ปีการศึกษา 2531

การใช้คอมพิวเตอร์ในทางดนตรี กลุ่ม 2

(Computer Music II)

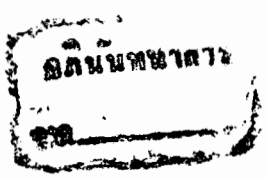
โดย

นายมีโชค เมรัมพนกุล

นางสาวจุฬา ศิวเสวัสดิ์

อาจารย์ที่ปรึกษา

ผศ.เกรวชัย ไมตรี



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

023145 -8 ส.ค. 532

ปริญญานิพนธ์ปีการศึกษา 2531

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การใช้คอมพิวเตอร์ในทางดนตรี กลุ่ม 2 (Computer Music 2)

ผู้จัดทำ

1. นายมีโชค อมรนัฑนกุล
2. นางสาวริลดา ติรณสวัสดิ์

.....อาจารย์ที่ปรึกษา

(ผศ. ครรชิต ไมตรี)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

023145

การใช้คอมพิวเตอร์ในทางดนตรี กลุ่ม2

Computer Music II

โดย

นายมีโชค อมรพัฒนกุล

นางสาววิลดา ติรณสวัสดิ์

อาจารย์ที่ปรึกษา

ผศ. ครรชิต ไมตรี

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2531

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้คอมพิวเตอร์ในทางดนตรี กลุ่ม 2

นายมีโชค อมรวัฒนกุล
นางสาววิลดา ติรถสวัสดิ์
ผศ.ดร.ชัช ไมตรี อาจารย์ที่ปรึกษา
ปีการศึกษา 2531

บทคัดย่อ

ในปัจจุบันนี้ คอมพิวเตอร์ได้เข้ามามีบทบาทในงานหลาย ๆ ด้านของมนุษย์ สำหรับงานทางด้านศิลปะนั้น คอมพิวเตอร์ถูกนำมาใช้ในงานต่าง ๆ เช่น คอมพิวเตอร์เพื่อการออกแบบ การประยุกต์ใช้คอมพิวเตอร์ในงานทางด้านกราฟฟิค แต่ทางด้านดนตรีนั้น ถึงแม้จะเป็นศิลปะสาขาหนึ่ง แต่ก็ยังมีการประยุกต์ใช้คอมพิวเตอร์น้อยมาก การนำคอมพิวเตอร์มาใช้ในทางดนตรีนั้นเริ่มเกิดขึ้นเมื่อเครื่องดนตรีเปลี่ยนจากการใช้สัญญาณอนาลอกมาเป็นสัญญาณดิจิตอล และมีการทำพคมาตรฐานขึ้นมาเพื่อใช้ในการติดต่อระหว่างเครื่องดนตรีตนละประเภทหรือยี่ห้อ ซึ่งมาตรฐานนี้เรียกว่า "มิดี" (MIDI : Musical Instrument Digital Interface) เนื่องจากการที่สัญญาณเป็นดิจิตอล เราจึงสามารถนำคอมพิวเตอร์มาใช้ควบคุมการทำงานของเครื่องดนตรีได้ ในโครงงานนี้ จึงทำการศึกษาเรื่องดังกล่าว มีการต่อวงจรอินเตอร์เฟสอย่างง่าย ๆ ที่ราคาไม่แพงนัก และเขียนโปรแกรมควบคุมการติดต่อระหว่างคอมพิวเตอร์กับคีย์บอร์ดคาสิโอ รุ่น CT-640 โดยโปรแกรมที่เขียนจะทำงานในระดับเบื้องต้น คือ บันทึกเพลง เล่นกลับ และเก็บเพลงลงไฟล์ได้ ซึ่งถือว่าเป็นพื้นฐานสอบกฎหมายต่าง ๆ ในเบื้องต้น เพื่อให้สามารถนำไปใช้งานได้จริงและมีประสิทธิภาพยิ่งขึ้นต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Computer Music II

Meechoke Amornpattanakul

Rilada Tiranaswadi

Professor Assistant Kanchit Maitri Advisor

1988

Abstract

Nowadays , we can say that computers play many roles in human life. Many applications of computer in art are developed , for example , computer aid design , computer graphics application. Although music is one part of the art , the applications are rarely seen. The applications of computer in music began when musical instruments changed from analog instruments to digital instruments. A standard was established for the communication between different type or different manufacturer instruments. This standard is called "MIDI" (Musical Instrument Digital Interface). Since the signal is digital , we can use a computer for control the performance of the instrument. In this project , we studied about MIDI , made the easy MIDI interface circuit which is not too expensive and wrote a program that control the communication between computer and keyboard CASIO CT-640. This program acts in a basic level , such as record a song,play back and store in a file. This project is the step of testing the theory for apply to use later.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ประวัติความเป็นมาของมิดี

เครื่องดนตรีจากอดีตถึงปัจจุบัน ได้มีการพัฒนามาเรื่อย ๆ จากเครื่องดนตรีที่อาศัยเพียงการสั่นสะเทือนของเส้นสายหรืออากาศ ก็เริ่มพัฒนานำเทคโนโลยีสมัยใหม่มาใช้ โดยมีการนำวงจรอิเลคโทรนิคส์ต่าง ๆ เช่น วงจรออสซิลเลเตอร์ , วงจรฟิลเตอร์ มาประกอบกัน เพื่อทำการผลิตเสียง นอกจากนี้ยังมีการสังเคราะห์เสียงของเครื่องดนตรีชนิดต่าง ๆ หรือเสียงอื่น ๆ ในธรรมชาติ แล้วนำมาเก็บไว้ในหน่วยความจำของเครื่องดนตรีเพียงชิ้นเดียว หลังจากนั้นได้มีการพัฒนาเครื่องดนตรีที่เป็นอนาลอก มาเป็นสัญญาณดิจิตอล เพื่อพัฒนาคุณภาพของเสียงและลดข้อจำกัดของเสียงทางด้านอนาลอก

เมื่อเครื่องดนตรีได้พัฒนากันขึ้นเรื่อย ๆ นักดนตรีก็เริ่มให้ความสนใจกับเครื่องดนตรีอิเลคโทรนิคส์เพิ่มขึ้น ทำให้มีบริษัทผู้ผลิตผลิตกันขึ้นหลายราย แต่ปัญหาที่ตามมาคือความไม่เป็นมาตรฐานเดียวกัน ซึ่งจะทำให้ผู้ใช้ไม่มีอิสระที่จะเลือกใช้เครื่องดนตรีต่างยี่ห้อมาทำงานร่วมกัน ซึ่งจะเกิดข้อผูกมัดให้ซื้ออุปกรณ์กับผู้ผลิตรายเดียวตลอด

ต่อมา บริษัท ซีเควนเชียล เซอร์คิท อินคอร์โพเรชั่น จำกัด (Sequential Circuit Incorporation) ร่วมกับผู้ผลิตจากญี่ปุ่นได้ร่วมกันตั้งมาตรฐานสำหรับเครื่องดนตรีอิเลคโทรนิคส์ประเภทซินธิไซเซอร์ขึ้น เรียกว่า "มิดี" โดยมีจุดประสงค์เพื่อแก้ไขปัญหาดังกล่าวข้างต้น ทำให้เรามีอิสระในการเลือกเครื่องดนตรีเพื่อใช้งานร่วมกันตามที่ต้องการ

เมื่อมีการนำคอมพิวเตอร์มาประยุกต์ใช้ร่วมกับเครื่องดนตรี จึงต้องมีการเปลี่ยนระบบข้อมูลของคอมพิวเตอร์มาเป็นระบบมิดี เพื่อให้การสื่อสารติดต่อกันระหว่างเครื่องมือทั้งสองนั้นเป็นไปได้ ซึ่งจะทำได้โดยการต่อวงจรมิดีอินเตอร์เฟส เชื่อมต่อระหว่างคอมพิวเตอร์กับเครื่องดนตรี

เมื่อสามารถนำคอมพิวเตอร์มาต่อเชื่อมโยงกับเครื่องดนตรีได้แล้ว จึงได้มีการพัฒนาซอฟท์แวร์ต่าง ๆ ทางด้านดนตรี เช่น โปรแกรมประเภทซีเควนเซอร์ ซึ่งช่วยในระบบอัดเสียง โปรแกรมประเภทคอมโพสเซอร์ ซึ่งช่วยในการแต่งเพลง เป็นต้น ซึ่งนักดนตรีจะสามารถใช้โปรแกรมต่าง ๆ เหล่านี้ มาสร้างขัดเกลาโครงสร้างทางเสียงดนตรี ซึ่งจะช่วยให้นักดนตรีสร้างเพลงได้ง่ายและมีคุณภาพมากขึ้น เหมือนกับที่โปรแกรมเวอร์ด โปรเซสเซอร์ ช่วยงานของนักประพันธ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับปัจจุบันนี้ การใช้งานคอมพิวเตอร์ในทางดนตรีเริ่มมีบทบาทมากขึ้น มีการนำมาใช้งานในวงการอุตสาหกรรมด้านเสียงและบันเทิงอย่างแพร่หลาย

1.2 วัตถุประสงค์ของโครงงาน

เนื่องจากมิดียังเป็นเรื่องใหม่ และมีแนวโน้มในการขยายตัวอย่างกว้างขวาง แต่วงจรอินเตอร์เฟสและวงจรอื่นที่เกี่ยวข้องยังมีราคาแพง โครงงานนี้จึงทำขึ้น เพื่อมีวัตถุประสงค์ คือ

1. ศึกษาเรื่องต่าง ๆ ของมิดี
2. สร้างวงจรอินเตอร์เฟสที่ราคาไม่แพงนัก เพื่อใช้ในการศึกษา
3. สามารถประยุกต์ใช้คอมพิวเตอร์กับงานศิลป์ด้านดนตรี เช่น ใช้คอมพิวเตอร์ควบคุมการเก็บและส่งโน้ตดนตรี เป็นต้น

1.3 เนื้อหาของวิทยานิพนธ์

วิทยานิพนธ์เล่มนี้มีเนื้อหาดังนี้ คือ

บทที่ 1 เป็นบทนำ กล่าวถึงประวัติความเป็นมาของมิดี , ทำไมจึงต้องมีการวางมาตรฐานมิดี และจุดประสงค์ในการทำโครงงาน

บทที่ 2 เป็นความรู้ทั่วไปเกี่ยวกับมิดี เช่น มิดีคืออะไร วิธีการต่อเชื่อมเครื่องดนตรีที่ถูกวิธี ประโยชน์ของมิดี วิธีหลีกเลี่ยงปัญหาต่าง ๆ อันอาจจะเกิดขึ้น

บทที่ 3 เป็นการกล่าวถึงรหัสต่าง ๆ และโพรแกรมการทำงานของมิดี ซึ่งรหัสต่าง ๆ เหล่านี้นี่เองที่เราต้องทำการศึกษาอย่างละเอียด เพื่อนำไปใช้ในการเขียนโปรแกรม

บทที่ 4 เป็นวิธีออกแบบวงจรมิดีอินเตอร์เฟสเชื่อมต่อระหว่างคอมพิวเตอร์กับเครื่องดนตรี รวมทั้งรูปวงจรที่ต่อ และนอกจากนี้ ยังมีการอธิบายซอฟท์แวร์ที่เขียนขึ้น โดยมีการอธิบายถึงความสามารถของซอฟท์แวร์ และการทำงานของรูทีนที่สำคัญ

บทที่ 5 เป็นบทสรุป ซึ่งจะกล่าวถึงปัญหาใหญ่ ๆ ที่เกิดขึ้นในระหว่างทำการทดลอง รวมทั้งแนวทางในการพัฒนาต่อ เพื่อให้ได้โปรแกรมที่สามารถนำไปใช้งานจริงได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
เรื่องทั่วไปของมิดี

2.1 มิดี (MIDI) คืออะไร

2.1.1 การสนทนาระหว่างเครื่องดนตรี

เครื่องดนตรีเริ่ม "สนทนา" กันได้ในปี 1980 ซึ่งถือเป็นการปฏิวัติครั้งยิ่งใหญ่ในโลกของดนตรี ปัจจุบันนี้ เครื่องดนตรีชิ้นหนึ่งสามารถพูดคุยกับเครื่องอื่นได้ เช่น บอกว่า "ให้เล่นโน้ตซี ออกเทฟกลาง ความแรงการกดประมาณ 60% แล้วเล่นโน้ตอี 4 ค้างชั่วนิดเดียวเล็กน้อย" เมื่อเครื่องดนตรีชิ้นที่สองได้ยินคำสั่งนี้ จะเล่นโน้ตตามสั่งออกมาตรายเท่าที่มันยังเข้าใจภาษาที่ใช้กัน "ภาษา" ที่ปัจจุบันใช้ร่วมกันระหว่างเครื่องดนตรีทางอิเลคทรอนิคส์ เรียกว่า "มิดี"

ถึงแม้ว่ามิดีจะเป็นเรื่องใหม่ แต่ปัจจุบันก็เป็นที่รู้จักกันทั่วไปในวงการดนตรี มิดีย่อมาจากคำว่า "Musical Instrument Digital Interface"

2.1.1 การเติบโตของมิดี

มิดีรู้จักกันกว้าง ๆ ว่า เป็นภาษาระหว่างเครื่องดนตรี และได้มีการขยายตัวไปช่วงระยะเวลาเพียงชั่ว ๆ เพราะมันสามารถประยุกต์ใช้ได้ในสถานการณ์ต่าง ๆ มากมาย

มิดีเป็นภาษาสากล ซึ่งเป็นเพียงภาษาเดียวที่สามารถจัดการการสื่อสารได้ระหว่างเครื่องดนตรีที่ผลิตจากคนละประเทศ หรือคนละยี่ห้อ

นอกจากนี้ มิดีสามารถประยุกต์ใช้ได้กับเครื่องดนตรีหลาย ๆ ชนิด ซึ่งลักษณะข้อนี้ทำให้เราสามารถที่จะทำเปียโนไฟฟ้ามา "คุย" กับซินธิไซเซอร์ หรือ ดรัมแมชชีน (Drum Machine) จะเห็นได้ว่า มิดีมีศักยภาพอย่างมากสำหรับการเติบโตและพัฒนา

2.1.3 วิธีสนทนาแบบมิดี

ดูจากที่ด้านหลังของเครื่องดนตรี จะเห็นคอนเน็คเตอร์ (Connector) มากมาย รวมทั้งมิดีอิน (MIDI in) , มิดีเอาท์ (MIDI out) และอาจรวมถึงมิดีทรู (MIDI thru) สิ่งเหล่านี้นับว่าเป็นหูและปาก ซึ่งเป็นส่วนที่สำคัญยิ่งของมิดี "การสนทนาของมิดี" จะเดินทางจากมิดีเอาท์ของเครื่องดนตรีชิ้นหนึ่งไปที่มิดีอินของเครื่องดนตรีอีกชิ้นหนึ่ง โดยการต่อสายมิดีเพียงสายเดียวระหว่างคอนเน็คเตอร์ทั้งสอง สายมิดี (MIDI cable) ใช้ DIN 5 ขา ซึ่งจะต่อ

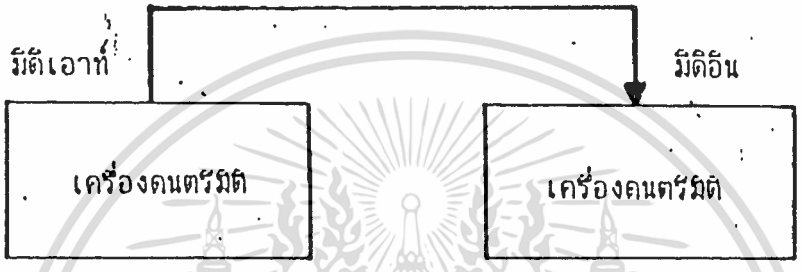
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เข้าพอดีกับรู 5 รูในแต่ละคอนเน็คเตอร์พอดี

หน้าที่ของคอนเน็คเตอร์ทั้ง 3 คือ

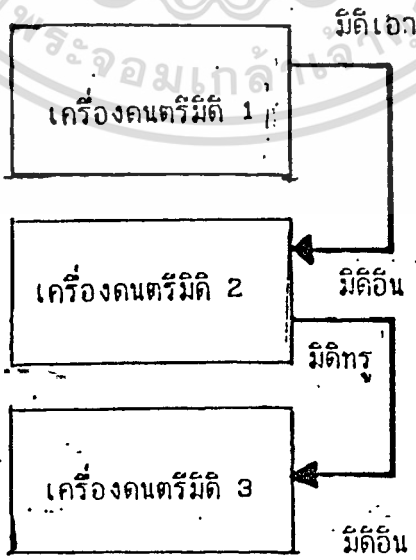
มิดีอิน สำหรับใช้ "ฟัง" การสนทนา นั่นคือ เป็นทางเข้าของข้อมูลมิดี

มิดีเอาท์ สำหรับใช้ "พูด" ส่งการสนทนาจากเครื่องดนตรี ดังนั้น จึงเป็นทางออกของข้อมูลมิดี



รูปที่ 1 แสดงถึงการต่อมิดีเอาท์กับมิดีอิน

จะเห็นว่า มิดีอินและมิดีเอาท์ง่ายต่อการเข้าใจและใช้งาน แต่มิดิทรูจะยากกว่านั้นเล็กน้อย มิดิทรูคล้าย ๆ กับมิดีเอาท์ นั่นคือ มันสามารถ "พูด" ไปที่เครื่องดนตรีถัดไป อย่างไรก็ตาม สิ่งที่พูดไปนั้นไม่ใช่ข้อมูลของมันเอง แต่เป็นข้อมูลที่ได้ยินมาจากทางมิดีอิน



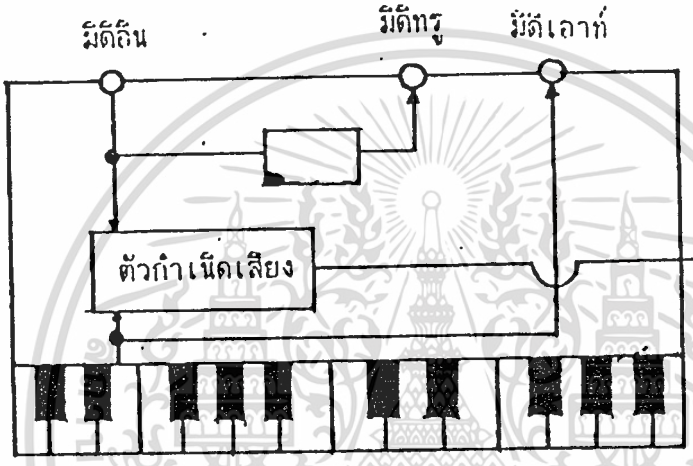
รูปที่ 2 แสดงการไหลของข้อมูลผ่านทางคีย์บอร์ด 3 ตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการติดตั้งลักษณะนี้ ข้อมูลมิดีจะออกจากมิดีเอาท์ของคีย์บอร์ดตัวที่ 1 และเข้าไปที่มิดีอินของคีย์บอร์ดที่ 2 และยังคงผ่านไปที่มิดีอินของคีย์บอร์ดที่ 3 ทางมิดิทรูของคีย์บอร์ดที่ 2 ในลักษณะนี้ ข้อมูลจากคีย์บอร์ดตัวที่ 1 สามารถควบคุมคีย์บอร์ดทั้ง 2 และ 3

การนำเครื่องดนตรีมากกว่า 2 ชิ้นมาต่อรวมกัน เราเรียกว่า ระบบมิดี (MIDI system) ซึ่งจะเห็นได้ว่า ถ้าไม่มีมิดิทรู เราจะไม่สามารถติดตั้งระบบมิดิได้

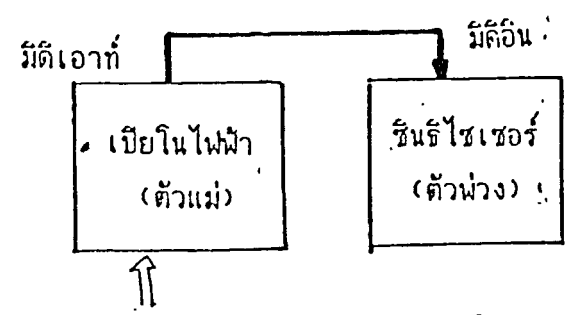
สำหรับโครงสร้างของมิดีคีย์บอร์ด เป็นดังรูปที่ 3



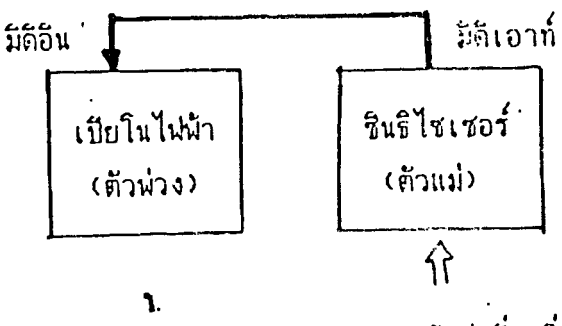
รูปที่ 3 โครงสร้างของมิดีคีย์บอร์ด

2.1.4 ทิศทางของการสเทนามิดี

ข้อมูลมิดีจะถูกส่งจากมิดีเอาท์ไปมิดีอิน หรือจากมิดิทรูไปมิดีอิน ข้อมูลจะถูกส่งเป็นแบบทางเดียวเสมอ เพื่อที่จะมีตัวส่งและตัวรับที่ทำหน้าที่เดียวตลอด เครื่องดนตรีเหล่านี้จะทำหน้าที่เป็นตัวข่วง (Slave) หรือไม่ก็เป็นตัวแม่ (Master)



ก.



ข.

รูปที่ 4 การต่อตัวแม่และตัวข่วง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4 ก แสดงให้เห็นถึงการส่งข้อมูลจากเปียโนไปยังซินธิไซเซอร์ ในกรณีนี้ เปียโนเป็นตัวแม่ และซินธิไซเซอร์เป็นตัวพ่วง

รูปที่ 4 ข ข้อมูลส่งจากซินธิไซเซอร์ไปเปียโน นั่นคือ ทำหน้าที่กลับกัน จะเห็นได้ว่า การต่อสายเป็นเรื่องที่สำคัญมาก

นอกจากนี้ ความสัมพันธ์ระหว่างตัวแม่กับตัวพ่วงไม่จำเป็นต้องเป็น 1:1 จากในรูปที่เคยแสดงไว้แล้ว จำนวนตัวพ่วงอาจจะเพิ่มขึ้นเป็น 1:2 , 1:3

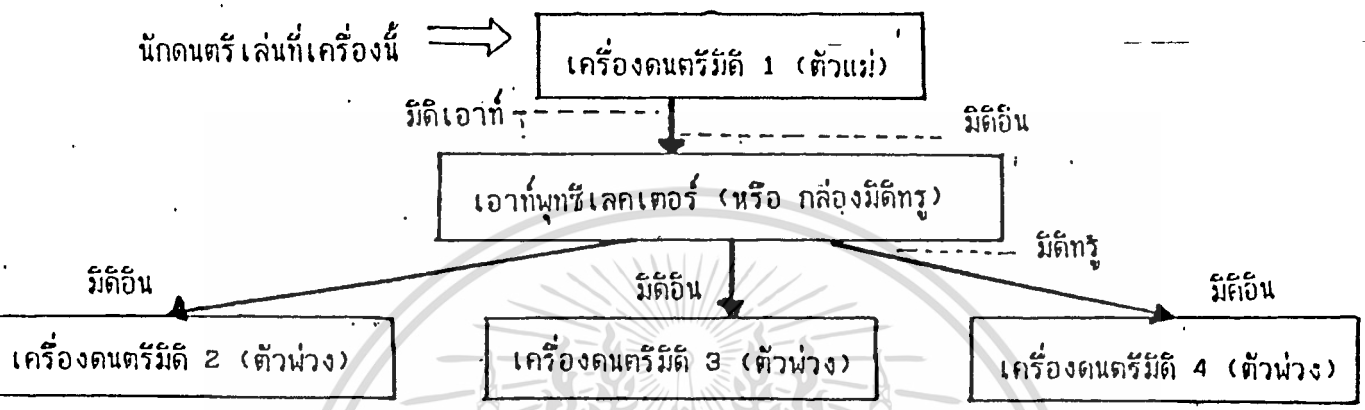
สรุปได้ว่า ในระบบมิดิขนาดใหญ่ ถ้าเราตรวจสอบว่า การไหลของข้อมูลเป็นแบบทางเดียวและถูกต้อง และความสัมพันธ์ระหว่างตัวแม่ - ตัวพ่วงเห็นได้อย่างชัดเจนแล้ว จะไม่มีปัญหาที่เกิดจากการต่อสาย

2.1.5 ระยะห่างของเครื่องดนตรี

จากที่กล่าวมาแล้ว จะดูเหมือนว่าเราสามารถต่อเครื่องดนตรีในลักษณะลูกโซ่ได้ โดยผ่านทางการต่อมิดิทรูหลาย ๆ ตัว อย่างไรก็ตาม ในความเป็นจริง ในการต่อมิดิทรูมากเกินไปอาจจะเกิดผลเสียได้ คือ เครื่องดนตรีตัวที่ 4 หรือ 5 อาจได้รับข้อมูลผิดพลาดได้

เพื่อลดปัญหานี้ ให้ต่อเครื่องดนตรีตัวแม่และตัวพ่วงใกล้กันที่สุดเท่าที่จะเป็นได้ ในที่นี้ไม่ได้หมายถึงตำแหน่งทางกายภาพ (Physical Placement) แต่หมายถึง จำนวนการต่อมิดิระหว่างตัวแม่และตัวพ่วง ซึ่งทำได้ง่าย ๆ โดยใช้กล่องมิดิทรู (MIDI Thru Box) หรือ เอาท์พุท ซีเลกเตอร์ (Output Selector) กล่องมิดิทรูจะส่งข้อมูลจากตัวแม่ไปตัวพ่วงในเวลาเดียวกัน ดังนั้น แต่ละตัวพ่วงจะได้รับข้อมูลเป็นตัวแรกพร้อม ๆ กัน ปัจจุบัน มีอุปกรณ์มิดิมากมายหลายชนิด แต่กล่องมิดิทรูนับว่าจำเป็นที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5 แสดงการต่อโดยใช้กล่องมิดีทรู

2.2 ประโยชน์ของมิดี

2.2.1 เครื่องดนตรี 2 ชิ้นสามารถเล่นพร้อมกัน

ตัวอย่างเช่น มิดีเอาท์ของเปียโนไปป้อนต่อกับมิดีอินของซินธิไซเซอร์ ถ้าซินธิไซเซอร์ถูกเซทให้เป็นเสียงเครื่องสาย การเล่นเปียโนจะทำให้ซินธิไซเซอร์เล่นเพลงในส่วนเดียวกันโดยเป็นเสียงเครื่องสาย การเล่นเครื่องดนตรี 2 ชนิดโดยพร้อมเพรียงกันเป็นการใช้มิดีอย่างพื้นฐาน ทำให้เราได้เสียงต่าง ๆ เพิ่มขึ้นมากมาย

2.2.2 การรวมเครื่องดนตรีคนละประเภท

จากได้กล่าวมาแล้ว เราต้องการให้คีย์บอร์ด "คุย" กัน แต่ถ้านำเปียโนมาคุยกับดรัมแมชีน ก็เป็นการยากที่จะคาดเดาผลที่ได้

สมมุติว่า เราต่อเปียโนกับดรัมแมชีน โดยให้เปียโนเป็นตัวแม่ ดรัมแมชีนเป็นตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

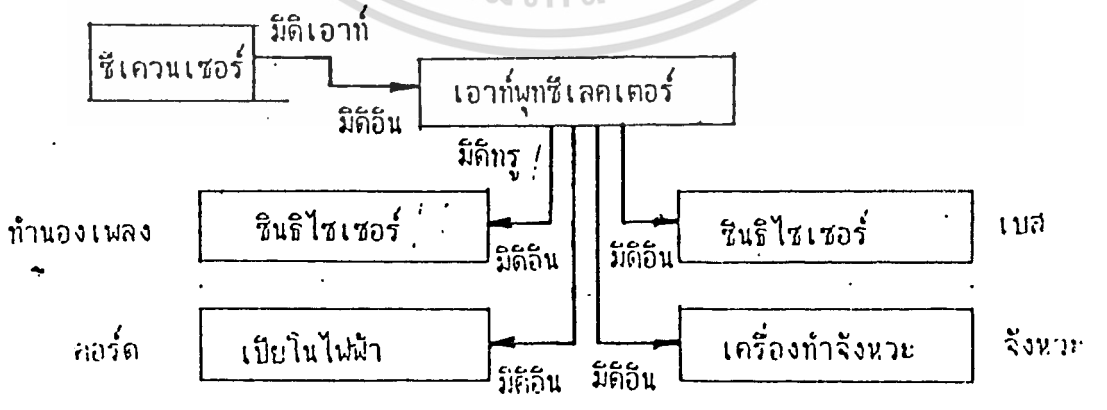
ผ่วง ถ้าส่งโน้ตซี (โด) ออกไป ดรัมแมชีนจะรับรู้เป็นเสียงกลองเบส (Bass Drum) ส่งโน้ตดี (เร) อาจจะเล่นเป็นเสียงฉาบ ฯลฯ ดังนั้น แทนที่จะใช้ไม้ตีกลอง เราสามารถเล่นกลองได้โดยกดคีย์ที่เหมาะสมบนคีย์บอร์ด

การรวมกันของดรัมแมชีนและเปียโนเป็นเพียงตัวอย่างหนึ่งเท่านั้น โดยการใช้มิดิ เราสามารถทำให้เครื่องดนตรีทำงานกลับกันได้

2.2.3 ระบบซีเควนเซอร์ (Sequency System)

ซีเควนเซอร์ คือ เครื่องมือในการบันทึก แก้ไข และเล่นกลับสำหรับเครื่องดนตรีไฟฟ้าทั่วไป ซึ่งเป็นวิธีหนึ่งในการเพิ่มความสามารถของมิดิ ก็คือการนำซีเควนเซอร์มาใช้ร่วมด้วย ในการเล่นเครื่องดนตรีพร้อม ๆ กัน ดังที่กล่าวมาแล้วในข้อ 1 มีความสามารถจำกัดอยู่ที่คีย์บอร์ดตัวแม่ แต่เนื่องจากซีเควนเซอร์มีความสามารถในการจำเครื่องดนตรีมากมาย ทำให้มันสามารถใช้งานกำเนิดเสียงได้หลายเสียง

ยกตัวอย่าง ดังรูปที่ 6 ซีเควนเซอร์ที่เป็นตัวแม่สามารถควบคุมซินธิไซเซอร์ให้เล่นทำนองเพลง (melody) ให้เปียโนเล่นคอร์ด (chord) ซินธิไซเซอร์อีกตัวเล่นเบส (bass) และ ดรัมแมชีนเล่นจังหวะ (rhythm) ในเวลาเดียวกัน ข้อมูลสำหรับการใช้งานทั้ง 4 ส่วนถูกเก็บอยู่ในซีเควนเซอร์ แล้วเมื่อข้อมูลถูกส่งไปเครื่องดนตรีทั้งสี่ การทำงานอัตโนมัติก็จะเริ่มขึ้น



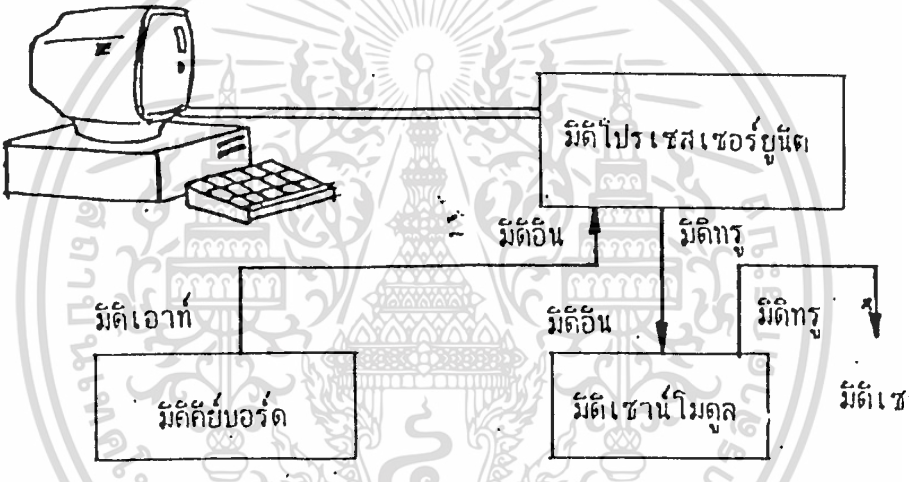
รูปที่ 6 การต่อระบบซีเควนเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.4 ระบบดนตรีคอมพิวเตอร์ (Computer Music System)

ก่อนที่จะมีมิดี ดูเหมือนว่าคอมพิวเตอร์จะไม่มีส่วนเกี่ยวข้องกับดนตรีมากนัก การสนทนาระหว่างคอมพิวเตอร์กับเครื่องดนตรีเป็นเรื่องยากที่จะเข้าใจกัน การที่มีมิดี ทำให้เราสามารถสร้างวงจรที่แปลภาษาของคอมพิวเตอร์เป็นภาษามิดี และแปลภาษามิดีให้เป็นภาษาคอมพิวเตอร์ วงจรนี้เราเรียกว่า "อินเตอร์เฟส" (Interface)

คอมพิวเตอร์จะใช้ซอฟท์แวร์ในการควบคุมการทำงาน ซอฟท์แวร์บางชิ้นจะทำให้คอมพิวเตอร์ทำหน้าที่เป็นซีเควนเซอร์ บางซอฟท์แวร์จะทำหน้าที่เป็น "ซาวน์ อีดิต" (Sound Edit) ซึ่งให้คอมพิวเตอร์ทำการเปลี่ยนแปลง (modify) เสียงของซินธิไซเซอร์



รูปที่ 7 ระบบดนตรีคอมพิวเตอร์

รูปที่ 7 แสดงระบบดนตรีคอมพิวเตอร์ที่มีซีเควนเซอร์ซอฟท์แวร์ (Sequency Software) ประโยชน์ของระบบนี้ คือ การที่มีหน่วยความจำจำนวนมากของคอมพิวเตอร์ เพื่อใช้เก็บเพลงได้หลายเพลง หรือ เพลงที่ยาว ๆ และจอภาพของคอมพิวเตอร์ทำให้ง่ายต่อการตรวจสอบข้อมูลการทำงานของมิดี

2.2.5 ระบบมิดีในงานด้านอื่น ๆ

เราได้พูดถึงระบบมิดีในแง่ต่าง ๆ แล้ว แต่ยังมีวิธีประยุกต์ใช้มิดีอีกมากมาย เช่น มิดีซีเควนเซอร์สามารถใช้ร่วมกับเครื่องอัดเทปแบบหลายแทรค (Multi-track Tape Recorder : MRT) เพื่อเพิ่มจำนวนแทรค ซึ่งวิธีประยุกต์ใช้เหล่านี้ยังมีอนาคตเปิดกว้างอีกสำหรับผู้ที่มีความคิดสร้างสรรค์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

023145

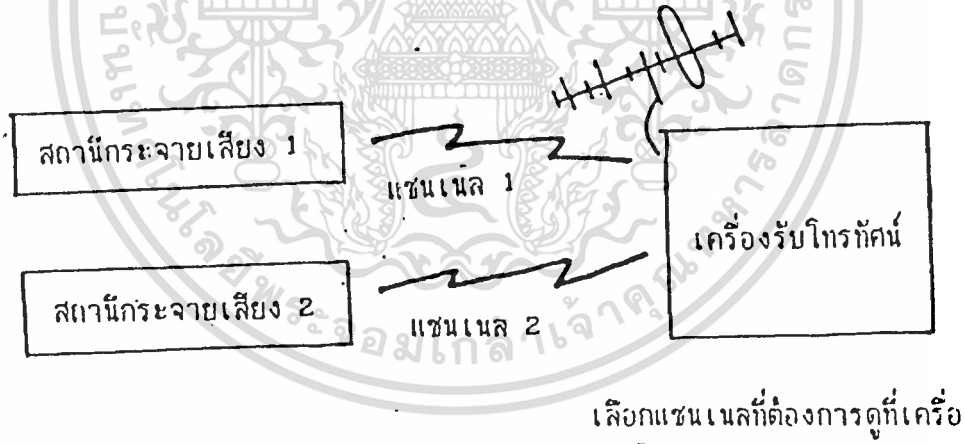
2.3 การส่งข้อมูลดนตรีทางมิดิ

2.3.1 แชนเนล (Channel)

มิดิสามารถส่งข้อมูลหลาย ๆ ประเภทจากตัวแม่ไปตัวพ่วง ไม่ว่าจะส่วนเดียวหรือหลายส่วน เช่น ในระบบซีเควนเซอร์

จากจุดประสงค์นี้ มิดิจึงมี 16 แชนเนล ภาระสนทนาของมิดิในประเภทที่ต่างกันสามารถส่งไปทาง 16 แชนเนลแยกกัน ดังนั้น ด้วยสายมิดิ 1 เส้น จึงสามารถเล่นได้ 16 ส่วน ในเวลาเดียวกัน

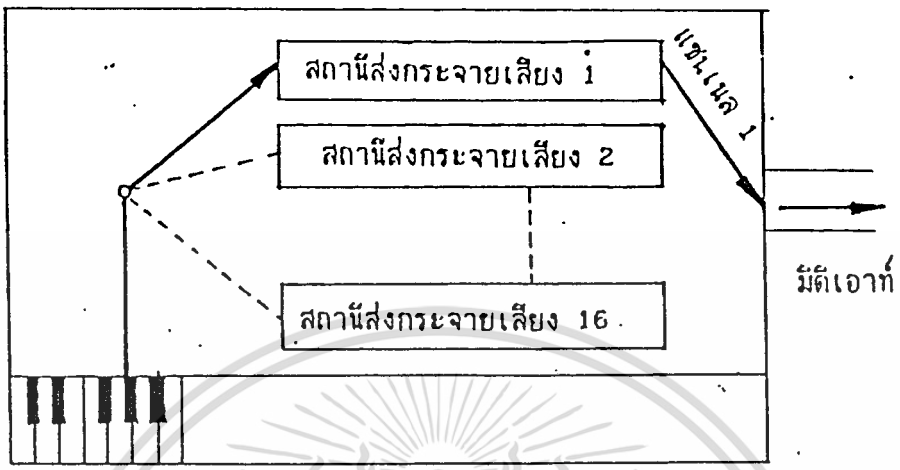
หลักการของมิดิแชนเนลก็คล้าย ๆ กับการส่งแพร่เสียงและภาพของโทรทัศน์ ทุก ๆ สถานีโทรทัศน์จะส่งรายการของตนในเวลาเดียวกัน นั่นคือ เสาอากาศเครื่องรับโทรทัศน์จะได้รับแชนเนลทั้งหมดในเวลาพร้อมกัน แต่อย่างไรก็ตาม ทุก ๆ สถานีมีการส่งในแชนเนลที่ต่างกัน เราก็จะต้องทำการเลือกแชนเนลที่เราต้องการดู ดังรูปที่ 8



รูปที่ 8 หลักการของแชนเนลในระบบโทรทัศน์

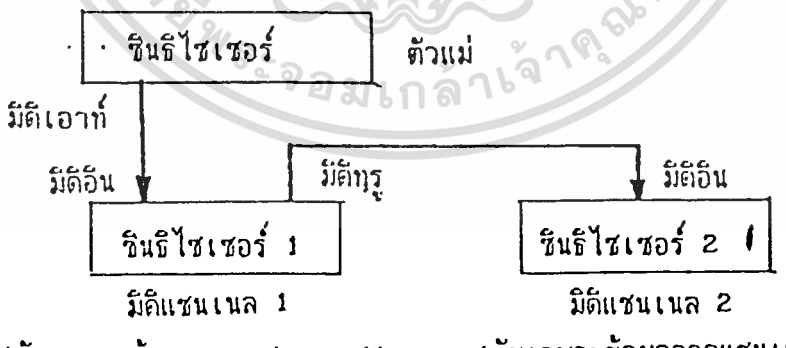
มิดิแชนเนลก็ทำในลักษณะเดียวกัน ตัวแม่มิดิทำหน้าที่เหมือนกับตัวส่งกระจายเสียงโทรทัศน์ ตัวพ่วงมิดิก็เหมือนเครื่องรับโทรทัศน์ จะต่างกันก็เฉพาะการส่งสัญญาณโทรทัศน์ผ่านทางอากาศ แต่การส่งสัญญาณมิดิผ่านทางสายเคเบิ้ล ตัวแม่มิดิจะเลือกว่าจะส่งไปที่แชนเนลไหน ดังรูปที่ 9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 9 การส่งมิดีแชนเนล

ส่วนตัวพ่วงจะเลือกว่าต้องการได้ยินแชนเนลไหน ถึงแม้ว่าจะได้รับข้อมูลทั้ง 16 แชนเนล แต่ตัวพ่วงจะได้ยินเพียงแชนเนลที่เลือกไว้



รูปที่ 10 ตัวอย่างการรับมิดีแชนเนล

ตัวอย่าง ถ้าเครื่องดนตรีมีการต่อไว้ดังรูปที่ 10 ซินธิไซเซอร์ 1 จะรับรู้เพียงข้อมูลจากแชนเนล 1 ของมิดีเคเบิ้ล ซินธิไซเซอร์ 2 รับรู้แชนเนล 2 ดังนั้น ถ้าคีย์บอร์ดตัวแม่ส่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลไปยังมิดิแชนเนล 1 จะมีเพียงซินธิไซเซอร์ 1 เท่านั้นที่ตอบสนอง เช่นกัน ถ้าส่งข้อมูลไปยังมิดิแชนเนล 2 จะมีเพียงซินธิไซเซอร์ 2 เท่านั้นที่ตอบสนอง ถึงแม้ว่าซินธิไซเซอร์ 1 จะได้รับข้อมูลเช่นกัน แต่มันไม่สามารถตอบสนองได้เพราะมันถูกเสวิตช์ไปใช้แชนเนลที่ต่างกัน หรือกล่าวอีกนัยหนึ่ง เพื่อให้ระบบทำงานถูกต้อง เราต้องเลือกมิดิแชนเนลด้วยความระมัดระวัง

ด้วยหลักการของมิดิแชนเนล เราจึงสามารถเล่นเพลงต่าง ๆ กัน 16 ส่วนบนเครื่องดนตรี 16 ชิ้น โดยมีการส่งโดยการใช้มิดิเคเบิ้ลเพียงเส้นเดียว

2.3.2 โหมด

เป็นการบอกถึงโหมดการทำงานของมิดิ ซึ่งประกอบด้วย 4 โหมดดังต่อไปนี้

1. ออมนิออน , โพลี (OMNI ON , POLY)
2. ออมนิออน , โมโน (OMNI ON , MONO)
3. ออมนิออฟ , โพลี (OMNI OFF , POLY)
4. ออมนิออฟ , โมโน (OMNI OFF , MONO)

ซึ่งจะกล่าวรายละเอียดในบทต่อไป

2.3.3 ส่วนประกอบหลักในข้อมูลมิดิ

มิดิประกอบไปด้วยข้อมูลหลายประเภท ซึ่งใช้ส่งรายละเอียดของการทำงานต่าง ๆ จากตัวแม่ไปตัวลูก ข้อมูลดังกล่าวก็เช่น การกดหรือปล่อยโน้ต , การกดหรือปล่อยแดมเปอร์เพเดิ้ล (Damper Pedal) ข้อมูลต่าง ๆ เหล่านี้ จะแบ่งเป็นข้อมูลแชนเนล (Channel Message) และข้อมูลระบบ (System Message) ซึ่งจะได้กล่าวถึงโดยละเอียดในบทต่อไป

2.4 การหลีกเลี่ยงปัญหาของมิดิ

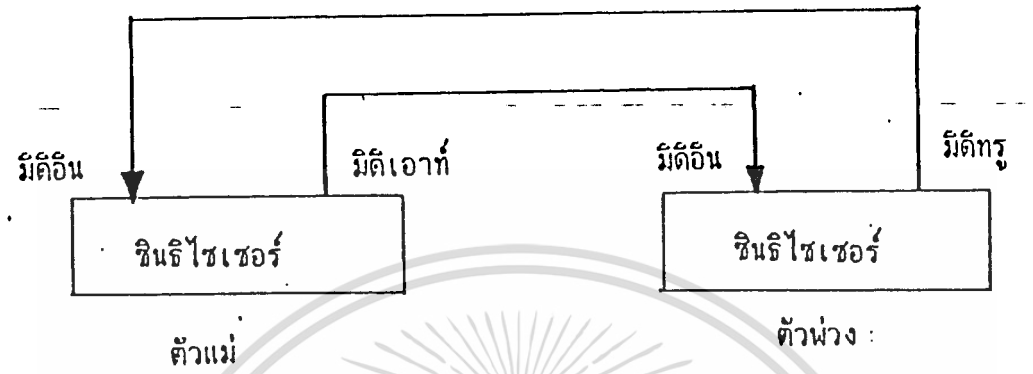
การทำงานของมิดิต้องการการจัดการแลกเปลี่ยนข้อมูลจำนวนมากมาย ซึ่งเป็นการเปิดโอกาสในการเกิดความผิดพลาด ปัญหาที่มิดิว่าจะเกิดอาจจะมาจากการต่อที่ผิด ๆ การขาดความรู้ และองค์ประกอบอื่น ๆ ในส่วนนี้ จะทำการพิจารณาปัญหาที่เกิดขึ้นทั่ว ๆ ไปและวิธีแก้ไข

2.4.1 การต่อสายมิดิเคเบิ้ล

ถ้าการต่อสายเคเบิ้ลทำไม่ถูกวิธี การทำงานของข้อมูลจะไม่เป็นการสื่อสารที่สม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บูรณ์ ต้องให้แน่ใจว่า เราได้ต่อสายจากมิดิเอาท์ของตัวแม่ไปที่มิดิอินของตัวพ่วง (หรือจากมิดิทรูไปมิดิอิน)



รูป 11 การต่อเคเบิ้ลที่ผิดวิธี

จากรูปที่ 11 จะเห็นว่าตัวแม่ถูกต่อกลับมาที่มิดิอินของตัวเอง ในกรณีนี้ ให้ปิดเครื่องและต่อเคเบิ้ลใหม่ให้ถูกต้อง อย่าต่อมิดิเอาท์ไปที่มิดิอินของเครื่องดนตรีชิ้นเดียวกัน

2.4.2 การเสียบเคเบิ้ลมิดิและการเปลี่ยนเซ็นเนล

เมื่อต้องการเปลี่ยนการต่อเคเบิ้ล ให้ปิดเครื่องก่อนดึงเคเบิ้ลเสมอ ถ้าขณะที่กำลังเล่นแล้วดึงเคเบิ้ลออก เราอาจไม่สามารถหยุดเสียงได้ บางคีย์บอร์ดไม่สามารถเปลี่ยนมิดิแชนเนลได้ขณะที่กำลังเล่นโน้ต ดังนั้นเราจึงต้องหยุดการเล่นก่อนเปลี่ยนแชนเนล

2.4.3 ลำดับการเปิดเครื่องดนตรี

ให้เปิดเครื่องดนตรีตัวแม่เป็นตัวสุดท้ายเสมอ ที่จริงแล้ว การเปิดตัวแม่ก่อนจะไม่มีผลให้เกิดความเสียหายแต่อย่างไร แต่ที่ต้องเปิดเป็นตัวสุดท้ายก็เพื่อให้การตีดตั้ง (set up) ระบบง่ายขึ้น เพราะว่าตัวแม่จะส่งข้อมูลโหมด (mode message) ออกมาเสมอ เพื่อให้ตัวพ่วงทำงานอย่างถูกต้อง

2.4.4 ช่วงของพิทช์เบนด์ (pitch bend) และผลของแรงกดคีย์ (after touch)

ถึงแม้ว่ามิดิจะสามารถส่งข้อมูลของพิทช์เบนด์และแรงกดคีย์ มันสามารถบอกได้

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ ห้ามนำไปใช้ในการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

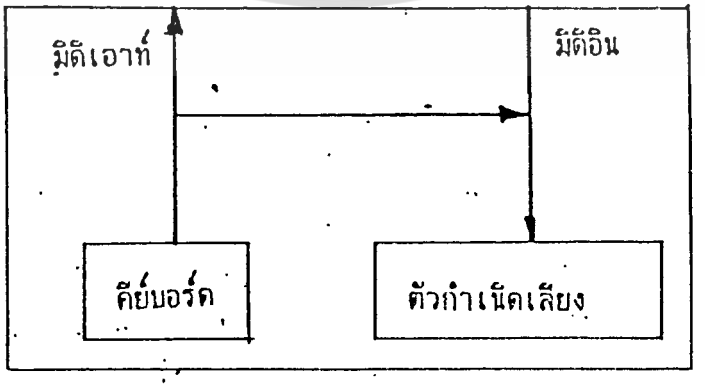
เพียงว่าระดับพิทช์เคลื่อนไปมากแค่ไหน และคีย์ที่กดมีความแรงแค่ไหน แต่ไม่สามารถบอกได้ว่าจริง ๆ แล้วพิทช์เคลื่อนที่ไปไกลแค่ไหน หรือ มีผลของแรงกดคีย์ใดบ้างที่เกิดขึ้น การกำหนดช่วงของพิทช์ที่เปลี่ยนไปได้ (Bender Range) และผลที่เกิดจากแรงกดคีย์ต้องถูกเซทไว้ที่ทั้งเครื่องดนตรีตัวแม่และตัวพ่วงโดยผู้ใช้

2.4.5 มิดีและดรัมแมชีน

ดรัมแมชีนสามารถนำมาทำงานร่วมกับซีเควนเซอร์หรือเป็นต้นกำเนิดเสียงสำหรับมิดีคีย์บอร์ด เมื่อซินธิไซเซอร์เป็นตัวพ่วง ถ้าต้องการเปลี่ยนเสียงต้องส่งรหัสข้อมูลเปลี่ยนโปรแกรม แต่ในการเปลี่ยนเสียงของดรัมแมชีน เราต้องการเพียงการเล่นโน้ตที่ต่างกันเท่านั้น เช่น โน้ตซี เป็นเสียงกลองเบส โน้ตดี เป็นฉาบ เป็นต้น ฉะนั้น ในระบบมิดีที่ใช้ซีเควนเซอร์จึงเป็นเกาวดีที่จะเซทมิดีแชนเนลของดรัมแมชีนเป็นแชนเนลที่ไม่ได้ใช้อยู่ เพื่อที่มันจะไม่ได้รับข้อมูลเปิดโน้ต (note on) ที่ไม่ได้ตั้งใจ

2.4.6 การควบคุมโลคอล (Local Control)

ในซินธิไซเซอร์บางตัวมีพารามิเตอร์ที่เกี่ยวกับมิดีที่เรียกว่า การควบคุมโลคอล เมื่อการควบคุมนี้ถูก "เปิด" (LOCAL ON) คีย์บอร์ดสามารถเล่นเสียงของตัวมันเองได้ แต่ถ้าโลคอลถูก "ปิด" (LOCAL OFF) เครื่องดนตรีสามารถส่งข้อมูลออกได้อย่างเดียว ไม่สามารถเล่นเสียงของตัวเองได้



รูปที่ 12 โครงสร้างพื้นฐานของคีย์บอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โครงสร้างพื้นฐานของซินธิไซเซอร์และเปียโนไฟฟ้าเป็นดังรูป 12 ข้อมูลที่ถูกสร้างจากคีย์บอร์ดจะถูกส่งไปส่วนกำเนิดเสียง (Sound Generator) เพื่อจะเล่นเสียงออกมา การส่งข้อมูลในส่วนนี้จะถูกทำลายลงถ้าโลคอลถูกปิด อย่างไรก็ตาม ถึงแม้โลคอลจะปิด แต่การต่อมิดีเอาท์ก็ยังคงทำงานได้อยู่

2.5 มิดีในอนาคต

นับตั้งแต่ที่เริ่มมีมิดี บริษัทผู้ผลิตเครื่องดนตรีส่วนมากประสบปัญหาความล่าช้าในการปรับมาตรฐานเดิมที่ต่างคนต่างใช้ให้เป็นไปตามมาตรฐานมิดี เพราะรหัสมิดีมีความยาวมากกว่าเดิม เมื่อเทียบกันโน้ตต่อโน้ต ทำให้ใช้เวลามากขึ้นในการรับส่งข้อมูลกัน และจะยิ่งใช้เวลามากขึ้นถ้าข้อมูลนั้น ๆ เป็นกลุ่มโน้ตที่ส่งให้เครื่องดนตรีพลายชนิด (มิดีใช้เวลา 320 ไมโครวินาที สำหรับรหัส 1 ไบท์ และโน้ต 1 ตัวใช้รหัส 3 ไบท์ ดังนั้น กินเวลา 960 ไมโครวินาที หรือประมาณ 1 มิลลิวินาที) และถ้าข้อมูลนั้นมาจากคีย์บอร์ดเครื่องดนตรีที่ประกอบด้วยกลุ่มโน้ต 5 ตัว ก็จะใช้เวลาประมาณ 5 มิลลิวินาทีทีเดียว

ในกรณีที่ข้อมูลที่จะส่งมีจำนวนมากในเวลาอันสั้น อาจทำให้ระบบส่งรับข้อมูลของมิดีทำงานผิดพลาด ปัญหานี้ได้ถูกแก้ไขในซีเควนเซอร์รุ่นใหม่ ๆ โดยเพิ่มสายมิดีเอาท์แยกสำหรับเครื่องดนตรีแต่ละแชนเนล และต่างสายกับเครื่องดนตรีนั้นเพียงเส้นเดียว ทำให้ระบบมิดีเปลี่ยนจากลูกโซ่เป็นแบบขนาน ซึ่งทำให้การส่งข้อมูลเร็วขึ้นและไม่ต้องใช้รหัสบอกแชนเนล แล้วเปลี่ยนมาเป็นโน้ตไปใช้การส่งข้อมูลด้วยโน้ตที่มีความเร็วเป็น 0 ดังนี้ รหัส 3 ไบท์ก็จะลดเหลือ 2 ไบท์ ซึ่งจะช่วยลดปัญหาได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

โหมดการทำงานของมิดีและรหัสข้อมูล

3.1 โหมดการทำงานของมิดี

โหมดการทำงานของมิดี มีไว้เพื่อควบคุมการรับส่งข้อมูลเสียง โดยจะแบ่งการพิจารณาเป็น 2 ส่วน คือ

3.1.1. ควบคุมจำนวนแชนเนล

จะใช้ออมนิโหมด (omni mode) ซึ่งถ้าออมนิออน (omni on) จะทำให้ตัวพ่วงตอบสนองต่อทั้ง 16 แชนเนลทันที ในการติดต่อระบบอย่างง่าย ๆ โดยมีตัวแม่ 1 ตัว และตัวพ่วง 1 ตัว ออมนิออนจะมีประโยชน์มาก เพราะตัวพ่วงจะไม่สนใจว่าตัวแม่กำลังใช้แชนเนลใดอยู่ มันจะ "ได้ยิน" ทุก ๆ แชนเนล อย่างไรก็ตาม ถ้ามีการนำซีเควนเซอร์มาต่อด้วย โดยทำการเล่นหลาย ๆ แชนเนล ถ้าใช้โหมดออมนิออน จะทำให้ไม่สามารถแยกแยะข้อมูลแต่ละแชนเนลได้ ดังนั้น ตัวพ่วงจึงต้องใช้โหมดออมนิออฟ เพื่อรับเฉพาะข้อมูลในเชนเนลที่เลือกไว้

3.1.2 ควบคุมจำนวนโน้ต

จะใช้โพลี (poly) และโมโน (mono) โหมด โพลีโหมด คือ การส่งในลักษณะโพลีโฟนิค (polyphonic) คือ หลาย ๆ โน้ต ซึ่งมักใช้ในเครื่องดนตรีประเภทคีย์บอร์ด และซินธ์ ส่วนโมโนโหมด จะส่งลักษณะโมโนโฟนิค (monophonic) คือ ทีละตัวโน้ต ซึ่งนำไปใช้กับเครื่องดนตรีเช่น เครื่องควบคุมกีตาร์ (guitar controller)

ในการเลือกว่า จะใช้โหมดใด มิดีจะมี 4 โหมดให้เลือก ดังรูปที่ 1

	โพลี	โมโน
ออมนิ ออน	โหมด 1	โหมด 2
ออมนิ ออฟ	โหมด 3	โหมด 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 1 โหมดทั้ง 4 ของมิดี

โหมดที่ 1:ออมนิออน / โพลี จะรับข้อมูลเข้ามาทุก ๆ แชนเนล และให้เสียงเป็นโพลีโฟนิค

โหมดที่ 2:ออมนิออน / โมโน รับข้อมูลทุก ๆ แชนเนล แต่จะเล่นทีละหนึ่งตัวโน้ตในขณะหนึ่ง ๆ

โหมดที่ 3:ออมนิออฟ / โพลี รับข้อมูลเฉพาะแชนเนลที่เลือกไว้ และเล่นเสียงที่ละหลายตัวโน้ต (มีประโยชน์ในระบบซีเควนเซอร์)

โหมด 4:รับเฉพาะมิดีแชนเนลที่กำหนดไว้ และเล่น 1 โน้ต ต่อ 1 แชนเนล ใช้กับเครื่องควบคุมกีตาร์

3.2 รหัสข้อมูล

ในการส่งรับข้อมูล จะมีการส่งทีละไบท์ ซึ่งจะมีจำนวนกี่ไบท์ ก็แล้วแต่ข้อมูลนั้น ๆ เราสามารถแบ่งประเภทของไบท์ที่ส่งได้เป็น 2 ประเภท คือ

1. ไบท์สถานะ (status byte) จะมีบิทที่ 7 เป็น 1 เสมอ
2. ไบท์ข้อมูล (data byte) จะมีบิทที่ 7 เป็น 0 เสมอ

การส่งข้อมูลมิดี จะเกิดจากการส่งไบท์ทั้ง 2 ประเภทรวมกัน โดยจะส่งไบท์สถานะไปก่อน 1 ไบท์ และส่งไบท์ข้อมูลตามไปกี่ไบท์ก็แล้วแต่รหัสนั้น ๆ

สำหรับข้อมูลมิดีนั้น แบ่งได้เป็น 2 ประเภทใหญ่ ๆ คือ

3.2.1 ข้อมูลแชนเนล (channel messages)

ข้อมูลเหล่านี้จะถูกส่งไปในแต่ละแชนเนลของเครื่องดนตรีในระบบ และจะมีผลเฉพาะกับเครื่องดนตรีที่รับแชนเนลนั้นเท่านั้น ข้อมูลแชนเนลยังสามารถแบ่งเป็น 2 ประเภทย่อย ๆ อีกคือ

3.2.1.1 ข้อมูลเสียงของแชนเนล (channel voice messages) ประกอบด้วยข้อมูลต่อไปนี้

-ข้อมูลโน้ต เป็นข้อมูลพื้นฐาน ซึ่งบอกว่า คีย์ไหนถูกกดและถูกกดด้วยความเร็ว (velocity) เท่าไร

-เปลี่ยนโปรแกรม (program change) จะทำให้ตัวผ่วงเปลี่ยนเสียง ซินธ์, เปียโนไฟฟ้าจะมีหน่วยความจำที่เก็บเสียงต่าง ๆ ไว้มากมาย เมื่อเลือกการเปลี่ยนโปรแกรม ไบท์ข้อมูลจะเป็นเลขที่โปรแกรม (program number) ใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-เปลี่ยนการควบคุม (control change) จะส่งเมื่อมีการเปลี่ยนตัวควบคุมต่าง ๆ ในระบบมิดิ ที่ไม่ใช่คีย์บอร์ดและพิทช์เบนด์ เช่น มอดดูเลชั่น (modulation) เพเด้ล (pedal), พอร์ทาเม็นโต (portamento) ซึ่งตัวควบคุมเหล่านี้ไม่ใช่จะมีในทุกเครื่องดนตรี บางทีการที่ตัวแม่มีตัวควบคุมนี้ เช่นมีมอดูเลชั่น แต่ตัวพ่วงไม่มี ถึงแม้จะมีการส่งมอดูเลชั่นไป ตัวพ่วงก็จะไม่ตอบสนอง ซึ่งการที่จะรู้ว่า เครื่องดนตรีใดจะตอบสนองตัวควบคุมใดได้ ให้ดูจากแผนภาพมิดิอิมพลีเมนเตชั่น ที่จะกล่าวถึงต่อไป หลังจากส่งไบท์สถานะแล้ว จะส่งไบท์ข้อมูลตามไป 2 ไบท์คือ เลขที่ตัวควบคุม (controller number) และค่าของตัวควบคุม (controller value)

-แรงกดคีย์ (polyphonic key pressure/after-touch) เป็นการกดคีย์ในลักษณะกดแล้วชะงัก ซึ่งก็คือการกดให้แรงขึ้นหลังจากเริ่มกดแล้ว ไบท์ข้อมูลที่ใช้ก็คือ เลขที่โน้ต และค่าการกด (pressure value)

-พิทช์เบนด์ (pitch bend) เป็นตัวควบคุมอีกชนิดหนึ่ง ที่มักพบในซินธ์ราคาแพง ๆ โดยจะมีลักษณะเป็นล้อ (wheel) หรือจอยสติ๊ก (joy stick)

-แรงกดแชนเนล (channel pressure) ใช้บอกความดังของแชนเนล

3.2.1.2 ข้อมูลโหมดของแชนเนล (channel mode messages)

ใช้ในการเปลี่ยนโหมดของตัวพ่วง ซินธ์หรือเปียโนไฟฟ้าของเครื่องจะถูกตั้งให้เป็นโหมด 1 เมื่อเริ่มเปิดเครื่อง เราจึงต้องเซทให้เป็นโหมด 3 เพื่อใช้กับซีเควนเซอร์ ซึ่งซีเควนเซอร์บางตัว จะทำการเปลี่ยนโหมด 3 ให้เองในกรณีใช้มิดิ

3.2.2 ข้อมูลระบบ (system message)

เป็นข้อมูลสำหรับใช้กับทุก ๆ อุปกรณ์ในระบบโดยไม่สนใจว่าขณะนั้นตัวแม่หรือตัวพ่วงถูกกำหนดให้เป็นแชนเนลไหน ข้อมูลนี้จะถูกส่งในเวลาใดก็ได้ ถึงแม้จะเป็นขณะที่กำลังส่งไบท์ของข้อมูลอื่นอยู่ก็ตาม ข้อมูลระบบมีหน้าที่ควบคุมระบบมิดิ เช่น ควบคุมเวลา, การทำงานที่สัมพันธ์กัน (synchronization) โดยเฉพาะในการใช้ซีเควนเซอร์ และดรัมแมชีนร่วมกัน

ยังมีข้อมูลที่เรียกว่า "ข้อมูลพิเศษของระบบ" (system exclusive messages) ซึ่งเป็นข้อมูลพิเศษสำหรับแต่ละบริษัทผู้ผลิต โดยที่แต่ละบริษัทจะมีเลขประจำตัวของตนเอง (Identifier number) ถ้าข้อมูลพิเศษถูกส่งมาโดยมีเลขประจำตัวที่ผิด เครื่องจะไม่สนใจข้อมูลที่ส่งมาระหว่างนั้น

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับรายละเอียดของรหัสต่าง ๆ ดูที่ภาคผนวก ก.

3.3 แผนภาพมิดิอิมพลีเมนเตชั่น (MIDI Implementation Chart)

ถึงแม้ว่า มิดิจะทำให้การสื่อสารระหว่างเครื่องดนตรีเป็นไปได้อย่างกว้างขวาง แต่ก็ไม่ได้หมายความว่า ทุกเครื่องดนตรีจะเข้าใจภาษามิดิได้ทั้งหมด

ตัวอย่างเช่น การต่อซินธ์ที่มีพิทช์เบนด์เข้ากับเปียโนไฟฟ้าที่เปลี่ยนพิทช์ไม่ได้ เปียโนไฟฟ้าก็จะไม่มีผลตอบสนองต่อรหัสของพิทช์เบนด์ ดังนั้นเพื่อให้การติดต่อประสบความสำเร็จ เครื่องดนตรี จึงต้องเข้าใจข้อมูลนั้น ๆ ทั้งคู่

ดังนั้น ในระบบมิดิที่มีเครื่องดนตรีหลาย ๆ ชิ้น เราจึงต้องทำการตรวจสอบข้อมูลที่เครื่องดนตรีทุกตัวสามารถรับ-ส่งได้ เพื่อให้การตรวจสอบเป็นไปโดยรวดเร็ว เครื่องดนตรีแต่ละชนิดจะมีแผนภาพมิดิอิมพลีเมนเตชั่น ซึ่งแสดงถึงข้อมูลที่รับ-ส่งได้

3.3.1 วิธีอ่านแผนภาพมิดิอิมพลีเมนเตชั่น

ทางด้านซ้ายมือของแผนภาพ จะเป็นชื่อประเภทต่าง ๆ ของข้อมูลมิดิ คอลัมน์ส่ง (transmit) และรับ (receive) จะแสดงให้เห็นถึงความสามารถของเครื่องดนตรีชิ้นนั้นในการรับส่งข้อมูลประเภทนั้น ๆ โดยที่ใช้ "O" สำหรับการทำได้ และ "x" คือทำไม่ได้

ประเภทต่าง ๆของข้อมูลมิดิ จะเรียงกันตามนี้ คือ

- แชนเนลพื้นฐาน จะมี 2 คอลัมน์คือ "เปิดเครื่อง" (power on) และ "เซทได้" (can be set) "เปิดเครื่อง" จะบอกว่าแชนเนลไหนที่ถูกเซท เมื่อเราเปิดเครื่อง เครื่องดนตรีบางเครื่องจะสามารถจำแชนเนลที่ใช้ครั้งล่าสุดได้ ซึ่งในกรณีนี้ ในแผนภาพจะบอกว่า "หน่วยความจำยังทำงานอยู่ หลังจากปิดเครื่อง" ส่วน "เซทได้" จะบอกว่า สามารถเซทแชนเนลไหนได้บ้าง นั่นคือ เครื่องดนตรีชิ้นนั้น ๆ สามารถรับรู้หรือส่งแชนเนลไหนได้บ้าง

- โหมด ในกรณีนี้จะมี 3 คอลัมน์ คือ "เปิดเครื่อง" ซึ่งจะบอกว่า เมื่อเปิดเครื่องจะเซทให้เป็นโหมดใด "ข้อมูล" ที่บอกว่าเครื่องดนตรีสามารถรับข้อมูลการเปลี่ยนโหมดได้หรือไม่ "อื่น ๆ" ที่บอกว่า เครื่องดนตรีสามารถรับข้อมูลซึ่งจะทำการเปลี่ยนเป็นโหมดพิเศษอื่นได้หรือไม่ ซึ่งโหมดพิเศษนี้จะกล่าวถึงในภายหลังในคอลัมน์นี้

- เลขที่โน้ต จะบอกช่วงของโน้ตที่สามารถรับหรือส่งได้ โน้ตที่ส่งมักจะมีเลขที่โน้ตอยู่ในช่วงของคีย์ที่มันมี ในขณะที่ในการรับอาจจะรับได้ในช่วงกว้างกว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ความเร็ว จะมี 2 คอลัมน์คือ เปิดโน้ต และปิดโน้ต คอลัมน์นี้จะแสดงว่า เครื่องดนตรีสามารถส่งหรือรับความเร็วได้หรือไม่ในการเปิด-ปิดโน้ต ถ้าเป็น"x"ในคอลัมน์ใดคอลัมน์หนึ่ง นั่นไม่ได้หมายความว่า เครื่องดนตรีไม่สามารถรับการเปิดปิดโน้ตได้ แต่หมายความว่า มันไม่สามารถแยกความเร็วของการกดโน้ตได้

- แรงกด ใช้บอกว่า สามารถรับข้อมูลแรงกดคีย์ได้หรือไม่ โดยจะแบ่งเป็นแรงกดของแชนเนล (ค่าหนึ่งต่อหนึ่งมิดีแชนเนล) และแรงกดโพลีโฟนิค (แต่ละค่าสำหรับแต่ละโน้ต)

- พิทช์เบนด์ รับข้อมูลพิทช์เบนด์ได้หรือไม่

- การเปลี่ยนแปลงตัวควบคุม ข้อมูลนี้สำคัญมากในการต่อซินธ์ 2 ตัวที่แตกต่างกัน หรือซินธ์กับเปียโน เพื่อตรวจสอบตัวควบคุมของเครื่องดนตรีทั้งสองว่ามีตรงกันหรือไม่

- การเปลี่ยนโปรแกรม จะบอกว่าเปลี่ยนได้หรือไม่ และเลขที่โปรแกรมที่ใช้คืออะไร

- ข้อมูลพิเศษ ใช้บอกว่าข้อมูลประเภทไหนบ้างที่สามารถรับ-ส่งทางข้อมูลพิเศษของระบบ

- ข้อมูลร่วม (system common) ใช้บอกว่าสามารถเข้าใจข้อมูลร่วมของระบบหรือไม่ นั่นคือ เข้าใจข้อมูลตัวชี้ตำแหน่งเพลง, เพลงที่เลือก หรือไม่

-เวลาจริงของระบบ (system real time) หมายถึงข้อมูลที่ทำให้เครื่องดนตรีต่าง ๆ ทำงานสัมพันธ์กันโดยมิดี ถ้าเครื่องดนตรีเข้าใจ ข้อมูลนาฬิกา (clock) มันจะสามารถเล่นในเวลาตรงกับเครื่องอื่น หรือถ้าเข้าใจข้อมูลคำสั่ง (command) มันจะรู้ว่าเมื่อไรที่เริ่มเมื่อไรที่จะหยุด

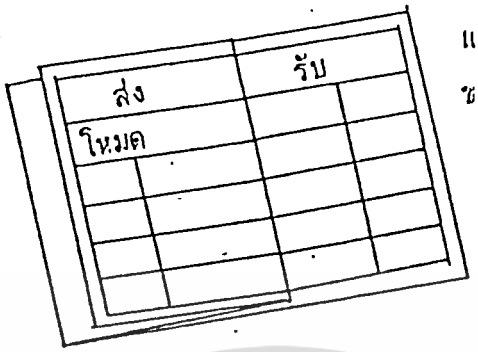
- ข้อมูลช่วย (AUX message) ใช้บอกว่าเครื่องดนตรีสามารถรับข้อมูลที่จะช่วยหลีกเลี่ยงปัญหาของมิดีหรือไม่

(ข้อมูลพิเศษ, ข้อมูลร่วม, เวลาจริง, ข้อมูลช่วย ดูรายละเอียดจากภาคผนวก ก.)

ดังนั้น เมื่อเราทำการต่อเครื่องดนตรีหลาย ๆ เครื่อง เราเบื้องต้นทำการตรวจคอลัมน์ที่ตรงกันของแผนภาพของเครื่องดนตรีทั้งหมด เพื่อดูว่าการสื่อสารนั้นเป็นไปได้หรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แผนภาพมิติอิมพลีเมนเตชั่น
ของเครื่องคนตรีตัวแม่



แผนภาพมิติอิมพลีเมนเตชั่น
ของเครื่องคนตรีตัวพ่วง

รูปที่ 2 วิธีเทียบแผนภาพ

เนื่องจาก แผนภาพนี้มีรูปแบบเป็นมาตรฐาน เราจึงทำการเทียบการทำงานได้ง่าย โดยเปรียบคอลัมน์ "ส่ง" ของตัวแม่ กับคอลัมน์ "รับ" ของตัวพ่วง ถ้ามีตัวใดตัวหนึ่งเป็น "X" ดังนั้นข้อมูลในคอลัมน์นั้นจะติดต่อไม่ได้

ตัวอย่างของแผนภาพมิติอิมพลีเมนเตชั่น ดูจากภาคผนวก ข.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

วงจรมิดีอินเตอร์เฟสและซอฟท์แวร์

4.1 วงจรมิดีอินเตอร์เฟส

วงจรมิดีอินเตอร์เฟส เป็นการอินเตอร์เฟสประเภท อะซิงโครนัสแบบอนุกรม (Serial Asynchronous) ซึ่งมีความเร็วของการส่งผ่านข้อมูลเป็นมาตรฐาน คือ 31250 bps. เราสามารถแบ่งการทำงานของวงจรออกเป็น 5 ส่วนใหญ่ ๆ คือ

4.1.1 ส่วนที่ติดต่อกับซินธิไซเซอร์

ในส่วนนี้ จะเป็นส่วนของการติดต่อกับมิดีเคเบิ้ล ซึ่งประกอบไปด้วย 2 ส่วนย่อย คือ

4.1.1.1 ส่วนรับข้อมูล ซึ่งต่อมาจากมิดีเอาท์ของซินธิไซเซอร์ ส่วนนี้จะเป็นการนำเอาข้อมูลที่ส่งผ่านไปให้ Z-80 SIO เพื่อส่งต่อไปให้คอมพิวเตอร์ประมวลผลต่อไป ในส่วนนี้จะมีการต่อวงจรโดยใช้ออปโตไอโซเลเตอร์ (opto-isolator) เพื่อแยกสัญญาณที่รับมาออกจากวงจรภายในของเครื่องอย่างเด็ดขาด ซึ่งจะมีผลให้สัญญาณไม่เกิดการรบกวนกัน และขจัดปัญหาการยัมหรือการรบกวนที่จะเพี้ยวนำเข้าสู่ระบบ ในที่นี้ใช้ออปโตไอโซเลเตอร์เบอร์ TIL-111 แล้วสัญญาณที่ได้ออกมาจะถูกบัฟเฟอร์โดยอินเวอร์เตอร์ 2 ตัว ก่อนส่งไปขา RxDA ของ SIO

4.1.1.2 ส่วนส่งข้อมูล จะทำการส่งข้อมูลที่ต้องการจากคอมพิวเตอร์ ผ่านทาง SIO ไปออกที่มิดีเคเบิ้ล เพื่อไปเข้ามิดีอินของซินธิไซเซอร์ ข้อมูลจะออกจากขา TxDA แล้วถูกบัฟเฟอร์โดยอินเวอร์เตอร์ ก่อนจะผ่านตัวต้านทานขนาด 220 โอห์ม เพื่อให้มีค่าตรงกับอิมพีแดนซ์ของมิดีเคเบิ้ล

4.1.2 ส่วนของการแปลงข้อมูลจากอนุกรมเป็นขนานและขนานเป็นอนุกรม

เนื่องจากมิดีรับส่งข้อมูลเป็นอนุกรม ความเร็ว 31,250 bps และมีการส่งแบบอะซิงโครนัส โดยมี 1 บิทเริ่มต้น (start bit), 8 บิทข้อมูล(data bit) และ 1 บิทหยุด (stop bit) แต่คอมพิวเตอร์รับส่งข้อมูลเป็นแบบขนาน จึงมีการนำ Z-80 SIO มาใช้แปลงลักษณะการส่งข้อมูลโดยจะทำการรับส่งข้อมูลกันในเซคชั่น A ของ SIO

การส่งข้อมูลไปมิดีเคเบิ้ล จะรับข้อมูลจากคอมพิวเตอร์ที่เข้าขา D0-D7 ของ SIO แล้วนำข้อมูลไปออกที่ขา TxDA ในลักษณะของอนุกรม สำหรับการรับข้อมูลของ SIO จากมิดีเค

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เบิ้ลจะรับข้อมูลเข้ามาทางขา RxDA ในลักษณะอนุกรม เมื่อข้อมูลมาครบแล้ว จะส่งออกไปที่คอมพิวเตอร์ทางขา D0-D7 นอกจากนั้น ยังมีการส่งสัญญาณอินเตอร์รัพท์ ไปบอกคอมพิวเตอร์ด้วยว่าได้รับข้อมูลเข้ามาแล้ว ให้คอมพิวเตอร์มาทำการอ่านเอาไป ขาอินเตอร์รัพท์ของ SIO จะต่อไปที่ขา IRQ2 (อินเตอร์รัพท์ 10) ของสล็อต โดยผ่าน IC เบอร์ 74244 เพื่อขับกระแสให้เพียงพอ

สำหรับการทำงานของ SIO จะถูกโปรแกรมโดยซอฟท์แวร์เพื่อให้มีการทำงานตามกล่าวข้างต้น

4.1.3 ส่วนของ 8253 PIT

จะทำหน้าที่นับเวลาในตอนบันทึกหรือเล่นกลับของเพลง โดยใช้การทำงานใน 2 โหมดคือ เคาน์เตอร์ 0 (Counter 0) จะทำงานในโหมด 2 คือ เป็นตัวหารความถี่เพื่อใช้เป็นสัญญาณนาฬิกาสำหรับเคาน์เตอร์ 1 ดังนั้น เคาน์เตอร์ 0 จะเป็นตัวควบคุมจังหวะ (tempo) ของการบันทึกหรือเล่นกลับ และ เคาน์เตอร์ 1 จะทำงานในโหมด 0 คือนับลง แล้วมีการอ่านค่าที่กำลังนับอยู่เป็นระยะ ๆ

การทำงานของ 8253 จะถูกโปรแกรมโดยซอฟท์แวร์เช่นกัน

4.1.4 ส่วนผลิตสัญญาณนาฬิกาของระบบ

มีการใช้ออสซิลเลเตอร์ขนาด 2 MHz เพื่อเป็นสัญญาณนาฬิกา ให้กับ SIO และ 8253 โดย SIO จะทำการหารความถี่ด้วย 64 เพื่อให้ได้ค่า 31,250 bps ตามต้องการ ขณะที่ 8253 จะทำการหารความถี่ ด้วยค่าในเคาน์เตอร์ 0 เพื่อให้ได้จังหวะตามที่ต้องการ

4.1.5 ส่วนเลือกพอร์ท

เราใช้ IC เบอร์ 74138 มาเป็นตัวดีโค้ดเดอร์ (Decoder) โดยปกติแล้ว ในการเลือกแอดเดรสพอร์ท ของ IPM PC จะใช้แอดเดรส 9 บิท แต่ในที่นี้ ใช้ถึง 15 บิท เพื่อลดปัญหาการเกิดแอดเดรสที่ซ้ำซ้อน กับอินพุท/เอาท์พุท ตัวอื่นในระบบ นอกจากนี้ ยังมีการนำเอาสัญญาณ SYSIORD (I/O Read) และ SYSIOWR (I/O Write) มารวมกัน เพื่อช่วยในการเลือกชิปด้วย

ในวงจรนี้ พอร์ทสถานะของแชนเนล A ของ SIO คือ พอร์ทแอดเดรสที่ OFFA2H และ

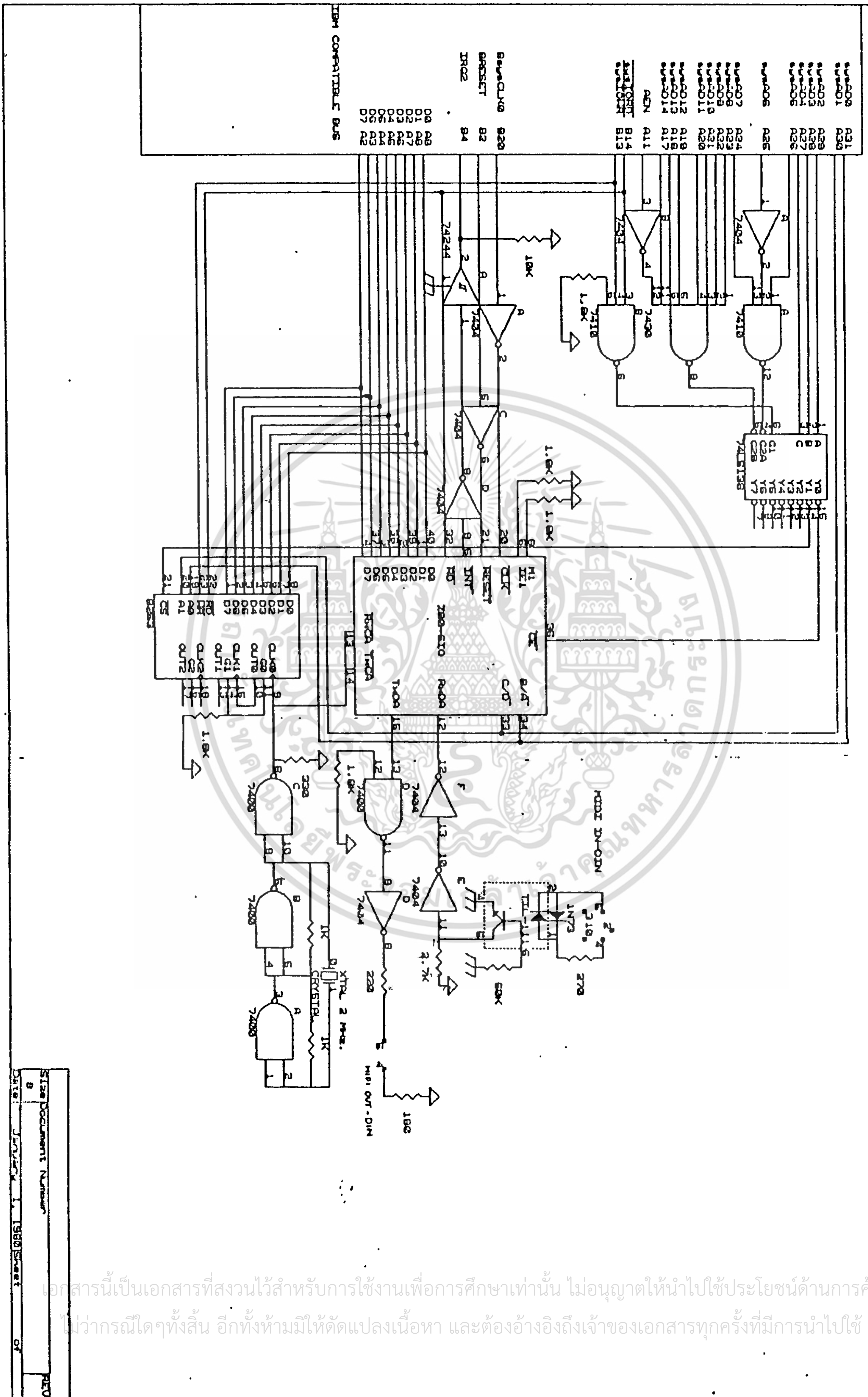
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พอร์ทข้อมูลแชนเนล A ของ SIO คือพอร์ท OFFA0H

พอร์ทของเคาน์เตอร์ 0 ของ 8253 คือ OFFA4H และเคาน์เตอร์ 1 คือ OFFA5H พอร์ทควบคุมของ 8253 คือ OFFA7H

รูปวงจรแสดงในรูปที่ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 ซอฟท์แวร์

สำหรับซอฟท์แวร์ของโครงงานนี้ เป็นการทำโปรแกรมซีเควนเซอร์ ซึ่งสามารถอัดเพลงได้ 2 แทรค 4 แชนเนล (บางทีจะเรียกแต่ละแทรคแต่ละแชนเนลว่า บัฟเฟอร์ ก็ได้) ความสามารถของโปรแกรมนี้ คือ ทำการอัดเพลงได้ทีละแทรคละ 4 แชนเนล และในการเล่นกลับออกมา (Play back) สามารถเล่นกลับได้ทั้ง 2 แทรค 4 แชนเนลพร้อมกันเลย นอกจากนี้ยังสามารถอัดเพลงลงบัฟเฟอร์หนึ่งในขณะที่กำลังเล่นเพลงในบัฟเฟอร์อื่นอยู่ก็ได้ เพื่อความพร้อมเพรียงเข้าจังหวะกันของบทเพลง และนอกจากนั้น ยังสามารถกำหนดให้เล่นเพลงออกไปยังแชนเนลของซินธิไซเซอร์ตามต้องการได้

โปรแกรมนี้ เขียนด้วย เทอร์โบ ซี เวอร์ชั่น 2.0 (TURBO C Version 2.0) และ เทอร์โบแอสเซมบลี้ (TURBO ASSEMBLY) ซึ่งประกอบไปด้วย 3 ส่วนใหญ่ ๆ คือ

4.2.1 ส่วนรับข้อมูล MIDI

ส่วนรับข้อมูลมีการทำงาน คือ รับข้อมูลที่เข้ามาทางมิดิอิน และแปลงข้อมูลดิบเหล่านั้นให้อยู่ในรูปแบบที่กำหนดไว้เพื่อเก็บลงในหน่วยความจำ (ซึ่งข้อมูลที่เก็บนั้น รวมไปถึงค่าของเวลาที่มีการกดให้เต็มตัวนั้นด้วย)

ในส่วนรูทีนการรับข้อมูล คือส่วนของการบันทึกเพลงซึ่งอยู่ที่รูทีน recbuff() ซึ่งจะเริ่มทำงานโดยให้พอยเตอร์ของแต่ละบัฟเฟอร์ชี้ไปที่ตอนต้นของบัฟเฟอร์นั้น จากนั้นจะมีการถามว่าต้องการเรคอร์ดไปที่แทรคไหนแชนเนลไหน ซึ่งถ้าบัฟเฟอร์นั้นมีข้อมูลอยู่แล้ว จะถามเพื่อความแน่ใจว่ายังต้องการเรคอร์ดในบัฟเฟอร์นี้อยู่หรือไม่ เมื่อผ่านขั้นตอนนี้แล้ว ต่อไปจะแสดงข้อมูลของแต่ละบัฟเฟอร์บนหน้าจอ เพื่อให้เราเลือกข้อกำหนดต่าง ๆ เช่น ต้องการให้บัฟเฟอร์ไหนทำงาน (enable) หรือ ต้องการให้บัฟเฟอร์ที่จะเล่น ออกไปเล่นที่แชนเนลไหน

ต่อไปจะทำการ setrec() ซึ่งกำหนดค่าต่าง ๆ ที่ต้องใช้รวมทั้งกำหนด setint() คือ การนำโปรแกรมตอบสนองการอินเตอร์รัพท์ไปไว้ที่ตำแหน่งของอินเตอร์รัพท์ 10 และเซตค่าต่าง ๆ ของ SIO แล้วทำการอีนาเบิ้ลอินเตอร์รัพท์ จากนั้นจะเริ่มทำรูทีน play() ซึ่งถ้ามีการกดซินธิไซเซอร์ระหว่างที่ทำ play() จะมีการกระโดดไปทำที่โปรแกรมตอบสนองการอินเตอร์รัพท์ ซึ่งการทำงานในลักษณะนี้จะทำให้สามารถมีการรับข้อมูลได้ในขณะเดียวกับที่กำลังเล่นอยู่ด้วย

สำหรับรูทีน inton() และ intoff() จะทำการอีนาเบิ้ลและดิสเอเบิ้ลอินเตอร์รัพท์โดยการอีนาเบิ้ล และ ดิสเอเบิ้ล จาก SIO

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมการตอบสนองอินเตอร์รัพท์นี้ อยู่ที่ฟังก์ชัน Rxint ซึ่งเมื่อเริ่มเข้าโปรแกรมจะทำการดิสเอเบิ้ลอินเตอร์รัพท์ของไทม์เมอร์ (timer) ของระบบ เนื่องจากเป็นตัวเดียวที่มีไพโอริตี้สูงกว่าอินเตอร์รัพท์ 10 เพื่อให้การทำงานของรูทีน Rxint ไม่มีผลกระทบ และก่อนออกจากโปรแกรมจะต้องทำการอีนาเบิ้ล ไทม์เมอร์ เพื่อให้การทำงานเป็นไปอย่างปกติ นอกจากนั้น ก่อนการออกจากโปรแกรมต้องมีการส่งสัญญาณ EOI (End of interrupt) ให้แก่ 8259 PIC (Programable interrupt controller) ด้วย

Rxint จะทำการบันทึกข้อมูลต่าง ๆ ของมิดี ดังนี้ คือ

- เปิดโน้ต
- ปิดโน้ต
- เปลี่ยนโปรแกรม
- เปลี่ยนตัวควบคุม
- ความดังของแชนเนล
- พิทช์เบนด์

ซึ่งรูปแบบการเก็บของข้อมูลแต่ละชนิดในหน่วยความจำเป็นดังนี้

ข้อมูล	ไบท์แรก	ไบท์สอง	ไบท์สาม	ไบท์สี่	ไบท์ห้า
เปิดโน้ต	00VVVVVV	LLLLLLLL	MMMMMMMM	1nnnnnnn	-
ปิดโน้ต	00VVVVVV	LLLLLLLL	MMMMMMMM	0nnnnnnn	-
เปลี่ยนตัวควบคุม	11000000	LLLLLLLL	MMMMMMMM	0NNNNNNN	0ccccccc
เปลี่ยนโปรแกรม	01000000	LLLLLLLL	MMMMMMMM	0ppppppp	-
ความดังแชนเนล	01000001	LLLLLLLL	MMMMMMMM	0aaaaaaa	-
พิทช์เบนด์	10WWWWWW	LLLLLLLL	MMMMMMMM	-	-

เมื่อ VVVV เป็นความเร็วในการกดคีย์

LLLL เป็นไบท์ต่ำของเวลา

MMMM เป็นไบท์สูงของเวลา

nnnn เป็นเลขที่โน้ต

NNNN เป็นเลขที่ตัวควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

cccc เป็นค่าของตัวควบคุม

pppp เป็นโปรแกรมใหม่

aaaa เป็นค่าความดังของแชนเนล

wwww เป็นค่าของพิทช์เบนด์

สำหรับการบันทึกเวลานั้น ซินธิไซเซอร์จะส่งรหัสของสัญญาณนาฬิกาของระบบ (System Timing) ออกมา ซึ่งในการรับสัญญาณนาฬิกาครั้งหนึ่งจะไปทำการลดค่าตัวแปรตัวนับทำดังนี้ไปเรื่อย ๆ ถ้ามีการส่งรหัสตัวอื่น เช่น เปิดโน้ต , ปิดโน้ต รูทีนจะไปดูว่า ในขณะนั้นค่าของตัวแปรตัวนับมีค่าเท่าไร และจะเก็บค่าเหล่านั้นไว้ในบัฟเฟอร์ เพื่อใช้ในการเล่นกลับภายหลัง

ในการจะเปลี่ยนจังหวะของเวลาให้เร็วขึ้นหรือช้าลง ทำโดยการกดปุ่ม "จังหวะ" (tempo) บนซินธิไซเซอร์ ซึ่งจะทำการเปลี่ยนความถี่ของสัญญาณนาฬิกาที่ส่งออกมา (รหัส 0F8H)

เมื่อเลิกทำการบันทึกเพลงแล้ว จะมีการเก็บค่า 0FFH ไว้ที่ไบท์สุดท้ายของเพลง เพื่อเป็นการบอกว่า เพลงได้จบลงแล้วในบัฟเฟอร์นั้น ๆ

4.2.2 ส่วนส่งข้อมูลของมิดิ

ส่วนส่งข้อมูลจะมีหลักการทำงานโดยเปรียบเทียบเวลาในขณะนั้นกับเวลาที่เก็บไว้ เมื่อเวลาใกล้เคียงกันแล้ว ก็ส่งรหัสที่เหมาะสมออกมา

รูทีนในการเล่นจะเริ่มทำไปที่ pbuff() โดยทำการกำหนดพอยน์เตอร์ของบัฟเฟอร์ให้ชี้ไปที่จุดเริ่มต้นของบัฟเฟอร์ก่อน จากนั้นก็แสดงข้อมูลของบัฟเฟอร์ที่หน้าจอ เพื่อให้เลือกช่วงกำหนดตามที่ต้องการ จากนั้น จึงเข้าสู่ setrec() เพื่อทำการกำหนดค่าเริ่มต้นให้กับตัวแปรที่จำเป็นต้องใช้ แล้วจึงไปทำรูทีน play()

ในรูทีน play() จะทำการวนดูบัฟเฟอร์ไปทีละ 1 ตัวโน้ต โดยจะตรวจสอบว่า บัฟเฟอร์นั้น ๆ อีกาเบิ้ลหรือไม่ ถ้าอีกาเบิ้ลก็จะทำการเรียก playfrom() เพื่อส่งค่าตัวโน้ตออกไปที่มิดิเอาท์ ทำจนกระทั่ง แทรค 0 แชนเนล 0 ซึ่งเป็นคอนดัคเตอร์แทรค (conductor track) ถูกเล่นหมดไปแล้วทุกตัวโน้ต หรือบัฟเฟอร์ถูกเล่นหมดแล้ว ก็จะเลิกทำงาน

ในการส่งตัวโน้ตออก ของรูทีน playfrom() จะทำการเปรียบเทียบเวลาที่เก็บไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กับเวลาที่เป็นอยู่ขณะนั้น ว่า ถ้าค่าเวลาขณะนั้นเท่ากับหรือน้อยกว่าเวลาที่เก็บไว้ รูทีน playfrom() จะพิจารณา 2 บิทบนของไบท์แรก เพื่อพิจารณาว่า เป็นข้อมูลอะไร จะได้ส่งรหัสออกไปได้ถูกต้อง สำหรับรูทีนที่ทำการส่งข้อมูลนั้น เขียนด้วยภาษาแอสเซมบลี้ เพื่อความรวดเร็วของการทำงาน

4.2.3 ส่วนอื่น ๆ

เป็นส่วนของรูทีนอื่น ๆ เช่น

- การจองเนื้อที่ของหน่วยความจำเพื่อใช้เป็นบัฟเฟอร์สำหรับเก็บข้อมูล โดยจะจองเนื้อที่ทีละ 20K จนกว่าหน่วยความจำจะไม่พอ หรือมีการจองหมดทุกบัฟเฟอร์แล้ว

- การลบแทรก ทำโดยการทำให้ขนาดของบัฟเฟอร์ที่อยู่ในตัวแปร ssize มีค่าเป็น 0 และ เปลี่ยนค่าไบท์แรกของบัฟเฟอร์เป็น OFFH ซึ่งคือ จบเพลง

- การเก็บลงไฟล์ จะทำการเก็บบัฟเฟอร์ที่ต้องการลงไฟล์ที่มีส่วนขยาย (extension) ของไฟล์เป็น .sng โดยข้อมูลที่เก็บจะเก็บตามรูปแบบเดิมที่อยู่ในหน่วยความจำ โดยจะเก็บไฟล์ละ 1 แทรค

- การเรียกจากไฟล์ จะทำการเรียกได้เฉพาะไฟล์ที่มีส่วนขยายเป็น .sng โดยจะเรียกไปไว้บัฟเฟอร์ของแทรคที่ต้องการ

สำหรับตัวโปรแกรมที่เขียนรวมทั้งโฟลว์ชาร์ตและคู่มือการใช้โปรแกรม ได้แสดงอยู่ในภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

สรุป

ในโครงงานนี้มีการทำวงจรอินเตอร์เฟส ซึ่งสามารถใช้งานในชั้นเบื้องต้นได้ดีพอสมควร นอกจากนี้ยังมีการเขียนซอฟท์แวร์โปรแกรมซีเควนเซอร์ ซึ่งมีความสามารถอัดเพลงได้ 2 แทรค 4 แชนเนล และสามารถเล่นกลับได้ในขณะที่อัดเพลง การที่ต้องทำบัฟเฟอร์ไว้เป็นจำนวนมากก็เพื่อสะดวกในการอีดิตเพลง นั่นคือ ถ้าเราอัดเพลงทั้งเพลงไว้ในบัฟเฟอร์เดียว ถ้ามีการกดคีย์ผิดหรือเกิดผิดพลาด ต้องการอัดใหม่ เราจะต้องอัดเพลงนั้นใหม่ทั้งเพลง แต่ถ้าเราแบ่งส่วนต่าง ๆ ของเพลงไว้ในคนละบัฟเฟอร์ เราจะแก้ไขเฉพาะบางส่วนของเพลงเท่านั้น ซึ่งประหยัดเวลาและลดความยุ่งยาก

สำหรับปัญหาที่เกิดและแนวทางพัฒนาต่อ จะขอกล่าวต่อไป

5.1 ปัญหาที่เกิดขึ้นระหว่างการทดลอง

5.1.1 เป็นปัญหาที่เกิดขึ้นทางฮาร์ดแวร์ในส่วนของการอินเตอร์รัพท์ ซึ่งในครั้งแรกมีการต่อขาอินเตอร์รัพท์จาก SIO ผ่านทางอินเวอร์เตอร์ตัวหนึ่ง แล้วนำเอาท์พุทไปต่อกับขา IRQ2 ของสล็อต IBM (เนื่องจากอินเตอร์รัพท์จาก SIO จะเปลี่ยนสัญญาณจาก 1 เป็น 0 (ทริกที่ขาลง) แต่อินเตอร์รัพท์ของสล็อตเปลี่ยนสัญญาณจาก 0 ไป 1 (ทริกที่ขอบขาขึ้น) ดังนั้น จึงต้องมีการต่ออินเวอร์เตอร์) เมื่อลองจับสัญญาณอินเตอร์รัพท์ที่ขาของ SIO พบว่ามีการเปลี่ยนจาก 5 โวลท์ เป็น 0.8 โวลท์ ซึ่งเปลี่ยนจาก 1 เป็น 0 จริง (เนื่องจากขาอินเตอร์รัพท์ของ SIO เป็น โอเพ่น เดรน (open drain) เพื่อให้สัญญาณในขณะที่ไม่มีอินเตอร์รัพท์อยู่ที่ระดับ 1 จริง จึงต้องทำการพูลอัพ (pull up) ไว้) แต่เมื่อลองจับสัญญาณที่ขา IRQ2 ของสล็อต พบว่า มีการเปลี่ยนสัญญาณจาก 2.4 โวลท์เป็น 5 โวลท์ ซึ่งสัญญาณ 2.4 โวลท์นี้อยู่ในช่วงของสัญญาณที่ไม่ทราบว่ามีระดับเป็นเท่าไร ดังนั้น จึงไม่มีการอินเตอร์รัพท์เกิดขึ้นเพราะไม่ถือว่ามีการเปลี่ยนระดับสัญญาณจาก 0 เป็น 1

ข้อสันนิษฐานของความผิดพลาดคือ อาจเป็นไปได้ว่าสล็อตมีโหลดมาก ทำให้กระแสไม่พอ ดังนั้น จึงมีการแก้ปัญหาโดยใช้ IC 74244 มาต่อก่อนเข้าสล็อต เพื่อทำการไดรฟ์กระแส หลังจากนั้น ลองจับสัญญาณดูพบว่ามีการเปลี่ยนระดับแรงดัน จาก 0.8 โวลท์เป็น 5 โวลท์ ซึ่งเป็นการเปลี่ยนจาก 0 เป็น 1 ทำให้มีการอินเตอร์รัพท์เกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.2 เป็นปัญหาที่เกิดขึ้นจากเคาน์เตอร์ ซึ่งตอนแรกในการเก็บเวลาของการบันทึกเพลง มีการนำเคาน์เตอร์มาใช้ โดยใช้ PIT 8253มาเป็นเคาน์เตอร์นับ โดยใช้เคาน์เตอร์ 2 ตัว คือ เคาน์เตอร์ 0 และ เคาน์เตอร์ 1 ซึ่งมีหน้าที่และโหมดการทำงานดังนี้ คือ

เคาน์เตอร์ 0 ทำหน้าที่หารความถี่ 2 MHz ให้เหลือความถี่ที่ต้องการ ซึ่งจะนำมาเป็นสัญญาณนาฬิกาให้เคาน์เตอร์ 1

เคาน์เตอร์ 1 ทำงานในโหมด 0 คือ ทำการนับลง โดยมีค่าที่เริ่มต้นนับ คือ OFFFFH

เมื่อทำการกดคีย์เพื่อบันทึกเพลง จะมีการอ่านค่าของเคาน์เตอร์ 1 ว่า ขณะนั้นนับถึงค่าเท่าไร แล้วนำค่านั้นมาเก็บในบัฟเฟอร์ โดยจะอ่านค่าไบท์ต่ำ (LSB) ก่อนแล้วจึงอ่านไบท์สูง (MSB) ซึ่งในเวลาเล่นกลับจะทำการอ่านค่าของเคาน์เตอร์ 1 มาเรื่อย ๆ จนได้ค่าใกล้เคียงกับค่าที่เก็บในบัฟเฟอร์ จึงส่งโน้ตตัวนั้นออกมา

แต่ปัญหาที่เกิดขึ้นคือ การอ่านค่าของเคาน์เตอร์ 1 นั้น ในตอนแรกอ่านไบท์ต่ำก่อนแล้วจึงอ่านไบท์สูง ในช่วงแรก ๆ ของการอ่านค่ายังไม่ผิด แต่เมื่ออ่านไปได้สักครู่จะอ่านมาได้เป็นไบท์ต่ำ ทั้งคู่ ต่อมาจะอ่านได้ไบท์สูงมาก่อน แล้วตามด้วยไบท์ต่ำ ซึ่งเป็นภาวะผิดพลาด

สำหรับต้นเหตุนั้น คิดว่ามาจากการที่อ่านค่าจากเคาน์เตอร์ไม่เหมือกับค่าที่แท้จริงที่เคาน์เตอร์ส่งออกมาที่แลทช์ อาจจะพลาดการอ่านบางค่า ทำให้บางไบท์ของเคาน์เตอร์หายไป ทำให้ได้ข้อมูลที่ผิด ๆ

สำหรับปัญหานี้ ยังไม่สามารถแก้ที่ต้นเหตุ ดังนั้น จึงมีการแก้ไขที่ปลายเหตุ คือ มีการเปลี่ยนการบันทึกเวลา โดยใช้สัญญาณนาฬิกาที่ถูกส่งออกมาจากซินธิไซเซอร์หลังจากกดปุ่มสตาร์ท (start) ของการเล่นจังหวะซินธิไซเซอร์ เมื่อมีสัญญาณนาฬิกาครั้งหนึ่ง จะไปทำการเรียกโปรแกรมการสนองตอบอินเตอร์รัพท์ ซึ่งจะทำการลดค่าของเคาน์เตอร์ซึ่งในตอนแรกตั้งค่าไว้ OFFFFH จะทำดังนี้ไปเรื่อย ๆ ถ้ามีการกดคีย์ จะมีการเก็บค่าของเคาน์เตอร์ในบัฟเฟอร์ เพื่อประโยชน์ในการเล่นกลับในภายหลัง ซึ่งจะเล่นออกมาเมื่อค่าของเคาน์เตอร์มีค่าใกล้เคียงกับค่าในบัฟเฟอร์ที่เก็บเอาไว้

5.1.3 เป็นปัญหาที่เกิดจากการรับข้อมูล นั่นคือ ถ้ารับข้อมูลทีละน้อย ๆ เช่น 1 ตัวโน้ต การทำงาน คือ ทำการแปลงรหัสเป็นรูปแบบที่เก็บในหน่วยความจำ จะไม่ผิดพลาด แต่ถ้ามีการรับข้อมูลมาทีละหลายตัวโน้ต เช่น การกดคอร์ด ซึ่งกดทีละ 5 ตัวโน้ต จะมีข้อมูลบางไบท์ที่หายไป อาจจะเนื่องมาจาก รูทีนตอบสนองอินเตอร์รัพท์ทำงานช้า จึงไม่ทันกับข้อมูลที่เข้ามา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารนี้ทุกครั้งที่มีการนำไปใช้

ทำให้มีข้อมูลบางตัวหายไป

5.2 <u>แนวทางพัฒนาต่อ</u>

การทำงานของโปรแกรมซีเควนเซอร์ในโครงงานนี้ ยังนับว่าไม่ค่อยจะสมบูรณ์นัก เนื่องจากการใช้งานยังไม่สะดวกนัก โดยเฉพาะส่วนของเวลา นั่นคือ ก่อนจะบันทึกเพลงหรือเล่นกลับต้องกดปุ่ม สตาร์ท เพื่อให้เครื่องซินธิไซเซอร์ส่งสัญญาณนาฬิกาออกมา เมื่อต้องการหยุดบันทึกเพลงหรือเล่นกลับ ต้องกดปุ่ม สต็อป เพื่อให้หยุดส่งสัญญาณ ซึ่งพบว่าเป็นการไม่สะดวกอย่างยิ่ง ที่เวลาจะเล่นต้องกดปุ่มต่าง ๆ เหล่านี้ เราควรจะให้มันเล่นกลับออกมาโดยอัตโนมัติ แต่เนื่องจากคาสิโอ CT-640 รุ่นที่ใช้อยู่นี้ไม่สามารถรับรู้รหัสของ สตาร์ท , สต็อป ได้ ดังนั้น เครื่องจึงไม่สามารถจะผลิตสัญญาณนาฬิกาออกมาได้ นอกจากเราจะกดปุ่ม สตาร์ท เท่านั้น (เราไม่สามารถส่งรหัสดังกล่าวออกไปได้ เพราะถึงส่งออกไปเครื่องก็ไม่รู้รู้ คือ เราต้องใช้มือกดปุ่ม ไม่สามารถจัดการโดยโปรแกรมได้)

ข้อเสนอแนะของการพัฒนาต่อในกรณีนี้ คือ ควรจะหาวิธีกำหนดเวลาของการบันทึกเพลงและเล่นกลับใหม่ โดยไม่ให้ขึ้นอยู่กับเครื่อง ทำให้ไม่ต้องทำอะไรมากก่อนที่จะเวฟอร์มหรือเล่น

นอกจากนี้ โปรแกรมที่เขียนขึ้น ยังมีการทำงานบางส่วนที่ติดอยู่กับการใช้งานกับเครื่องคาสิโอ CT-640 นั่นคือ มีรหัสบางตัวที่ CT-640 ไม่สามารถส่งหรือรับได้ เช่น ข้อมูลอินเสทของระบบ , แรงกดคีย์ , พิทช์เบนด์ ทำให้ไม่สามารถทดสอบได้ว่าการทำงานของรหัสต่าง ๆ เหล่านี้ให้ผลเป็นอย่างไร จึงควรทำการทดลองต่อไป โดยอาจจะนำซินธิไซเซอร์รุ่นอื่น ยี่ห้ออื่นมาลองใช้แทน หรือลองนำซินธิไซเซอร์ 2 เครื่องมาต่อกันโดยผ่านทางอินเตอร์เฟส หรือทำการต่อเพิ่มมิดิทรูที่วงจรอินเตอร์เฟส ทำให้สามารถเล่นซินธิไซเซอร์ได้ครั้งละหลาย ๆ เครื่อง ลองทดลองดูในรูปแบบต่าง ๆ เพื่อประยุกต์นำมิดิไปใช้งานจริง

ข้อเสนอแนะสุดท้ายคือ ในโปรแกรมมิดิ ควรจะมีความสามารถอื่น ๆ นอกเหนือจากที่เขียนนี้ เช่น มีความสามารถในการอีดิตบทเพลงต่าง ๆ เช่น คัดลอก (Copy) บางส่วนของเพลง เลือกเล่นเพลงเฉพาะในบางท่อน ลบเพลงเฉพาะในบางส่วน เป็นต้น , ความสามารถในการแก้ไขเองโดยอัตโนมัติ (Auto-correction) คือ การทำให้ไม่เป๊ะเกินจังหวะ (ซึ่งผู้ที่อัดอาจจะเล่นไม่ตรงจังหวะพอดี) ซึ่งจะทำให้การเล่นเป็นไปอย่างนุ่มนวลยิ่งขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

ตารางรหัสมิติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TABLE I

SUMMARY OF STATUS BYTES

STATUS D7—D0	# OF DATA BYTES	DESCRIPTION
Channel Voice Messages		
1000nnnn	2	Note Off event
1001nnnn	2	Note On event (velocity=0 Note Off)
1010nnnn	2	Polyphonic key pressure/aftertouch
1011nnnn	2	Control change
1100nnnn	1	Program change
1101nnnn	1	Channel Pressure (Aftertouch)
1110nnnn	2	Pitch wheel change
CHANNEL MODE MESSAGES		
1011nnnn	2	Selects Channel Mode
SYSTEM MESSAGES		
11110000	******	System Exclusive
11110sss	0 to 2	System Common
11111ttt	0	System Real Time

NOTES:

nnnn: N−1, where N = Channel #,
i.e. 0000 is Channel 1.
0001 is Channel 2.
.
.
1111 is Channel 16.

*****: 0iiiiiii, data, . . ., EOX
iiiiiii: Identification
sss: 1 to 7
ttt: 0 to 7

TABLE II

CHANNEL VOICE MESSAGES

STATUS	DATA BYTES	DESCRIPTION
1000nnnn	0kkkkkkk 0vvvvvvv	Note Off (see notes 1-4) vvvvvvv: note off velocity
1001nnnn	0kkkkkkk 0vvvvvvv	Note On (see Notes 1-4) vvvvvvv≠0: velocity vvvvvvv=0: note off
1010nnnn	0kkkkkkk 0vvvvvvv	Polyphonic Key Pressure (After-Touch) vvvvvvv: pressure value
1011nnnn	0ccccccc 0vvvvvvv	Control Change ccccccc: control # (0-121) (see notes 5-8) vvvvvvv: control value ccccccc=122 thru 127: Reserved. See Table III.
1100nnnn	0ppppppp	Program Change ppppppp: program number (0-127)
1101nnnn	0vvvvvvv	Channel Pressure (After-Touch) vvvvvvv: pressure value
1110nnnn	0vvvvvvv 0vvvvvvv	Pitch Wheel Change LSB (see note 10) Pitch Wheel Change MSB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

Notes for Table II on the following page.

NOTES:

1. nnnn: Voice Channel # (1-16, coded as defined in Table I notes)
2. kkkkkkk: note # (0-127)
   kkkkkkk=60: Middle C of keyboard

0	12	24	36	48	60	72	84	96	108	120	127
		c	c	c	c	c	c	c	c		
		piano range									

3. vvvvvvv: key velocity
   A logarithmic scale would be advisable.

0	1		64		127
off	ppp	pp	p	mp	mf	f	f	fff

   vvvvvvv=64: in case of no velocity sensors
   vvvvvvv=0: Note Off, with velocity=64
4. Any Note On message sent should be balanced by sending a Note Off message for that note in that channel at some later time.
5. ccccccc: control number

ccccccc	Description
0	Continuous Controller 0 MSB
1	Continuous Controller 1 MSB (MODULATION WHEEL)
2	Continuous Controller 2 MSB
3	Continuous Controller 3 MSB
4-31	Continuous Controller 4-31 MSB
32	Continuous Controller 0 LSB
33	Continuous Controller 1 LSB (MODULATION WHEEL)
34	Continuous Controller 2 LSB
35	Continuous Controller 3 LSB
36-63	Continuous Controller 4-31 LSB
64-95	Switches (On/Off)
96-121	Undefined
122-127	Reserved for Channel Mode message (see Table III).

6. The controllers are not specifically defined. A manufacturer can assign the logical controllers to physical ones as necessary. The controller allocation table must be provided in the user's operation manual.
7. Continuous controllers are divided into Most Significant and Least Significant Bytes. If only seven bits of resolution are needed for any particular controllers, only the MSB is sent. It is not necessary to send the LSB. If more resolution is needed, then both are sent, first the MSB, then the LSB. If only the LSB has changed in value, the LSB may be sent without re-sending the MSB.
8. vvvvvvv: control value (MSB)
   (for controllers)

0	127
min	max

   (for switches)

0	127
off	on

   Numbers 1 through 126, inclusive, are ignored.
9. Any messages (e.g. Note On), which are sent successively under the same status, can be sent without a Status byte until a different Status byte is needed.
10. Sensitivity of the pitch bender is selected in the receiver. Center position value (no pitch change) is 2000H, which would be transmitted EnH-00H-40H.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TABLE III
CHANNEL MODE MESSAGES

STATUS	DATA BYTES	DESCRIPTION
1011nnnn	0ccccccc 0vvvvvvv	Mode Messages
		ccccccc=122: Local Control vvvvvvv=0, Local Control Off vvvvvvv=127, Local Control On
		ccccccc=123: All Notes Off vvvvvvv=0
		ccccccc=124: Omni Mode Off (All Notes Off) vvvvvvv=0
		ccccccc=125: Omni Mode On (All Notes Off) vvvvvvv=0
		ccccccc=126: Mono Mode On (Poly Mode Off) (All Notes Off) vvvvvvv=M, where M is the number of channels vvvvvvv=0, the number of channels equals the number of voices in the receiver.
		ccccccc=127: Poly Mode On (Mono Mode Off) vvvvvvv=0 (All Notes Off)

NOTES:

1. nnnn: Basic Channel # (1-16, coded as defined in Table I)
2. Messages 123 thru 127 function as All Notes Off messages. They will turn off all voices controlled by the assigned Basic Channel. Except for message 123, All Notes Off, they should not be sent periodically, but only for a specific purpose. In no case should they be used in lieu of Note Off commands to turn off notes which have been previously turned on. Therefore any All Notes Off command (123-127) may be ignored by receiver with no possibility of notes staying on, since any Note On command must have a corresponding specific Note Off command.
3. Control Change #122, Local Control, is optionally used to interrupt the internal control path between the keyboard, for example, and the sound generating circuitry. If 0 (Local Off message) is received, the path is disconnected; the keyboard data goes only to MIDI and the sound-generating circuitry is controlled only by incoming MIDI data. If a 7F H (Local On message) is received, normal operation is restored.
4. The third byte of 'Mono' specifies the number of channels in which Monophonic Voice messages are to be sent. This number, "M," is a number between 1 and 16. The channel(s) being used, then, will be the current Basic Channel (=N) thru N+M-1 up to a maximum of 16. If M=0, this is a special case directing the receiver to assign all its voices, one per channel, from the Basic Channel N through 16.

TABLE IV
SYSTEM COMMON MESSAGES

STATUS	DATA BYTES	DESCRIPTION
11110001		Undefined
11110010		Song Position Pointer
	0lllllll	lllllll: (Least significant) hhhhhhh: (Most significant)
11110011	0sssssss	Song Select sssssss: Song #
11110100		Undefined
11110101		Undefined
11110110	none	Tune Request
11110111	none	EOX: "End of System Exclusive" flag

NOTES:

1. Song Position Pointer: Is an internal register which holds the number of MIDI beats (1 beat = 6 MIDI clocks) since the start of the song. Normally it is set to 0 when the START switch is pressed, which starts sequence playback. It then increments with every sixth MIDI clock receipt, until STOP is pressed. If CONTINUE is pressed, it continues to increment. It can be arbitrarily preset (to a resolution of 1 beat) by the SONG POSITION POINTER message.
2. Song Select: Specifies which song or sequence is to be played upon receipt of a Start (Real-Time) message.
3. Tune Request: Used with analog synthesizers to request them to tune their oscillators.
4. EOX: Used as a flag to indicate the end of a System Exclusive transmission (see Table VI).

สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย เพื่อการศึกษาของคนไทย ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TABLE V

SYSTEM REAL TIME MESSAGES

STATUS	DATA BYTES	DESCRIPTION
11111000		Timing Clock
11111001		Undefined
11111010		Start
11111011		Continue
11111100		Stop
11111101		Undefined
11111110		Active Sensing
11111111		System Reset

NOTES:

1. The System Real Time messages are for synchronizing all of the system in real time.
2. The System Real Time messages can be sent at any time. Any messages which consist of two or more bytes may be split to insert Real Time messages.
3. Timing Clock (F8H)
   The system is synchronized with this clock, which is sent at a rate of 24 clocks/quarter note.
4. Start (from beginning of song) (FAH)
   This byte is immediately sent when the PLAY switch on the master (e.g. sequencer or rhythm unit) is pressed.
5. Continue (FBH)
   This is sent when the CONTINUE switch is hit. A sequence will continue at the time of the next clock.
6. Stop (FCH)
   This byte is immediately sent when the STOP switch is hit. It will stop the sequence.
7. Active Sensing (FEH)
   Use of this message is optional, for either receivers or transmitters. This is a "dummy" Status byte that is sent every 300 ms (max), whenever there is no other activity on MIDI. The receiver will operate normally if it never receives FEH. Otherwise, if FEH is ever received, it will expect to receive FEH or a transmission of any type every 300 ms (max). If a period of 300 ms passes with no activity, the receiver will turn off the voices and return to normal operation.
8. System Reset (FFH)
   This message initializes all of the system to the condition of just having turned on power. The System Reset message should be used sparingly, preferably under manual command only. In particular, it should not be sent automatically on power up.

TABLE VI

SYSTEM EXCLUSIVE MESSAGES

STATUS	DATA BYTES	DESCRIPTION
11110000	0iiiiiii	Bulk dump etc iiiiiii: identification
	(0*******)	Any number of bytes may be sent here, for any purpose, as long as they all have a zero in the most isgnificant bit
	11110111	EOX: "End of System Exclusive"

NOTES:

1. iiiiiii: identification ID (0-127)
2. All bytes between the System Exclusive Status byte and EOX of the next Status byte must have zeroes in the MSB.
3. The ID number can be obtained from the MIDI committee. See Table VII.
4. In no case should other Status or Data bytes (except Real-Time) be interleaved with System Exclusive, regardless of whether or not the ID code is recognized.
5. EOX or any other Status byte, except Real-Time, will terminate a System Exclusive message, and should be sent immediately at its conclusion.

TABLE VII

MANUFACTURERS' ID NUMBERS

Sequential Circuits, Inc.	01H
Big Briar	02H
Octave/Plateau	03H
Moog Music	04H
Passport Designs	05H
Lexicon	06H
Oberheim	10H
Bon Tempi	20H
S.I.E.L.	21H
Kawai	40H
Roland	41H
Korg	42H
Yamaha	43H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

แผนภาพมิติอิมพลีเมนเตชั่นของคาสิโอ CT-640

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

del CT-640 MIDI Implementation Chart

Version : 1

Function...		Transmitted	Recognized	Remarks
asic hannel	Default Changed	1 ×	1–4 ×	(*1)
ode	Default Messages Altered	Mode 3 ×	Mode 3 × ×	
ote umber:	True voice	36–96	0–127 36–96	0–11, 12–23, 24–35 = 36–47, 97–108, 109–120, 121–127 = 85–96 (*2)
locity	Note ON Note OFF	× 9n v = 64 × 9n v = 0	× 9n v = 1–127→64 × 9n v = 0,8n v = ××	(*2) ××...No function
ter uch	Key's Ch's	× ×	× ×	
ch Bender		×	×	
64 ntrol ange		O	O	Sustain pedal (*2)
g nge:	True*	O 0–29	O 0–29	(*3)
tem Exclusive		×	×	
tem mon	: Song Pos : Song Sel : Tune	× × ×	× × ×	
em Time	: Clock : Command	O O	× ×	Continue not sent
s	: Local ON/OFF : All Notes OFF : Active Sense : Reset	× × × ×	× × × ×	
s		MIDI messages transmitted/received only when set to the MIDI mode. *1) Multi messages received on CH 1–4 *2) Not received on CH-4 *3) Transmission/reception of 0–19 on CH-4		

1: OMNI ON, POLY
3: OMNI OFF, POLY

Mode 2: OMNI ON, MONO
Mode 4: OMNI OFF, MONO

O: Yes
× : No

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค

คู่มือการใช้งาน 8253 PIT , Z-80 SIO

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

intel

8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
- 3 Independent 16-Bit Counters
- DC to 2.6 MHz
- Programmable Counter Modes
- Count Binary or BCD
- Single +5V Supply
- Available in EXPRESS
  - —Standard Temperature Range
  - —Extended Temperature Range

The Intel® 8253 is a programmable counter/timer device designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2.6 MHz. All modes of operation are software programmable.

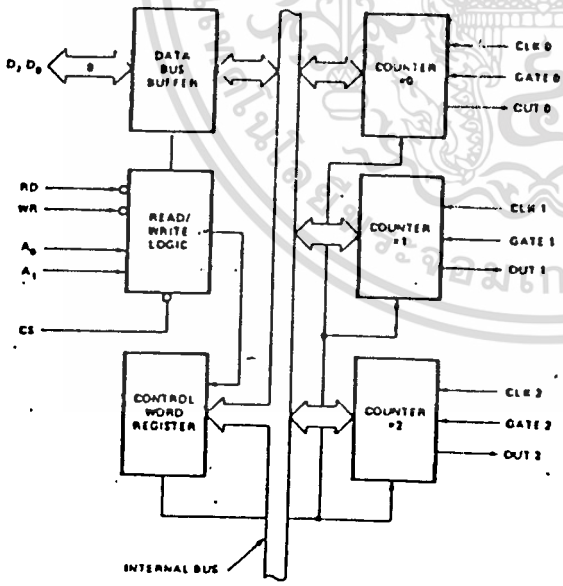


Figure 1. Block Diagram

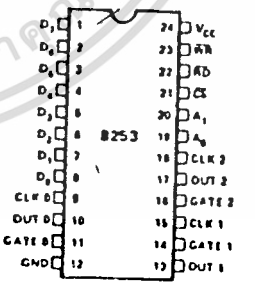


Figure 2. Pin Configuration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FUNCTIONAL DESCRIPTION

General

The 8253 is a programmable interval timer/counter specifically designed for use with the Intel™ Microcomputer systems. Its function is that of a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks. It is easy to see that the software overhead is minimal and that multiple delays can easily be maintained by assignment of priority levels.

Other counter/timer functions that are non-delay in nature but also common to most microcomputers can be implemented with the 8253.

- Programmable Rate Generator
- Event Counter
- Binary Rate Multiplier
- Real Time Clock
- Digital One-Shot
- Complex Motor Controller

Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 8253 to the system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput CPU instructions. The Data Bus Buffer has three basic functions.

1. Programming the MODES of the 8253.
2. Loading the count registers.
3. Reading the count values.

Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and in turn generates control signals for overall device operation. It is enabled or disabled by CS so that no operation can occur to change the function unless the device has been selected by the system logic.

$\overline{RD}$ (Read)

A "low" on this input informs the 8253 that the CPU is inputting data in the form of a counters value.

$\overline{WR}$ (Write)

A "low" on this input informs the 8253 that the CPU is outputting data in the form of mode information or loading counters.

A0, A1

These inputs are normally connected to the address bus. Their function is to select one of the three counters to be operated on and to address the control word register for mode selection.

$\overline{CS}$ (Chip Select)

A "low" on this input enables the 8253. No reading or writing will occur unless the device is selected. The $\overline{CS}$ input has no effect upon the actual operation of the counters.

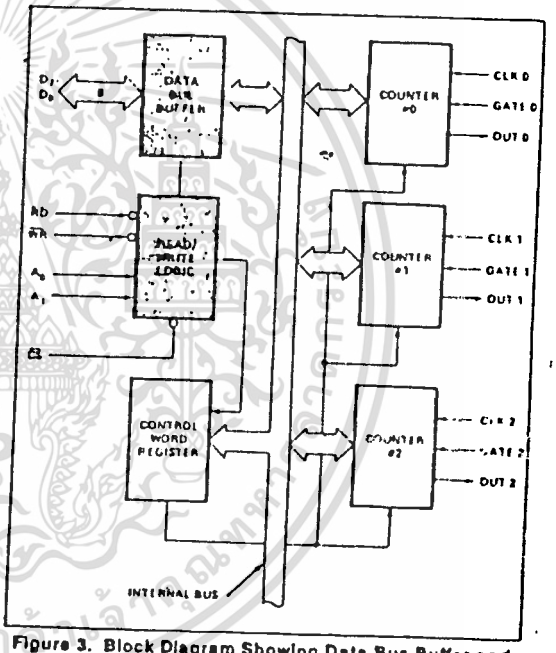


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3 State

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Control Word Register

The Control Word Register is selected when A0, A1 are 11. It then accepts information from the data bus buffer and stores it in a register. The information stored in this register controls the operational MODE of each counter, selection of binary or BCD counting and the loading of each count register.

The Control Word Register can only be written into; no read operation of its contents is available.

Counter #0, Counter #1, Counter #2

These three functional blocks are identical in operation so only a single Counter will be described. Each Counter consists of a single, 16-bit, pre-settable, DOWN counter. The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of MODES stored in the Control Word Register.

The counters are fully independent and each can have separate Mode configuration and counting operation, binary or BCD. Also, there are special features in the control word that handle the loading of the count value so that software overhead can be minimized for these functions.

The reading of the contents of each counter is available to the programmer with simple READ operations for event counting applications and special commands and logic are included in the 8253 so that the contents of each counter can be read "on the fly" without having to inhibit the clock input.

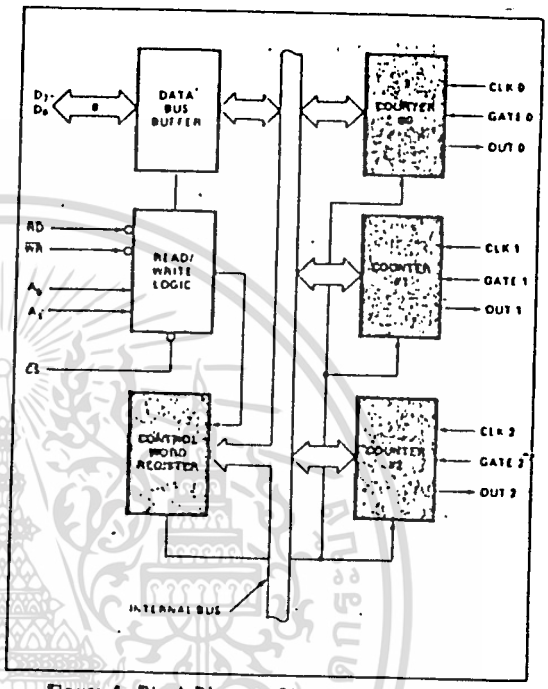


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

8253 SYSTEM INTERFACE

The 8253 is a component of the Intel™ Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the systems software as an array of peripheral I/O ports, three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A0, A1 connect to the A0, A1 address bus signals of the CPU. The $\overline{CS}$ can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel® 8205 for larger systems.

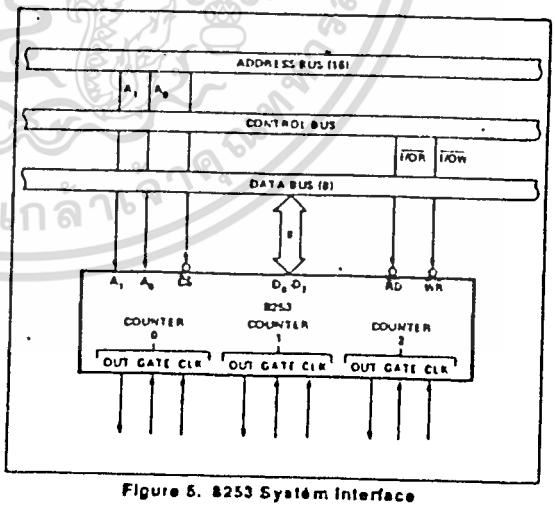


Figure 5. 8253 System Interface

231306-001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

OPERATIONAL DESCRIPTION

General

The complete functional definition of the 8253 is programmed by the systems software. A set of control words must be sent out by the CPU to initialize each counter of the 8253 with the desired MODE and quantity information. Prior to initialization, the MODE, count, and output of all counters is undefined. These control words program the MODE, Loading sequence and selection of binary or BCD counting.

Once programmed, the 8253 is ready to perform whatever timing tasks it is assigned to accomplish.

The actual counting operation of each counter is completely independent and additional logic is provided on-chip so that the usual problems associated with efficient monitoring and management of external, asynchronous events or rates to the microcomputer system have been eliminated.

Programming the 8253

All of the MODES for each counter are programmed by the systems software by simple I/O operations.

Each counter of the 8253 is individually programmed by writing a control word into the Control Word Register. (A0, A1 = 11)

Control Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definition of Control

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

RL — Read/Load:

RL1	RL0	
0	0	Counter Latching operation (see READ/WRITE Procedure Section)
1	0	Read/Load most significant byte only.
0	1	Read/Load least significant byte only.
1	1	Read/Load least significant byte first, then most significant byte.

M — MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD:

0	Binary Counter 16 bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

Counter Loading

The count register is not loaded until the count value is written (one or two bytes, depending on the mode selected by the RL bits), followed by a rising edge and a falling edge of the clock. Any read of the counter prior to that falling clock edge may yield invalid data.

MODE Definition

MODE 0: Interrupt on Terminal Count. The output will be initially low after the mode set operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When terminal count is reached the output will go high and remain high until the selected count register is reloaded with the mode or a new count is loaded. The counter continues to decrement after terminal count has been reached.

Rewriting a counter register during counting results in the following:

(1) Write 1st byte stops the current counting
(2) Write 2nd byte starts the new count.

MODE 1: Programmable One-Shot. The output will go low on the count following the rising edge of the gate input.

The output will go high on the terminal count. If a new count value is loaded while the output is low it will not affect the duration of the one-shot pulse until the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse.

The one-shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MODE 2: Rate Generator. Divide by N counter. The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value.

The gate input, when low, will force the output high. When the gate input goes high, the counter will start from the initial count. Thus, the gate input can be used to synchronize the counter.

When this mode is set, the output will remain high until after the count register is loaded. The output then can also be synchronized by software.

MODE 3: Square Wave Rate Generator. Similar to MODE 2 except that the output will remain high until one half the count has been completed (for even numbers) and go low for the other half of the count. This is accomplished by decrementing the counter by two on the falling edge of each clock pulse. When the counter reaches terminal count, the state of the output is changed and the counter is reloaded with the full count and the whole process is repeated.

If the count is odd and the output is high, the first clock pulse (after the count is loaded) decrements the count by 1. Subsequent clock pulses decrement the clock by 2. After timeout, the output goes low and the full count is reloaded. The first clock pulse (following the reload) decrements the counter by 3. Subsequent clock pulses decrement the count by 2 until timeout. Then the whole process is repeated. In this way, if the count is odd, the output will be high for $(N+1)/2$ counts and low for $(N-1)/2$ counts.

In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following $\overline{WR}$ of a new count value.

MODE 4: Software Triggered Strobe. After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, then will go high again.

If the count register is reloaded during counting, the new count will be loaded on the next CLK pulse. The count will be inhibited while the GATE input is low.

MODE 5: Hardware Triggered Strobe. The counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of any trigger.

Modes \ Signal Status	Low Or Going Low	Rising	High
0	Disables counting	—	Enables counting
1	—	1) Initiates counting 2) Resets output after next clock	—
2	1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
4	Disables counting	—	Enables counting
5	—	Initiates counting	—

Figure 8. Gate Pin Operations Summary

231306-001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

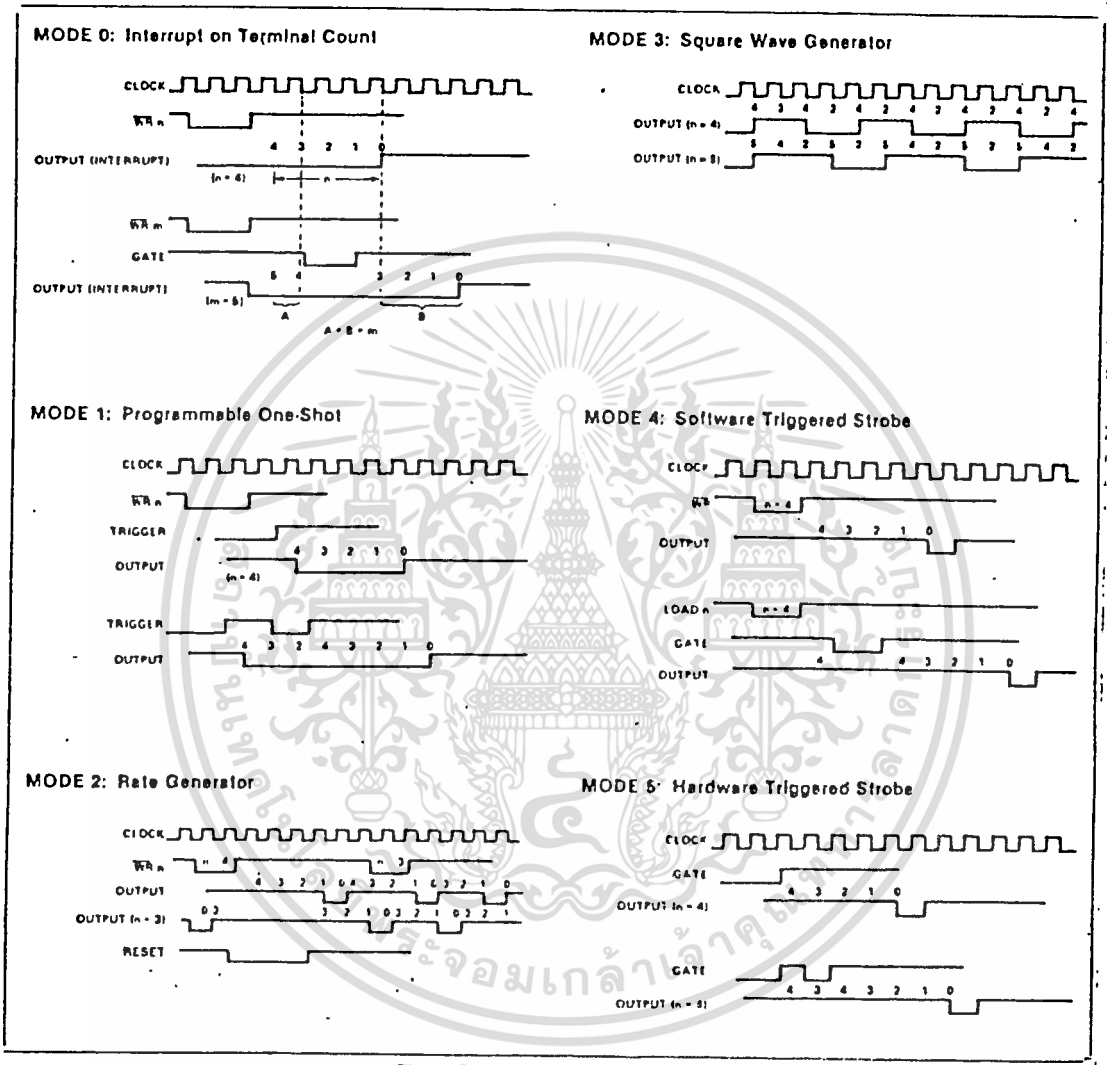


Figure 7. 8253 Timing Diagrams

231306-001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8253 READ/WRITE PROCEDURE

Write Operations

The systems software must program each counter of the 8253 with the mode and quantity desired. The programmer must write out to the 8253 a MODE control word and the programmed number of count register bytes (1 or 2) prior to actually using the selected counter.

The actual order of the programming is quite flexible. Writing out of the MODE control word can be in any sequence of counter selection, e.g., counter #0 does not have to be first or counter #2 last. Each counter's MODE control word register has a separate address so that its loading is completely sequence independent. (SC0, SC1)

The loading of the Count Register with the actual count value, however, must be done in exactly the sequence programmed in the MODE control word (RL0, RL1). This loading of the counter's count register is still sequence independent like the MODE control word loading, but when a selected count register is to be loaded it must be loaded with the number of bytes programmed in the MODE control word (RL0, RL1). The one or two bytes to be loaded in the count register do not have to follow the associated MODE control word. They can be programmed at any time following the MODE control word loading as long as the correct number of bytes is loaded in order.

All counters are down counters. Thus, the value loaded into the count register will actually be decremented. Loading all zeroes into a count register will result in the maximum count (2^{16} for Binary or 10^4 for BCD). In MODE 0 the new count will not restart until the load has been completed. It will accept one of two bytes depending on how the MODE control words (RL0, RL1) are programmed. Then proceed with the restart operation.

	MODE Control Word Counter n
LSB	Count Register byte Counter n
MSB	Count Register byte Counter n

Note: Format shown is a simple example of loading the 8253 and does not imply that it is the only format that can be used.

Figure 8. Programming Format

			A1	A0
No. 1		MODE Control Word Counter 0	1	1
No. 2		MODE Control Word Counter 1	1	1
No. 3		MODE Control Word Counter 2	1	1
No. 4	LSB	Count Register Byte Counter 1	0	1
No. 5	MSB	Count Register Byte Counter 1	0	1
No. 6	LSB	Count Register Byte Counter 2	1	0
No. 7	MSB	Count Register Byte Counter 2	1	0
No. 8	LSB	Count Register Byte Counter 0	0	0
No. 9	MSB	Count Register Byte Counter 0	0	0

Note: The exclusive addresses of each counter's count register make the task of programming the 8253 a very simple matter, and maximum effective use of the device will result if this feature is fully utilized.

Figure 9. Alternate Programming Formats

231306-001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Read Operations

In most counter applications it becomes necessary to read the value of the count in progress and make a computational decision based on this quantity. Event counters are probably the most common application that uses this function. The 8253 contains logic that will allow the programmer to easily read the contents of any of the three counters without disturbing the actual count in progress.

There are two methods that the programmer can use to read the value of the counters. The first method involves the use of simple I/O read operations of the selected counter. By controlling the A0, A1 inputs to the 8253 the programmer can select the counter to be read (remember that no read operation of the mode register is allowed A0, A1=11). The only requirement with this method is that in order to assure a stable count reading the actual operation of the selected counter must be inhibited either by controlling the Gate input or by external logic that inhibits the clock input. The contents of the counter selected will be available as follows.

first I/O Read contains the least significant byte (LSB).

second I/O Read contains the most significant byte (MSB)

Due to the internal logic of the 8253 it is absolutely necessary to complete the entire reading procedure. If two bytes are programmed to be read then two bytes must be read before any loading WR command can be sent to the same counter.

Read Operation Chart

A1	A0	RD	
0	0	0	Read Counter No. 0
0	1	0	Read Counter No. 1
1	0	0	Read Counter No. 2
1	1	0	Illegal

Reading While Counting

In order for the programmer to read the contents of any counter without effecting or disturbing the counting operation the 8253 has special internal logic that can be accessed using simple WR commands to the MODE register. Basically, when the programmer wishes to read the contents of a selected counter "on the fly" he loads the MODE register with a special code which latches the present count value into a storage register so that its contents contain an accurate, stable quantity. The programmer then issues a normal read command to the selected counter and the contents of the latched register is available.

MODE Register for Latching Count

A0, A1 = 11

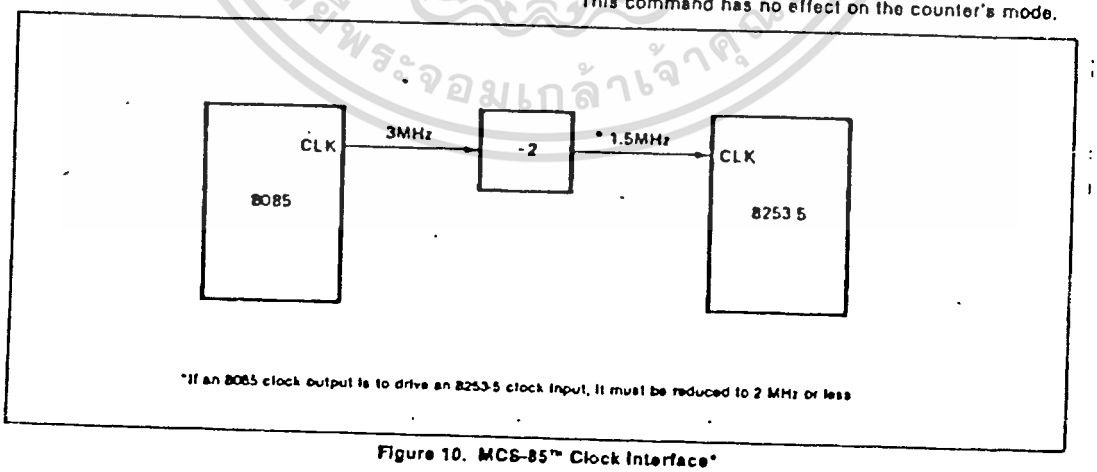
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1,SC0 — specify counter to be latched.

D5,D4 — 00 designates counter latching operation.

X — don't care.

The same limitation applies to this mode of reading the counter as the previous method. That is, it is mandatory to complete the entire read operation as programmed. This command has no effect on the counter's mode.



*If an 8085 clock output is to drive an 8253-5 clock input, it must be reduced to 2 MHz or less

Figure 10. MCS-85™ Clock Interface*

231306-001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground ..... −0.5 V to +7 V
Power Dissipation ..... 1 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5V ± 10%) *

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}+.5V	V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH}	Output High Voltage	2.4		V	Note 2
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to .45V
I_{CC}	V_{CC} Supply Current		140	mA	

CAPACITANCE (T_A = 25°C, V_{CC} = GND = 0V)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	f_c = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5.0V ± 10%, GND = 0V) *

Bus Parameters (Note 3)

READ CYCLE

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{AR}	Address Stable Before $\overline{READ}$	50		30		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		5		ns
t_{RR}	$\overline{READ}$ Pulse Width	400		300		ns
t_{RD}	Data Delay From $\overline{READ}$[4]		300		200	ns
t_{DF}	$\overline{READ}$ to Data Floating	25	125	25	100	ns
t_{RV}	Recovery Time Between READ and Any Other Control Signal	1		1		μs

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

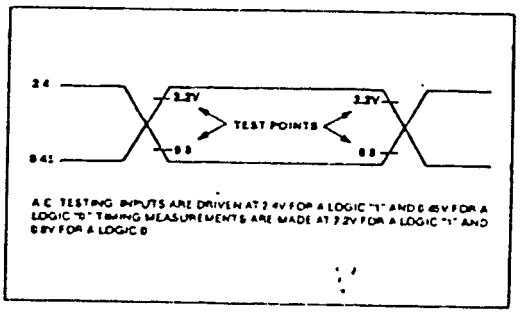
Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{AW}	Address Stable Before $\overline{WRITE}$	50		30		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	30		30		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		300		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		250		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		30		ns
t_{RV}	Recovery Time Between $\overline{WRITE}$ and Any Other Control Signal	1		1		μs

CLOCK AND GATE TIMING

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{CLK}	Clock Period	380	dc	380	dc	ns
t_{PWH}	High Pulse Width	230		230		ns
t_{PWL}	Low Pulse Width	150		150		ns
t_{GW}	Gate Width High	150		150		ns
t_{GL}	Gate Width Low	100		100		ns
t_{GS}	Gate Set Up Time to CLK↑	100		100		ns
t_{GH}	Gate Hold Time After CLK↑	50		50		ns
t_{OD}	Output Delay From CLK↓ [4]		400		400	ns
t_{ODG}	Output Delay From Gate↓ [4]		300		300	ns

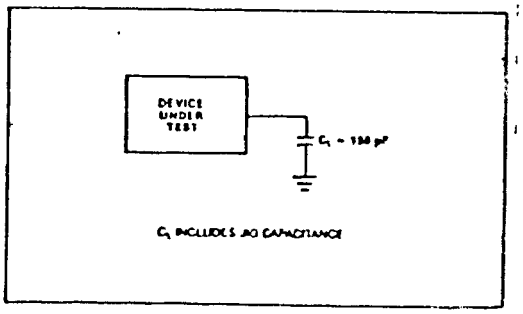
NOTES:
1. I_{OL} = 2.2 mA.
2. I_{OH} = −400 μA.
3. AC timings measured at V_{OH} 2.2, V_{OL} = 0.8.
4. C_L = 150pF.
* For Extended Temperature EXPRESS, use M8253 electrical parameters.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". TIMING MEASUREMENTS ARE MADE AT 2.2V FOR A LOGIC "1" AND 0.8V FOR A LOGIC 0.

A.C. TESTING LOAD CIRCUIT

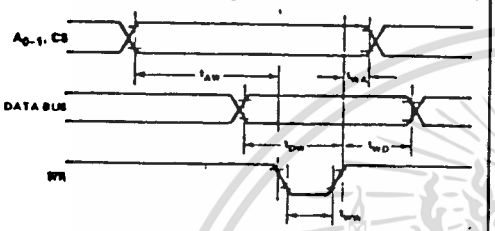


231306-001

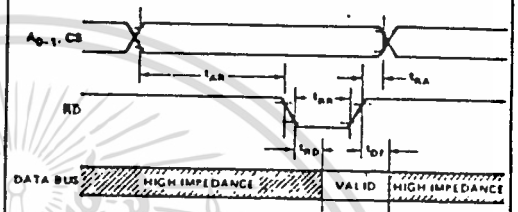
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

WAVEFORMS

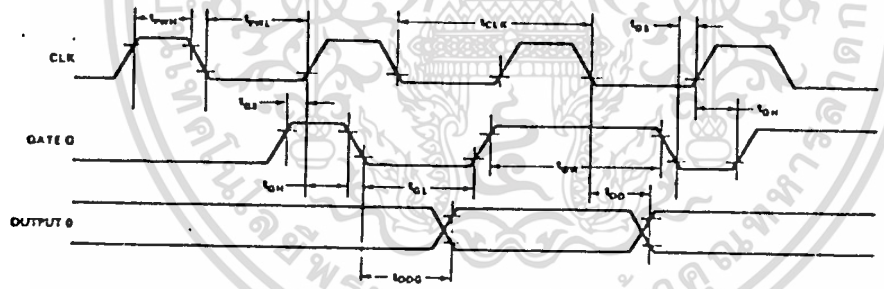
WRITE TIMING



READ TIMING



CLOCK AND GATE TIMING



231306-001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Z8440/1/2/4 Z80® SIO Serial Input/Output Controller

Zilog

Product Specification

April 1985

FEATURES

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7, or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7, or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection, and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

GENERAL DESCRIPTION

The Z80 SIO Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs, and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast, or slow, peripheral devices. While designed primarily as a member of the Z80 family, its versatility makes it well suited to many other CPUs.

The Z80 SIO is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5V power supply and the standard Z80 family single-phase clock. The Z8444 is packaged in a 44-pin ceramic LCC.

PIN DESCRIPTION

Figures 1 through 6 illustrate the three 40-pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together:

- Z80 SIO/2 lacks $\overline{SYNCB}$
- Z80 SIO/1 lacks $\overline{DTRB}$
- Z80 SIO/0 has all four signals but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The 44-pin package, the Z80 SIO/4, has all options (Figure 7).

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{A}$. *Channel A or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. *Control or Data Select* (input, High selects Control). This input defines the type of information transfer performed

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

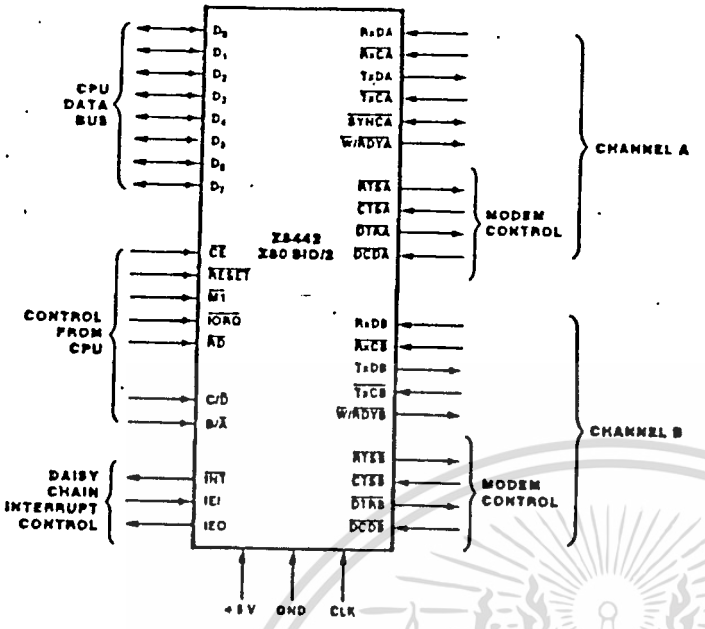


Figure 1. Pin Functions

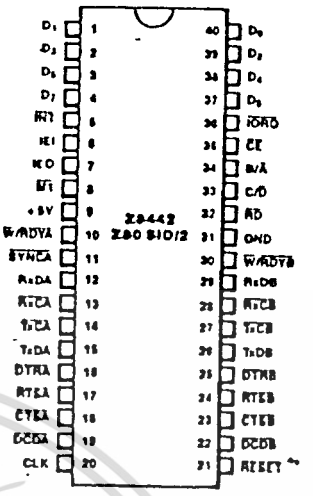


Figure 2. 40-pin Dual-In-Line Package (DIP), Pin Assignments

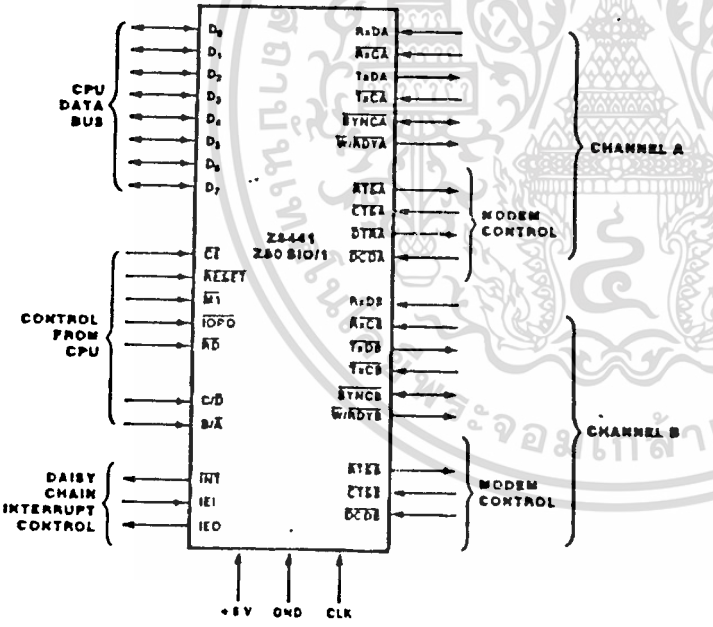


Figure 3. Pin Functions

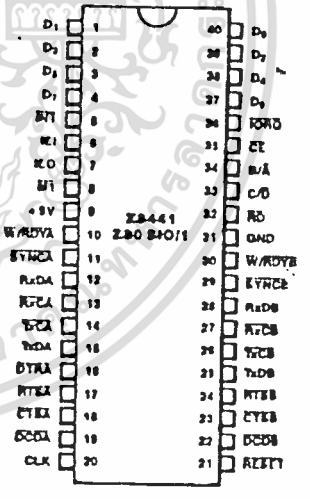


Figure 4. 40-pin Dual-In-Line Package (DIP), Pin Assignments

2042-001, 002, 003, 004

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

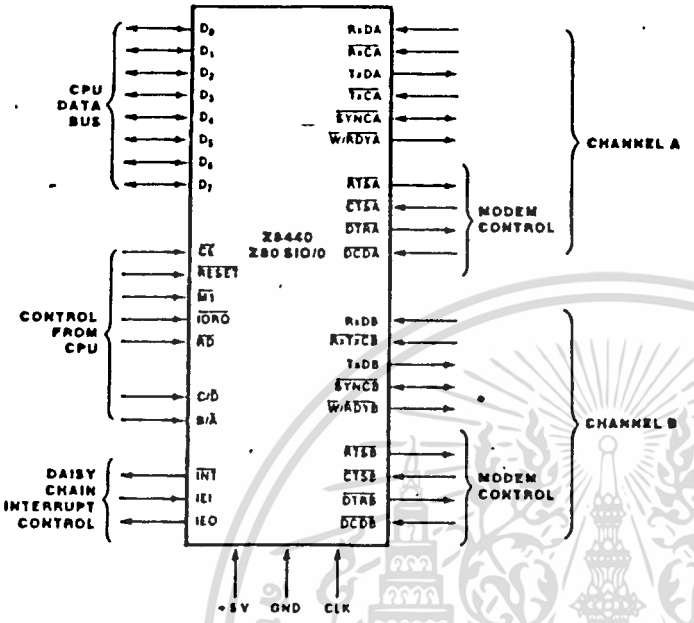


Figure 5. Pin Functions

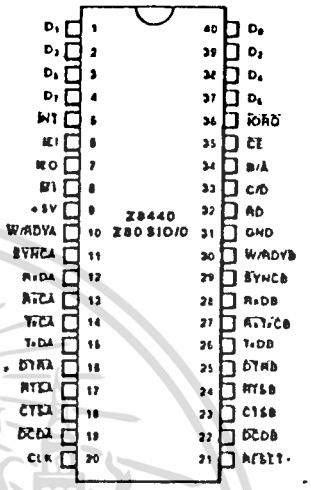


Figure 6. 40-pin Dual-In-Line Package (DIP), Pin Assignments

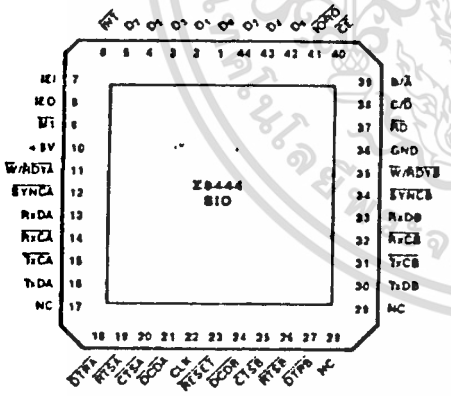


Figure 7. 44-pin Chip Carrier, Pin Assignments

between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/$\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (Input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The SIO uses the standard Z80 System Clock to synchronize internal signals. This is a single phase clock.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z80 SIO. D_0 is the least significant bit.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables, otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffering does not guarantee a specific noise-level margin.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into the Z80 SIO. They can also be programmed as general-purpose outputs.

In the Z80 SIO/1 bonding option, $\overline{DTRB}$ is omitted.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{RD}$ to transfer commands and data between the CPU and the SIO. When $\overline{CE}$, $\overline{RD}$, and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$. As mentioned previously, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{M1}$. *Machine Cycle One*(input from Z80 CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z80 CPU.

$\overline{RxCA}$, $\overline{RxCB}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32, or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered; no noise level margin is specified.

In the Z80 SIO/0 bonding option, $\overline{RxCB}$ is bonded together with $\overline{TxCB}$.

$\overline{RD}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with B/$\overline{A}$, $\overline{CE}$, and $\overline{IORQ}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{RESET}$. *Reset* (input, active Low). A Low $\overline{RESET}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High, and disables all interrupts. The control registers must be rewritten after the SIO is reset and before data is transmitted or received.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{RTS}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{SYNCA}$, $\overline{SYNCB}$. *Synchronization* (bidirectional, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{SYNC}$ must be driven Low on the second rising edge of $\overline{RxC}$ after that rising edge of $\overline{RxC}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{SYNC}$ input. Once $\overline{SYNC}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{RxC}$ that immediately precedes the falling edge of $\overline{SYNC}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{RxC}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z80 SIO/2 bonding option, $\overline{SYNCB}$ is omitted.

$\overline{TxCA}$, $\overline{TxCB}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32, or 64 times the data rate; however, the clock multiplier must be the same for the transmitter and the receiver. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements; no noise level margin is specified. Transmitter Clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z80 SIO/0 bonding option, $\overline{TxCB}$ is bonded together with $\overline{RxCB}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{TxC}$.

$\overline{W}$/$\overline{RDYA}$, $\overline{W}$/$\overline{RDYB}$. *Wait/Ready* (outputs, open drain when programmed for Wait function; driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FUNCTIONAL DESCRIPTION

The functional capabilities of the Z80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z80 family peripheral, it interacts with the Z80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling, and simple handshake capability. Figure 8 is a block diagram.

Figure 9 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

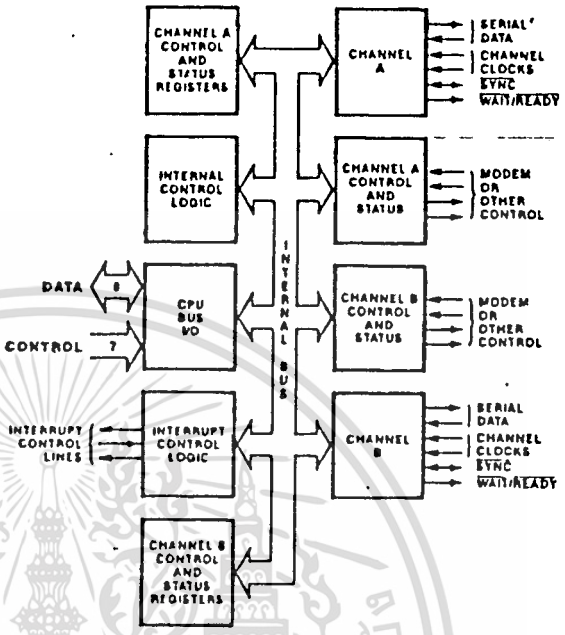


Figure 8. Block Diagram

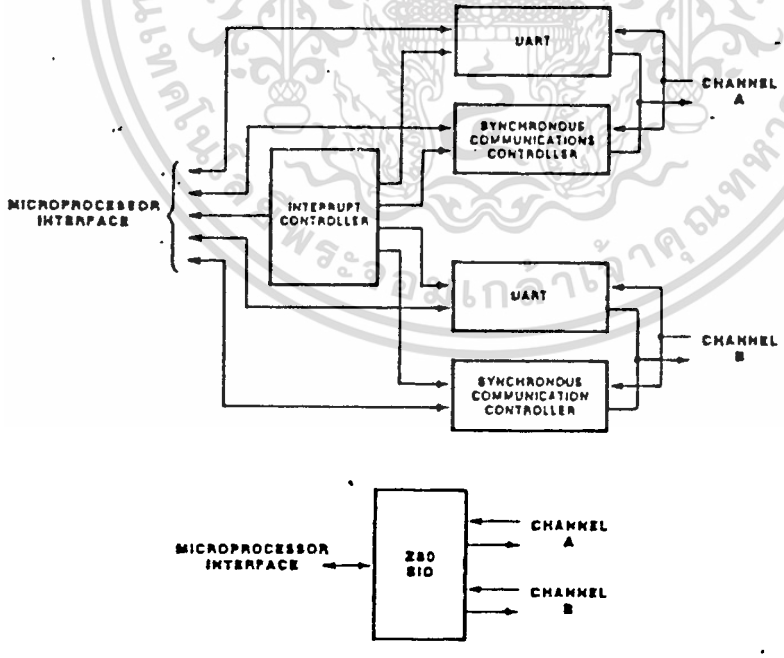


Figure 9. Conventional Devices Replaced by the Z80 SIO

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DATA COMMUNICATION CAPABILITIES

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous, or synchronous data-communication protocol. Figure 10a illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z80 SIO Technical Manual* (03-3033-01).

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist, as in the case of a transient, the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals, a feature that allows it to be used with a Z80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync characters can be removed without interrupting the CPU.

Five-, six-, or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple incoming sync characters, as shown in Figure 10b.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Figure 10a. Some Z80 SIO Protocols

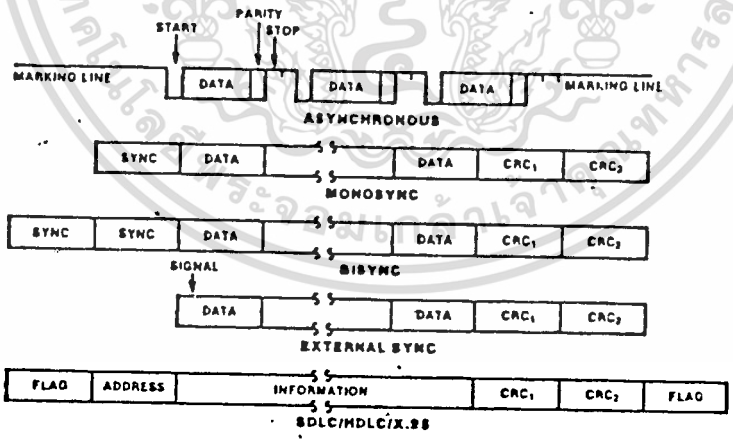


Figure 10b. Six-Bit Sync Character Recognition

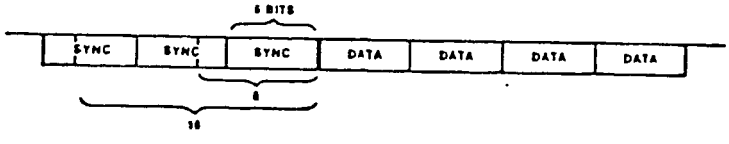


Figure 10. Data Communication

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0s; in SDLC modes, it is initialized to 1s. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disks, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the $\overline{SYNC}$ pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

I/O INTERFACE CAPABILITIES

The SIO offers the choice of polling, vectored or non-vectored interrupts and block-transfer modes to transfer data, status, and control information to, and from, the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts, and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit, and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{CTS}$), Data Carrier Detect ($\overline{DCD}$), and Synchronization ($\overline{SYNC}$) pins (Figures 1 through 7). In addition, an external/status

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

interrupt is also caused by a CRC-sending condition, or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

In a Z80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic:" the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a $\overline{WAIT}$ line in the CPU block-transfer mode or as a $\overline{READY}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{READY}$ output indicates that the SIO is ready to transfer data to, or from, memory. To the CPU, the $\overline{WAIT}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

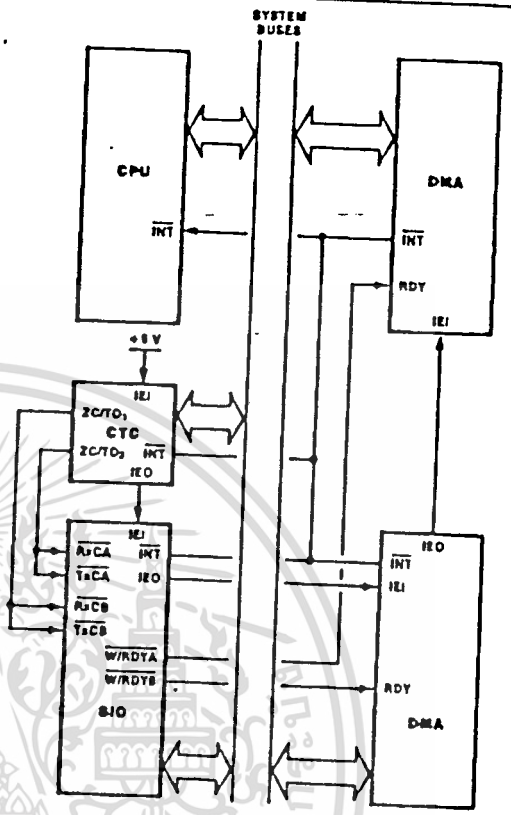


Figure 11. Typical Z80 Environment

INTERNAL STRUCTURE

The internal structure of the device includes a Z80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are

Table 1. Register Functions

	Read Register Functions
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

	Write Register Functions
WR0	Register pointers, CRC initialize, and initialization commands for the various modes.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

2042-011

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

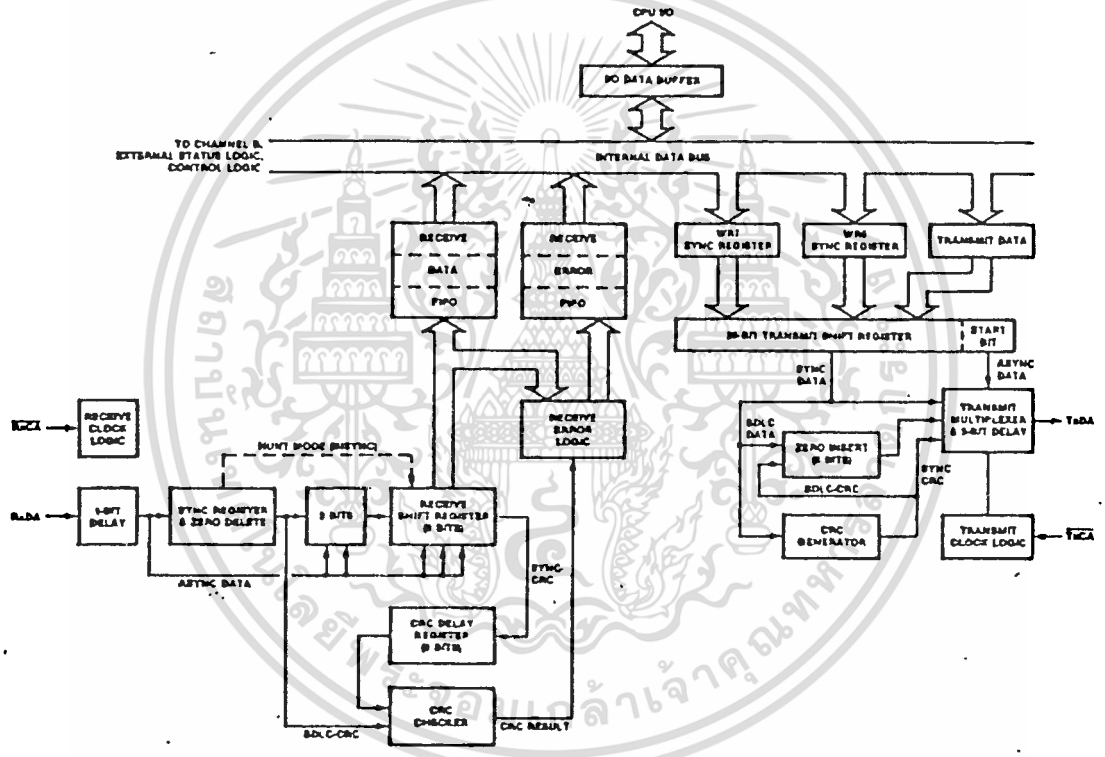


Figure 12. Transmit and Receive Data Path (Channel A)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PROGRAMMING

The system program first issues a series of commands that initialize the basic mode of operation and then issues other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input (B/$\overline{A}$) and the control/data (C/$\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector, and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register, the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

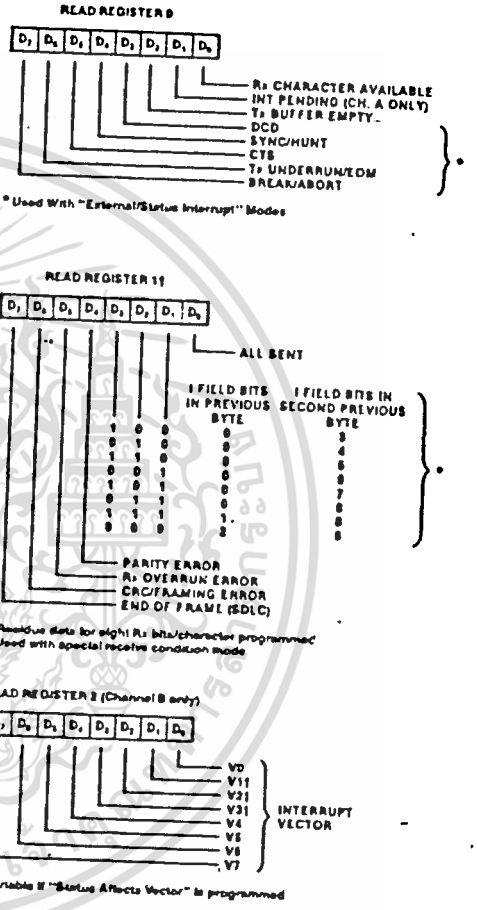


Figure 13. Read Register Bit Functions

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

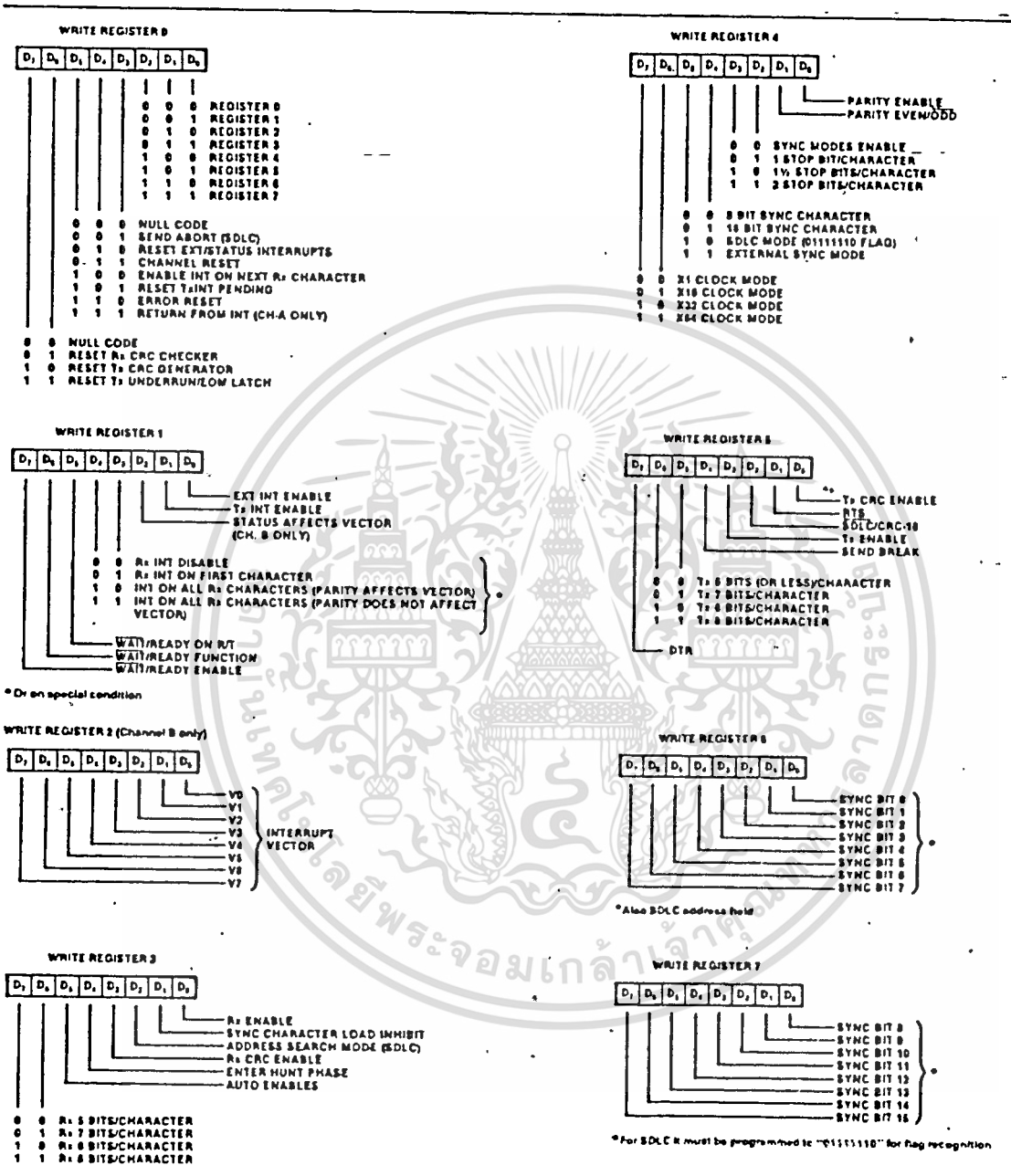


Figure 14. Write Register Bit Functions

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TIMING

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO ($\overline{INT}$ pulled Low), the Z80 CPU sends an interrupt-acknowledge sequence, $\overline{M1}$ Low and $\overline{IORQ}$ Low, a few cycles later (Figure 17).

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z80 CPU issues a Return From Interrupt (RETI) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever ED is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is 4D, the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z80 CPU Product Specification* in this document.

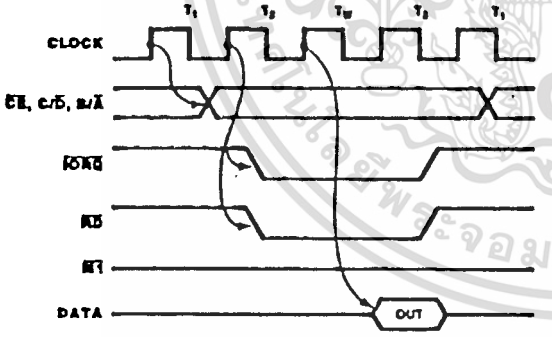


Figure 15. Read Cycle

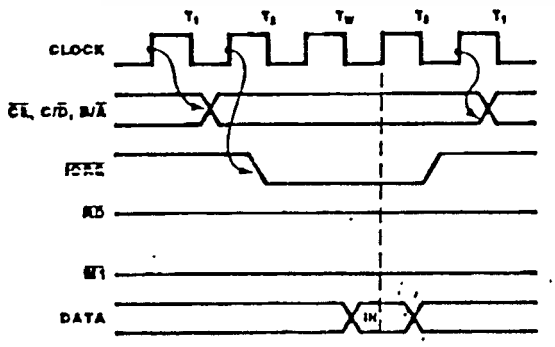


Figure 16. Write Cycle

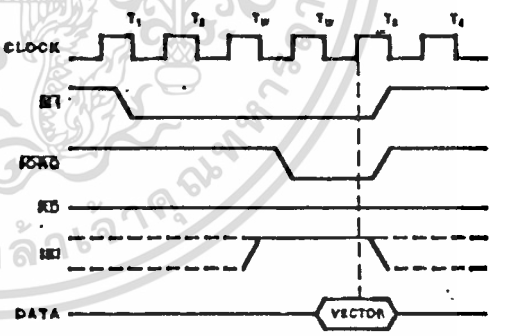


Figure 17. Interrupt Acknowledge Cycle

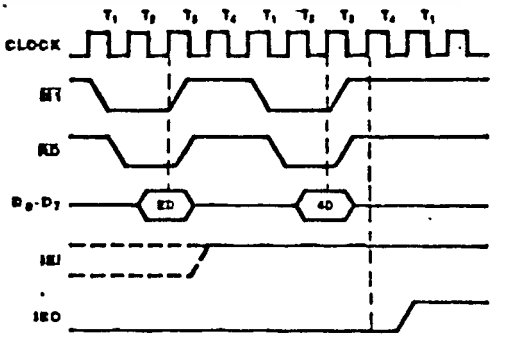


Figure 18. Return from Interrupt Cycle

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND ..... −0.3V to +7V
Operating Ambient Temperature ..... See Ordering Information
Storage Temperature ..... −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

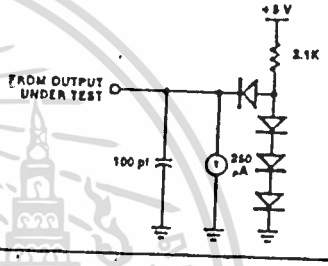
The Ordering Information section lists package temperature ranges and product numbers. Package drawings are in the Package Information section. Refer to the Literature List for additional documentation.

STANDARD TEST CONDITIONS

The DC characteristics and capacitance sections listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Available operating temperature ranges are:

- S = 0°C to +70°C, $+4.75V \leq V_{CC} \leq +5.25V$
- E = −40°C to +85°C, $+4.75V \leq V_{CC} \leq +5.25V$
- M = −55°C to +125°C, $+4.5V \leq V_{CC} \leq +5.5V$



DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} −0.6	V_{CC} +0.3	V	
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = −250 µA
I_{LI}	Input Leakage Current		±10	µA	V_{IN} = 0 to V_{CC}
I_{OL}	3-State Output Leakage Current In Float		±10	µA	V_{OUT} = 0.4V to V_{CC}
$I_{L(SY)}$	$\overline{SYNC}$ Pin Leakage Current		+10/−40	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100	mA	

Over specified temperature and voltage range.

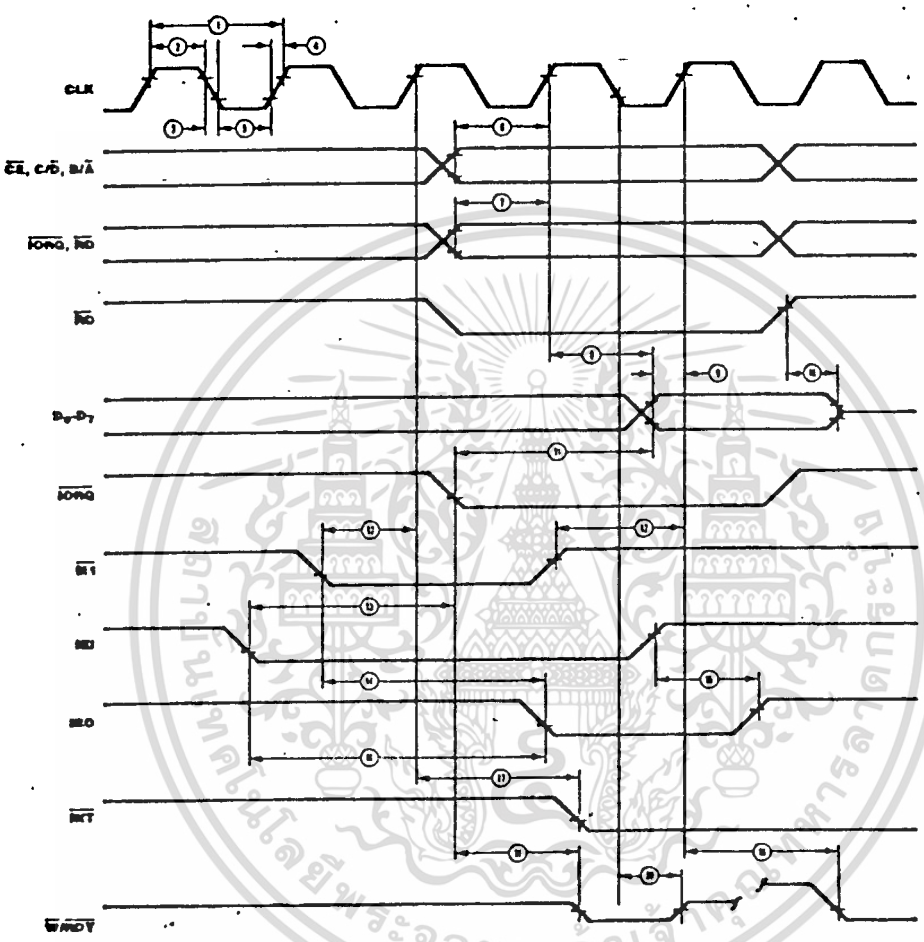
CAPACITANCE

Symbol	Parameter	Min	Max	Unit
C	Clock Capacitance		40	pf
C_{IN}	Input Capacitance		5	pf
C_{OUT}	Output Capacitance		15	pf

Over specified temperature range, f = 1 MHz.
Unmeasured pins returned to ground.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

AC CHARACTERISTICS TIMING



2044-012

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

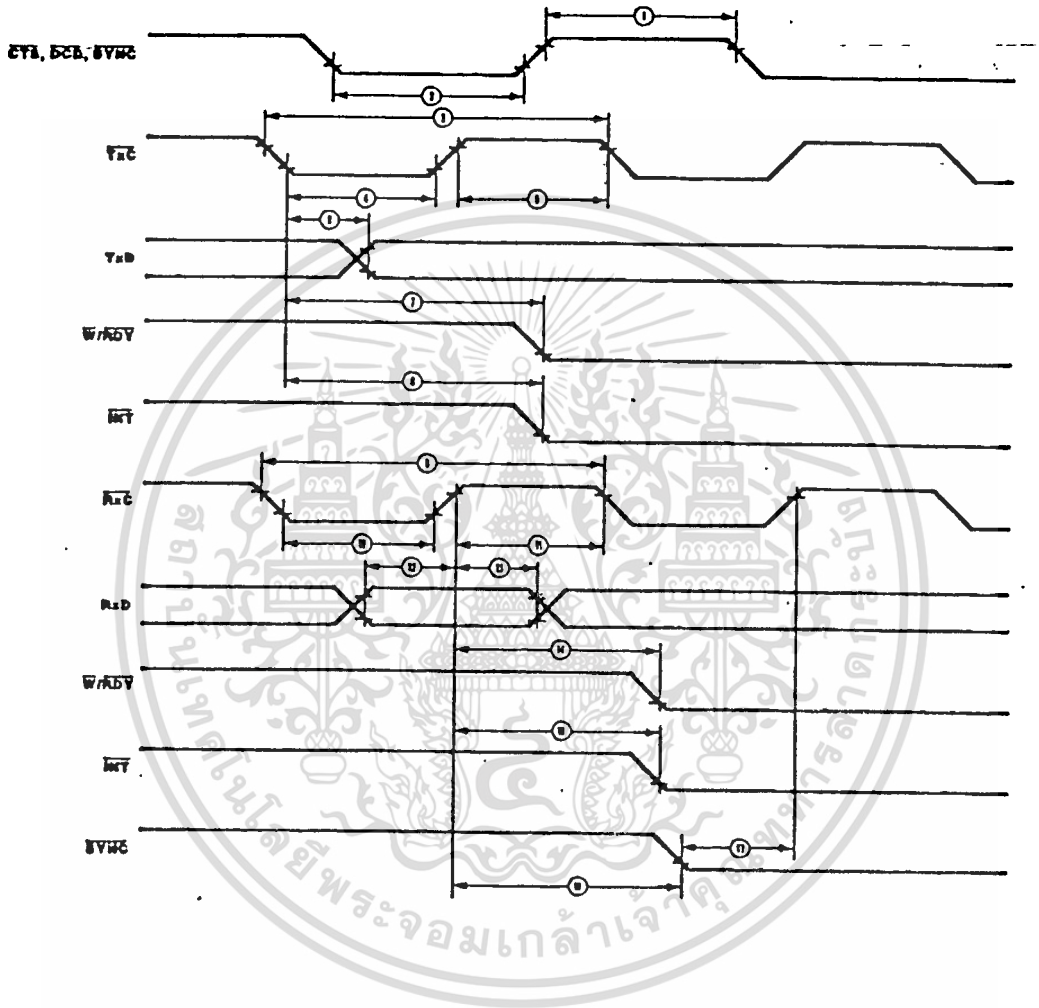
AC CHARACTERISTICS*

Number	Symbol	Parameter	Z80 SIO		Z80A SIO		Z80B SIO	
			Min	Max	Min	Max	Min	Max
1	TcC	Clock Cycle Time	400	4000	250	4000	165	4000
2	TwCh	Clock Width (High)	170	2000	105	2000	70	2000
3	TfC	Clock Fall Time		30		30		15
4	TrC	Clock Rise Time		30		30		15
5	TwCl	Clock Width (Low)	170	2000	105	2000	70	2000
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		60	
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		60	
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220		150
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	50		50		30	
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110		90
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		340		160		120
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		75	
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	200		140		120	
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		300		190		160
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100		70
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100		70
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		200		150
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Wait Mode)		300		210		175
19	TdC(W/RRf)	Clock ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)		120		120		100
20	TdC(W/RWz)	Clock ↓ to $\overline{W/RDY}$ Float Delay (Wait Mode)		150		130		110
21	Th	Any unspecified Hold when Setup is specified	0		0		0	

*Units in nanoseconds (ns).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

AC CHARACTERISTICS TIMING (Continued)



2044-013

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

AC CHARACTERISTICS (Continued)

Number	Symbol	Parameter	Z80 SIO Min	Z80 SIO Max	Z80A SIO Min	Z80A SIO Max	Z80B SIO Min	Z80B SIO Max	Notes*
1	TwPh	Pulse Width (High)	200		200		200		2
2	TwPl	Pulse Width (Low)	200		200		200		2
3	TcTxC	$\overline{TxC}$ Cycle Time	400	∞	400	∞	330	∞	2
4	TwTxCl	$\overline{TxC}$ Width (Low)	180	∞	180	∞	100	∞	2
5	TwTxCh	$\overline{TxC}$ Width (High)	180	∞	180	∞	100	∞	2
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay (x1 Mode)		400		300		220	2
7	TdTxC(W/RRf)	$\overline{TxC}$ ↓ to W/$\overline{RDY}$ ↓ Delay (Ready Mode)	5	9	5	9	5	9	1
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	5	9	1
9	TcRxC	$\overline{RxC}$ Cycle Time	400	∞	400	∞	330	∞	2
10	TwRxCl	$\overline{RxC}$ Width (Low)	180	∞	180	∞	100	∞	2
11	TwRxCh	$\overline{RxC}$ Width (High)	180	∞	180	∞	100	∞	2
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		0		2
13	ThRxD(RxC)	$\overline{RxC}$ ↑ to RxD Hold Time (x1 Mode)	140		140		100		2
14	TdRxC(W/RRf)	$\overline{RxC}$ ↑ to W/$\overline{RDY}$ ↓ Delay (Ready Mode)	10	13	10	13	10	13	1
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↓ Delay	10	13	10	13	10	13	1
16	TdRxC(SYNC)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ ↓ Delay (Output Modes)	4	7	4	7	4	7	1
17	TsSYNC(RxC)	$\overline{SYNC}$ ↓ to $\overline{RxC}$ ↑ Setup (External Sync Modes)	-100		-100		-100		2

* In all modes, the System Clock rate must be at least five times the maximum data rate. $\overline{RESET}$ must be active a minimum of one complete clock cycle.
1. Units equal to System Clock Periods.
2. Units in nanoseconds (ns).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ORDERING INFORMATION

Z80 SIO/0, 2.5 MHz

40-pin DIP	44-pin LCC
Z8440 PS	Z8444 LM*
Z8440 CS	Z8444 LMB*†
Z8440 PE	
Z8440 CE	
Z8440 CM*	
Z8440 CMB*	

Z80A SIO/0, 4.0 MHz

40-pin DIP	44-pin LCC
Z8440A PS	Z8444A LM*
Z8440A CS	Z8444A LMB*†
Z8440A PE	
Z8440A CE	
Z8440A CM*	
Z8440A CMB*	

Z80B SIO/0, 6.0 MHz

40-pin DIP
Z8440B PS
Z8440B CS

Z80 SIO/1, 2.5 MHz

40-pin DIP	44-pin LCC
Z8441 PS	Z8444 LM*
Z8441 CS	Z8444 LMB*†
Z8441 PE	
Z8441 CE	
Z8441 CM*	
Z8441 CMB*	

Z80A SIO/1, 4.0 MHz

40-pin DIP	44-pin LCC
Z8441A PS	Z8444A LM*
Z8441A CS	Z8444A LMB*†
Z8441A PE	
Z8441A CE	
Z8441A CM*	
Z8441A CMB*	

Z80B SIO/1, 6.0 MHz

40-pin DIP
Z8441B PS
Z8441B CS

Z80 SIO/2, 2.5 MHz

40-pin DIP	44-pin LCC
Z8442 PS	Z8444 LM*
Z8442 CS	Z8444 LMB*†
Z8442 PE	
Z8442 CE	
Z8442 CM*	
Z8442 CMB*	

Z80A SIO/2, 4.0 MHz

40-pin DIP	44-pin LCC
Z8442A PS	Z8444A LM*
Z8442A CS	Z8444A LMB*†
Z8442A PE	
Z8442A CE	
Z8442A CM*	
Z8442A CMB*	

Z80B SIO/2, 6.0 MHz

40-pin DIP
Z8442B PS
Z8442B CS

Codes

First letter is for package; second letter is for temperature.

C = Ceramic DIP
P = Plastic DIP
L = Ceramic LCC
V = Plastic PCC

R = Protopack
T = Low Profile Protopack
DIP = Dual-In-Line Package
LCC = Leadless Chip Carrier
PCC = Plastic Chip Carrier (Leaded)

TEMPERATURE
S = 0°C to +70°C
E = −40°C to +85°C
M* = −55°C to +125°C

FLOW
B = 883 Class B

Example: PS is a plastic DIP, 0°C to +70°C.

†Available soon.
*For Military Orders, contact your local Zilog Sales Office for Military Electrical Specifications

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

1. ซีเอ็ดยูเคชั่น , "คู่มือไอซีสัญญาณและหน่วยความจำ" , บริษัท ซีเอ็ดยูเคชั่นจำกัด , 605 หน้า , 2529
2. ซีเอ็ดยูเคชั่น , "คู่มือเทียบเบอร์ไอซี TTL" , บริษัท ซีเอ็ดยูเคชั่น จำกัด , 398 หน้า , 2529
3. ศิวพงษ์ ตั้งสุจริต , "มิติระดมสมองเอมพีวเตอร์" , วารสารเทคนิคคอมพิวเตอร์กิโลทรอนิคส์ , ฉบับที่ 90 , 2532 , หน้า 203 - 211
4. Roger Powell , "The Challenge of Music Software" , BYTE , vol.11 , No.6 , 1986 , pp.145 - 150
5. Jay Kubicky , "A MIDI Project." , BYTE , vol.11 , No.6 , 1986 , pp. 199 - 208
6. James W. Coffron , "Z-80 Application" , SYBEX Inc. ,295 p. ,1983
7. Roland Corporation , "MIDI guidebook" , Japan , 31 p. , 1988

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ผู้จัดทำขอขอบคุณ

- ผศ. ครรชิต ไมตรี ผู้เป็นอาจารย์ที่ปรึกษาและเอื้อเฟื้ออุปกรณ์ตลอดจนสถานที่ตลอดระยะเวลา 1 ปีของการทำโครงงาน

- คุณอาจ วิเชียรเจริญ , คุณศุภชัย แห่งบริษัท วรันดา , คุณวิศิษฐ์ แห่ง โรงเรียนดนตรีสยามกลการ สาขาปทุมวัน และ นักดนตรีวงบัตเตอร์ฟลายที่ช่วยให้ข้อมูลเกี่ยวกับมิดิตลอดจนเอกสารต่าง ๆ

- คุณวรปัญญา ปฐมกสิกุล และ คุณกาคภูมิ เสตะรัต ที่ช่วยให้คำปรึกษาเกี่ยวกับปัญหาของวงจร

นอกจากนี้ ยังขอขอบคุณเพื่อน ๆ ทุกคนที่ให้กำลังใจและคอยรับฟังปัญหาต่าง ๆ ทำให้โครงงานนี้สำเร็จลงตามที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้