (a= 13 ; a > 0 ; a- , x1+= 40)
    line (x1, 0, x1, 128);
```

```
y1= 32 ;
for (a= 3 ; a > 0 ; a-- , y1+= 32)
    line (0, y1, 560, y1);
size=imagesize (0, 0, 562, 129);
image9= malloc (size);
if (image9== NULL)
{   closegraph ();
    exit (1);    }
getimage (0, 0, 562, 129, image9);
setlinestyle (SOLID_LINE,0,NORM_WIDTH);
cleardevice ();
setactivepage (1);

setcolor (0);                          /* SCOPE BOX MONITOR 2 */
setfillstyle (SOLID_FILL,9);
bar (0,0,600,32);
rectangle (0,0, 600,12);
rectangle (0,0, 601,13);
rectangle (0,0, 602,13);
size=imagesize (0, 0, 602, 13);
image10= malloc (size);
if (image10== NULL)
{   closegraph ();
    exit (1);    }
getimage (0,0,602,13,image10);
cleardevice ();
```

```
setactivepage (1);

setcolor (0);                          /* SCOPE BOX MONITOR3 */
setfillstyle (SOLID_FILL,15);
bar (0,0,560,200);
rectangle (0,0, 512,120);
rectangle (0,0, 513,121);
rectangle (0,0, 514,121);
setlinestyle (DOTTED_LINE,0,NORM_WIDTH);
x1= 65;
for (a= 0 ; a <7 ; a++ , x1+= 64)
    line (x1, 0, x1, 128);
y1= 30 ;
for (a= 0 ; a < 3 ; a++ , y1+= 30)
    line (0, y1, 560, y1);
size=imagesize (0, 0, 514, 121);
image11= malloc (size);
if (image11== NULL)
{   closegraph ();
        exit (1);    }
getimage (0,0,514,121,image11);
setlinestyle (SOLID_LINE,0,NORM_WIDTH);
cleardevice ();
setactivepage (1);

setfillstyle (SOLID_FILL,15);          /* GET IMAGE For MENU */
```

```
    bar (0, 0, 35, 11);
    getimage (0, 0, 55, 11, image);
    putimage (0, 0, image, 1);

    bar (0, 0, maxx, 10);                    /* Get IMAGE for MENU */
    getimage (0, 0, maxx, 10, image8);
    cleardevice ();
    setactivepage (0);
}

/*
    Display menu on screen with 2 page and main menu to accress function
                                                                    */

void Main_Window (void)

{
    setvisualpage (1);                       /* Set Visual Page to 1 */
    Main_Win1 ();
    setvisualpage (0);
    setactivepage (1);
    Main_Win1 ();
    setactivepage (0);
    getimage (x, 0, x+131, y+100, image7);
    putimage (x, y, image, 1);
```

```
    x1= x+5;
    y1= y+25 ;

/* Set loop display bar menu */

    for (;;)
    {
        Read_kbd ();                          /* Get Key_Board */
        switch  (scancode)
        {
            case    1:                        /* Esc Key Press */

                if  (menuf==1)
                {   menuf=0;
                    putimage (x, 0, image7, 0);
                    putimage (x, y, image, 1);   }

                break;

            case    28:                       /* Enter */

                if   (menuf==0)
                {   menuf=1;
                    putimage (x, y+20, point[count], 0);
                    putimage (x1, y1,image0, 1);    }
                else
```

```
        Enter ();

    break;

case    77:                         /* Rigth Key Press */
    if  (count== 5)
        break;
    putimage (x, 0, image7, 0);
    count++ ;
    x+= 52;
    getimage (x, 0, x+131, y+100, image7);
    putimage (x, y, image, 1);
    idx+= 2 ;
    x1= x+5 ;
    y1= y+25+(count1[idx]*12);

    if  (menuf==1)
    {   putimage (x, y+20, point[count], 0);
        putimage (x1, y1,image0, 1);   }

    break;

case    75:                         /* Left Key Press */
    if  (count== 0)
        break;
    putimage (x, 0, image7, 0);
```

```
        count-- ;
        x-= 52;
        getimage (x, 0, x+131, y+100, image7);
        putimage (x, y, image, 1);
        idx-= 2 ;
        x1= x+5 ;
        y1= y+25+(count1[idx]*12);

        if  (menuf==1)
        {   putimage (x, y+20, point[count], 0);
            putimage (x1, y1,image0, 1);    }

        break;

case    80:                     /* DOWN Key Press */

        if  (menuf==0)
            break;

        if  (count1[idx]== count1[idx+1])
        {   putimage (x1, y1, image0, 1);
            y1= y+25 ;
            count1[idx]= 0 ;
            putimage (x1, y1, image0, 1);   }

        else
```

```
{   putimage (x1, y1, image0, 1);
    y1+= 12 ;
    count1[idx]++ ;
    putimage (x1, y1, image0, 1);   }

break;

case   72:                    /* UP Key Press */

if  (menuf==0)
    break;

if  (count1[idx]== 0)
{   putimage (x1, y1, image0, 1);
    y1= y+25+(count1[idx+1]*12);
    count1[idx]= count1[idx+1]  ;
    putimage (x1, y1, image0, 1);  }

else
{   putimage (x1, y1, image0, 1);
    y1-= 12 ;
    count1[idx]- ;
    putimage (x1, y1, image0, 1);  }

break;
```

```
        } /* END LOOP CASE */

/*          if  (scancode==59)
              break; */

    }   /* END LOOP Key_Board Recive */

}   /* END Function */

/*
    Draw menu on screen with active page
                                        */

void Main_Win1 (void)

{
    if  (GraphDriver== 7)
    {   setfillstyle (INTERLEAVE_FILL,14);
        bar (0,28,maxx,maxy-15);    }

    else
    {   setfillstyle (SOLID_FILL,7);
        bar (0,26,maxx,maxy-13);    }

    setcolor (6);
```

```
    outtextxy (0, 1, " DIGITAL STORAGE OSCILLSCOPE & FFT. POWER SPECTRUM ANALYZER.
Version 1.0 ");
    putimage (0, 1, image8, 1);
    setcolor (15);
    outtextxy (0, 14, " FILES   READ   DISP   SETUP   PRINT   SPECT ");
    setcolor (14);
    outtextxy ( 0, maxy-11, " F1:HELP F2:READ/STORE F3:LOAD F4:SAVE F5:TIME F6:VOLT
F7:SLOPE F8:ZOOM");
    putimage (0, maxy-11, image8, 1);
    setcolor (15);
    rectangle (0, 0, maxx, maxy);
    rectangle (0, 12, maxx, 25);
    rectangle (0, maxy-12, maxx, maxy);
    putimage ((maxx-562)/2,35,image9,0);
    putimage ((maxx-602)/2,35+134,image10,0);
    putimage ((maxx-602)/2,35+154+132,image10,0);
    setcolor (1);
    outtextxy ((maxx-610)/2, 30,"  5");
    outtextxy ((maxx-610)/2, 30+32," 2.5");
    outtextxy ((maxx-610)/2, 30+64,"  0");
    outtextxy ((maxx-610)/2, 30+96,"-2.5");
    outtextxy ((maxx-610)/2, 30+127," -5");
    outtextxy ((maxx-612)/2, 30+154," 15");
    outtextxy ((maxx-612)/2, 30+154+30,"  0");
    outtextxy ((maxx-612)/2, 30+154+60," -15");
    outtextxy ((maxx-612)/2, 30+154+90," -30");
```

```
    outtextxy ((maxx-612)/2, 30+154+120," -45");

    if  (GraphDriver==7)
        setcolor (0);
    else
        setcolor (15);

    outtextxy ((maxx-602)/2 , 35+134+1, " DIGITAL STORAGE OSCILLOSCOPE : TIME/DIV=5 uS
VOLT/DIV=2.5 V");
    outtextxy ((maxx-602)/2 , 35+154+133, " RESOLUTION=15625 Hz,500 kHz/DIV : MAGNITUDE=3
dB/OCTAVE (dB VP^2)");
    putimage ((maxx-562)/2,35+154,image11,0);
}

/*
    Read keyboard function use bios function 16h.
    Inline assembly.
                        */

void Read_kbd ( void )

{
union REGS  reg;
    reg.h.ah = 0;
    int86(0x16,&reg,&reg);
    scancode=reg.h.ah;
```

```
    ascii=reg.h.al;
}

/*
   Enter key press entry the other functions
                                        */

void Enter (void)

{
/*   int     a,b,c; */
   switch  (count)
   {
      case   1:                          /* READ A/D */
         putimage (x, 0, image7, 0);
         Read_AD ();
         putimage (x, y, image, 1);
         putimage (x, y+20, point[count], 0);
         putimage (x1, y1,image0, 1);
         break;

      case   0:                          /* FLIE SERVICE */
         Files_Ser ();
         break;

      case   2:
```

```
            Display ();
            break;

        case    5:
            Spectrum ();
            break;

        case    3:
            Setup ();
            break;

        case    4:
            Plotter ();
            break;

    }
}

/*
    Function to read data from A/D Converter.
    Display on screen both ega/vga & herculis card.
                                                    */

void Read_AD (void)

{
```

```
    unsigned    char     b, ex;
    unsigned    int      a,c,d,e;
    float       max, min;

    setfillstyle (SOLID_FILL,11);
    if  (GraphDriver==7)                    /* Detect COLOR */
        setcolor (0);

    else
        setcolor (1);

for (;;)
{

    acpage= acpage ^ 1;                     /* Set ACTIVE PAGE */
    setactivepage (acpage);
    putimage ((maxx-562)/2,35,image9,0);

    outportb (0x283,0x98);                  /* Enable A/D To Read */
    outportb (0x281,0xcc);
    outportb (0x281,0xdc);

    for (;;)
    {   b=inportb (0x282);                  /* Check Successive */
        b=b&0x10;
        if (b==0x10)
```

```
    break;  }

outportb (0x281,0xac);                  /* Copy From RAM to Buffer */
outportb (0x280,0xbd);

for (a=0;a<8000;a++)
{   outportb (0x281,0xb9);
    b=inportb (0x280);
    outportb (0x281,0xbd);
    outportb (0x281,0xbf);
    outportb (0x280,0xbd);
    b=b&0x3f;
    b<<=1;
    buffer[a]=b;    }

d=(maxx-560)/2;
moveto (d, 35+64);                      /* Plot ON SCREEN */
a=0;
max=-100.;
min=100.;
for (e=1 ; e <= 560 ; e++,d++,a+=stept)
{
   c=35+128-buffer[a];
   lineto (d,c);
}
vipage= vipage ^ 1;                     /* SET VISUAL PAGE */
```

```
    setvisualpage (vipage);

    if (bioskey(1)!=0)
    {
        Read_kbd ();
        if (scancode==60)
            break;
    }
}
        getimage (x, 0, x+131, y+100, image7);
        setactivepage (acpage);
        setcolor (15);
}

/*
    Files service  routine as  save , load , clear , quit
                                                        */

void Files_Ser (void)

{
    int     a,c,d,e;

    switch  (count1[idx])
    {
       case    0:
```

```
Get_Name ();                        /* GET KEY */

if  (scancode==1)                   /* Esc Check */
    break;

handle = _open(fname,O_RDWR);           /* Load FILE */
_read(handle,buffer,560);
_close(handle);
putimage (x, 0, image7, 0);             /* Restore Screen */
if  (GraphDriver==7)                /* Detect COLOR */
    setcolor (0);
else
    setcolor (1);

d=(maxx-560)/2;
moveto (d, 35+64);                  /* Plot ON SCREEN */
a=0;
for (e=1 ; e <= 560 ; e++,d++,a+=stept)
{
    c=35+128-buffer[a];
    lineto (d,c);   }

getimage (x, 0, x+131, y+100, image7);   /* Save SCREEN IMAGE */
putimage (x, y, image, 1);
putimage (x, y+20, point[count], 0);
```

```
    putimage (x1, y1,image0, 1);
    break;

case    1:

    Get_Name ();                    /* Get KEY */

    if  (scancode==1)               /* Esc key Check */
        break;

    handle=_creat(fname,FA_ARCH);           /* Save FILE */
    _write(handle,buffer,560);
    _close(handle);
    break;

case    2:

    putimage (x, 0, image7, 0);
    putimage ((maxx-562)/2,35,image9,0);
    getimage (x, 0, x+131, y+100, image7);
    putimage (x, y, image, 1);
    putimage (x, y+20, point[count], 0);
    putimage (x1, y1,image0, 1);
    break;

case    3:
```

```
            closegraph ();
            printf (" KMIT'L.\n\n");
            exit (0);
    }

}

/*
    Get input from keyboard press & display in graphics mode
                                                        */

void Get_Name (void)

{
    int     num, x, y, keyf;

    setviewport (100, 50, 400, 200, 1);          /* Adj View port */
    getimage (0, 0, 299, 100, image14);
    setcolor (0);
    setfillstyle (SOLID_FILL,11);                /* Fill Menu */
    bar (0,0,290,30);
    outtextxy (0, 11, " Enter name :");
    rectangle (0, 0, 290,30);
    rectangle (0, 0, 291,31);
    rectangle (0, 0, 292,31);
```

```
setcolor (0);

num= 0;
x= 13*6;
y= 11;
keyf= 0 ;
setcolor (0);

outtextxy (13*6, 11, "_");

for (;;)                              /* Loop KEY CHECK */
{
   Read_kbd ();

   if  (scancode==1)                  /* Esc key Press */
       break;

   if  (scancode==28)                 /* Enter Key Press */
       break;

   if  (ascii==0)                     /* Allow Key Press */
       continue;

   if  (scancode==14)                 /* BACK SPACE Key Press */
   {   if  (num!=0)
       {
```

```
            putimage(x,y,imagek,0);
            num--;
            fname[num]= 0x5f ;
            fname[num+1]= 0 ;
            outtextxy(13*6, 11, fname);
            continue;   } }

        else                                    /* SAVE DATA TO BUFFER */
        {   if  (num==20)
                continue;
            fname[num]= ascii;
            num++;
            fname[num]= 0x5f ;
            fname[num+1]= 0 ;
            putimage(x,y,imagek,0);
            outtextxy(13*6, 11, fname);
            keyf= 1;   }
    }

    if  (keyf!=0)                               /* NO Key Press */
        fname[num]= 0;

    putimage (0, 0, image14, 0);
    setviewport (0, 0, maxx, maxy, 1);
}
```

```
/*
    Display buffer on screen function.
                                        */

void Display (void)

{
    int     a, c, d, e ;

    switch  (count1[idx])
    {
        case    1:                          /* CLEAR BUFFER */

            for (e=0; e <=560; e++)
                buffer[e]=0;

            putimage (x, 0, image7, 0);
            putimage ((maxx-562)/2,35,image9,0);
            getimage (x, 0, x+131, y+100, image7);
            putimage (x, y, image, 1);
            putimage (x, y+20, point[count], 0);
            putimage (x1, y1,image0, 1);
            break;

        case    0:                          /* Display Buffer */
```

```
            if  (GraphDriver==7)                    /* Detect COLOR */
                setcolor (0);
            else
                setcolor (1);
            putimage (x, 0, image7, 0);
            putimage ((maxx-562)/2, 35, image9, 0);

            d=(maxx-560)/2;
            moveto (d, 35+64);                      /* Plot ON SCREEN */
            a=0;
            for (e=1 ; e <= 560 ; e++,d++,a++)
            {
               if  (buffer[a]<-64)
                   buffer[a]=-64;
               if  (buffer[a]>64)
                   buffer[a]=64;
               c=35+128-buffer[a];
               lineto (d,c);   }

            getimage (x, 0, x+131, y+100, image7);  /* Save SCREEN IMAGE */
            putimage (x, y, image, 1);
            putimage (x, y+20, point[count], 0);
            putimage (x1, y1,image0, 1);
            break;
        }
}
```

```
/*  Setup function to setup system
        Time / division
        Volt / division
        Slop
        Zoom
              */

void Setup (void)

{
   static  int     x= 5, y= 28 ;
   static  char    a= 0;
   switch (count1[idx])
   {
      case    0:

          setviewport (300, 50, 450, 200, 1);          /* Adj View port */
          getimage (0, 0, 150, 150, image14);
          setcolor (0);
          setfillstyle (SOLID_FILL,11);                /* Fill Menu */
          bar (0,0,129,120);
          rectangle (0, 0, 129,120);
          rectangle (0, 0, 130,119);
          rectangle (0, 0, 131,119);
          rectangle (0, 0, 131, 22);
```

```
outtextxy (0, 10, "   TIME/DIV    ");
outtextxy (0, 30, "   5 - uS   ");
outtextxy (0, 42, "   20 - uS   ");
outtextxy (0, 54, "   .1 - mS   ");
outtextxy (0, 66, "   .5 - mS   ");
outtextxy (0, 78, "   2 - mS   ");
outtextxy (0, 90, "   5 - mS   ");
outtextxy (0, 102, "   20 - mS   ");

if (a > 6)
{   x= 5;
    y= 28;
    a=0;    }

putimage (x, y, image0, 1);

for (;;)
{
    Read_kbd ();
    switch  (scancode)
    {
        case    72:             /* Up key press */

            if (a != 0)
            {   a--;
```

```
            putimage (x, y, image0, 1);
            y-=12;
            putimage (x, y, image0, 1); }
        break;

    case    80:             /*  Down key press */

        if (a != 6)
        {   a++;
            putimage (x, y, image0, 1);
            y+=12;
            putimage (x, y, image0, 1); }
        break;

    case    28:

        break;

    case    1:

        break;

}

if  (scancode==1)
    break;
```

```
            if  (scancode==28)
                break;

        }
        stept= timediv[a];
        stept1=a;
        putimage (0, 0, image14, 0);
        setviewport (0, 0, maxx, maxy, 1);
        break;

    }

}

/*      GPRINTF: Used like PRINTF except the output is sent to the      */
/*      screen in graphics mode at the specified co-ordinate.           */

int gprintf( int *xloc, int *yloc, char *fmt, ... )

{
  va_list  argptr;              /* Argument list pointer */
  char str[140];                /* Buffer to build sting into      */
  int cnt;                              /* Result of SPRINTF for return */

  va_start( argptr, format );           /* Initialize va_ functions */
```

```
  cnt = vsprintf( str, fmt, argptr );/* prints string to buffer */
  outtex ค-~+@ ~~~~~~~~~~~~~~~~~ฐ~~~า
Send string in graphics mode */
  *yloc += textheight( "H" ) + 2;       /* Advance to next line        */

  va_end( argptr );                          /* Close va_ functions          */

  return( cnt );                        /* Return the conversion count  */

}

Spectrum ()

{
   int     a, b, c, d ;
   float   res, p=3.141592, max, min;

   putimage ((maxx-562)/2,35+154,image11,0);
   setcolor (9);
   outtextxy ((maxx-562)/2, 35+160, "  PLEASE WAIT");
   for (d=0; d<512; d++)
   {
      xreal[d]=buffer[d]/12;
      res=(1.-cos(2*p*(d+1)/512));
      xreal[d]=xreal[d]*res;
      ximag[d]=0; }
```

```
fft (512, 9);
max=-1000.;
min=1000.;
for (a=0; a<256; a++)
{
    result[a]=(sqrt(xreal[a]*xreal[a]+ximag[a]*ximag[a]))/256;
    res=result[a]*result[a];
    if  (result[a]!=0)
        result[a]=5*log10(res);
    else
        result[a]=result[a-1];
    if  (result[a]<-45)
        result[a]=-45;
    if  (max<result[a])
        max=result[a];
    if  (min>result[a])
        min=result[a];
}

putimage ((maxx-562)/2,35+154,image11,0);
setcolor (4);
moveto  ((maxx-562)/2, 35+154+30);
b=(maxx-562)/2;
for (a=0; a<256; a++,b+=2)
{   c=result[a]*2;
```

```
        lineto (b, 35+154+30-c);   }
    sprintf (xx, "MAX=%.1f dB MIN=%.1f dB", max, min);
    bar (430, 35+154+133, 580, 35+154+142);
    outtextxy (440, 35+154+133, xx);
    acpage^=1;
    setactivepage (acpage);
    putimage ((maxx-562)/2,35+154,image11,0);
    moveto  ((maxx-562)/2, 35+154+30);
    b=(maxx-562)/2;
    for (a=0; a<256; a++,b+=2)
    {   c=result[a]*2;
        lineto (b, 35+154+30-c);   }
    sprintf (xx, "MAX=%.1f dB MIN=%.1f dB", max, min);
    bar (430, 35+154+133, 580, 35+154+142);
    outtextxy (440, 35+154+133, xx);
    acpage^=1;
    setactivepage (acpage);
}

/*
    Calculate Fast Fourier Transform.
    Input data in store in xreal & ximag.
                                        */

fft (int n, int nu)
```

```
{
    int     n2,nu1,k,l,i,m;
    float   p,c,s,arg,treal,timag;

    n2=n/2;
    nu1=nu-1;
    k=0;

    for ( l=1; l <= nu; l++)
    {

        for (;;)
        {
            for ( i=1; i <= n2; i++)
            {   m= k/pow(2,nu1);
                p= ibtr (m,nu);
                arg= 6.283185*p/n;
                c= cos (arg);
                s= sin (arg);
                treal= xreal[k+n2]*c+ximag[k+n2]*s;
                timag= ximag[k+n2]*c-xreal[k+n2]*s;
                xreal[k+n2]= xreal[k]-treal;
                ximag[k+n2]= ximag[k]-timag;
                xreal[k]= xreal[k]+treal;
                ximag[k]= ximag[k]+timag;
                k++;   }
```

```
            k=k+n2;
            if  (k >= n)
                break;  }

        k=0;
        n2=n2/2;
        nu1=nu1-1;  }

    for (k=0; k < n; k++)
    {   i=ibtr (k,nu);
        if ( i > k )
        {   treal= xreal[k];
            timag= ximag[k];
            xreal[k]= xreal[i];
            ximag[k]= ximag[i];
            xreal[i]= treal;
            ximag[i]= timag;    }   }

}

/*
    Bit reversing funtion for FFT
                                */

ibtr ( int j, int nu )
```

```
{
    int     i,j1,j2,k;

    j1=j;
    k=0;

    for (i=1; i <= nu ;i++)
    {   j2=j1/2;
        k=k*2+(j1-2*j2);
        j1=j2;    }

    return (k);
}

/*
    Ploter functions
                    */
Plotter ()

{
    int     a, b, c;
    float   max, min;

    Plot ("SP1;LT-1;VS5;PA;");
    Plot ("PU0,0;PD10000,0,10000,8000,0,8000,0,0;");
```

```
Plot ("SP2;PU1080,7200;PR;PD7840,0,0,-3000,-7840,0,0,3000;");   /* BOX 1 */
Plot ("LT2,1;");
for (a=0; a<13; a++)
    Plot ("PU560,-3000;PD0,3000;");
Plot ("PU560,-750;");
for (a=0; a<3; a++)
    Plot ("PD-7840,0;PU7840,-750;");

Plot ("LT-1;");
Plot ("PU-7840,-800;PD7680,0,0,-3000,-7680,0,0,3000;");          /* BOX 2 */
Plot ("LT2,1;");
for (a=0; a<7; a++)
    Plot ("PU960,-3000;PD0,3000;");
Plot ("PU960,-750;");
for (a=0; a<3; a++)
    Plot ("PD-7680,0;PU7680,-750;");

Plot ("VS42;SP3;PA;LT-1;PU1080, 5700;");
b=1080;
for (a=0; a<560; a++, b+=14)
{
/*      sprintf (xx, "PD%d,%d;", b, buffer[a]*23+5700);   */
    sprintf (xx, "PD%d,%d;", b, buffer[a]*23+4200);
    Plot (xx);  }
```

```
    Plot ("PU1080, 2650;");
    b=1080;
    for (a=0; a<256; a++, b+=30)
    {
        c=result[a]*50+2650;
        sprintf (xx, "PD%d,%d;", b, c);
        Plot (xx);  }

    Plot ("SP4;SI0.2,0.3;");
    Plot ("PA;PU0,8030;LBDIGITAL STORAGE OSCILLOSCOPE & POWER SPECTRUM ANALYZER
KMIT'L~");
    Plot ("PU500,7150;LB   5~PU500,6400;LB 2.5~PU500,5650;LB   0~PU500,4900;LB-
2.5~PU500,4150;LB  -5~");
    Plot ("PU500,3350;LB  15~PU500,2600;LB   0~PU500,1850;LB -15~PU500,1100;LB -
30~PU500,350;LB -45~");
    Plot ("PU1080,7500;LBDIGITAL STORAGE OSCILLOSCOPE : ");
    max=-1000.;
    min=1000.;
    for (a=0; a<256; a++)
    {
        if  (max<buffer[a])
            max=buffer[a];
        if  (min>buffer[a])
            min=buffer[a];
    }
    max=max/12;
```

```
    min=min/12;
    sprintf (xx, "MAX-MIN~");
    Plot (xx);
    Plot ("PU1080,7250;LBTIME/DIV=5 uS    VOLT/DIV=2.5 Volt~");
    Plot ("PU1080,3700;LBPOWER SPECTRUM ANALYZER : ");
    max=-1000.;
    min=1000.;
    for (a=0; a<256; a++)
    {
        if  (max<result[a])
            max=result[a];
        if  (min>result[a])
            min=result[a];
    }
    max=max;
    min=min;
    sprintf (xx, "MAX=+%.1f dB MIN=%.1f dB (VP^2)~", max, min);
    Plot (xx);
    Plot ("PU1080,3450;LBRESOLUTION=15625 Hz,500 kHz/DIV  MAGNITUDE=3 dB/OCTIVE
(VP^2)~");
    Plot ("PU1080,200;LB0~");
    Plot ("PU8500,200;LB 4MHz~PU0,0");
}

Plot (char  *xx)
```

```
{
    int     z;
    z=0;
    for (;;)
    {
        if  (xx[z]==0)
            break;
        biosprint(0,xx[z],0);
        z++;
    }
}
```

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้ สามารถสำเร็จลุล่วงลงได้ นอกจากการร่วมมือกัน การช่วยเหลือกันการร่วมกันแก้ปัญหา การทำงานในส่วนที่ตนเองรับผิดชอบ อย่างเต็มที่เต็มกำลังความสามารถของกลุ่มนักศึกษาโครงงานนี้แล้วยังมีท่านคณาจารย์และเพื่อนๆ ที่ให้กำลังใจและคำแนะนำที่เป็นประโยชน์ต่างๆ และความอนุเคราะห์ในเรื่องของเครื่องมืออุปกรณ์ ที่ใช้ในการทดลองตลอดเวลาที่ทำโครงงานนี้ จนกระทั่งโครงงานนี้สำเร็จลุล่วงไปด้วยดียิ่ง จึงขอบคุณทุกท่านที่กล่าวมาข้างต้นเป็นอย่างสูงมาไว้ ณ. ที่นี้ด้วย

คณะผู้จัดทำ

27 มีนาคม 2537

หนังสืออ้างอิง

1. ยืน ภู่วรวรรณ ," เทคโนโลยีฮาร์ดแวร์ IBM PC ", บ. ซีเอ็ดยูเคชั่น จำกัด ,กรุงเทพฯ, 2533

2. นิวัฒน์ มั่นคง , วันชัย ภาสุรพงศ์พันธ์, สหรัฐ คนองศิลป์, และ เอกวิทย์ จิตรดา, "เครื่องบันทึกข้อมูลอนาล็อกที่ใช้กับไมโครคอมพิวเตอร์ XT/ AT ", ปริญญานิพนธ์ของนักศึกษาปริญญาตรีสาขา วิศวกรรมโทรคมนาคม ภาควิชา โทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง , 2536

3. Al Strevens, " TUTBO C" : Memory - Resident Utility ,Screen I/O and programmingTechnique",Management Information Source , Inc. USA 1987

4. Borland International ," Turbo C reference Guide V.2.0 ",Borland international ,USA,1988

5. J. Terry Godfrey ," Apply C : The IBM Microcomputer", Pentic - Hall, Inc. , New Jersey , USA,1990