การ์ดมีดี้อินเตอร์เฟสสำหรับเครื่องพีซี

MIDI Interface Card for PC

โดย

นายสุรพงษ์ สิริขจร เลขประจำตัว 35103257

นายประเสริฐ เจริญรุ่งเรืองดี เลขประจำตัว 35103233

อาจารย์ที่ปรึกษา

อ.สมศักดิ์ มิตะถา

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า วิทยาเขตเจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชา วิศวกรรมคอมพิวเตอร์ (สมทบ)

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การ์ดมีดี้อินเตอร์เฟสสำหรับเครื่องพีซี

ผู้จัดทำ

1. นายสุรพงษ์ สิริขจร เลขประจำตัว 35103257
2. นายประเสริฐ เจริญรุ่งเรืองดี เลขประจำตัว 35103233

.....**อาจารย์ที่ปรึกษา**

(.....อ. สมศักดิ์ มิตะลา.....)

การ์ดมีดี้อินเตอร์เฟสสำหรับเครื่องพีซี

สุรพงษ์ สิริขจร
ประเสริฐ เจริญรุ่งเรืองดี
อ.สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ เป็นผลงานในหัวข้อเรื่องการ์ดมีดี้อินเตอร์เฟสสำหรับเครื่องพีซี ซึ่งทำให้เครื่องคอมพิวเตอร์สามารถที่จะเล่นดนตรีได้โดยไม่ต้องใช้นักดนตรีเล่น โดยโครงงานที่ทำขึ้นมานี้ จะประกอบด้วยทั้งฮาร์ดแวร์และซอฟต์แวร์ ซึ่งส่วนของฮาร์ดแวร์จะเป็นวงจรแปลงสัญญาณไปกลับ ระหว่างสัญญาณขนานกับสัญญาณอนุกรม และวงจรสร้างฐานเวลา ส่วนซอฟต์แวร์จะเป็นโปรแกรมที่ใช้รันบนไมโครซอฟต์วินโดวส์ (MS Windows) ซึ่งกำหนดให้มีความสามารถต่าง ๆ เช่น การอีดิตแก้ไขข้อมูลมีดี้, อ่านเขียนข้อมูลไฟล์มีดี้มาตรฐาน (format 1 - multi track) ฯลฯ

MIDI Interface Card for PC

Surapong Sirikajorn
Prasert Charoenrungreungdee
Somsak Mitatha Advisor
1994

Abstract

This thesis is an research about MIDI Interface Card for PC which help Computer to be able to play the song without the musician. This project composes of Hardware and Software part. For the hardware part, it composes of the signal converter circuit which converts the signal between parallel interface and serial interface , and master clock circuit . For the software part, it composes of the program which has an application on MS Windows which is assigned to has various ability such as an editing of MIDI data, read and write standard MIDI file format etc.

สารบัญ

บทที่ 1

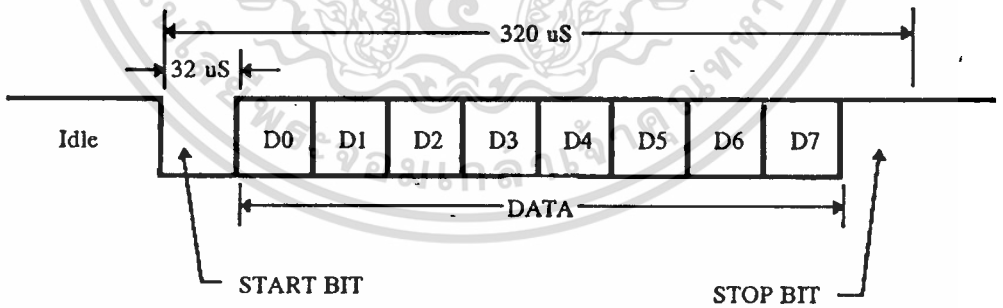
แนะนำให้รู้จักกับระบบมีดี้

(Introduction to MIDI System)

ระบบมีดี้ คือ ระบบการสื่อสารระหว่างเครื่องคนตรีกับเครื่องคนตรี หรือระบบการสื่อสารระหว่างเครื่องคนตรีกับเครื่องคอมพิวเตอร์ เพื่อให้เครื่องคนตรีสามารถที่จะเล่นเพลงได้โดยไม่ต้องใช้นักดนตรีเป็นผู้เล่น

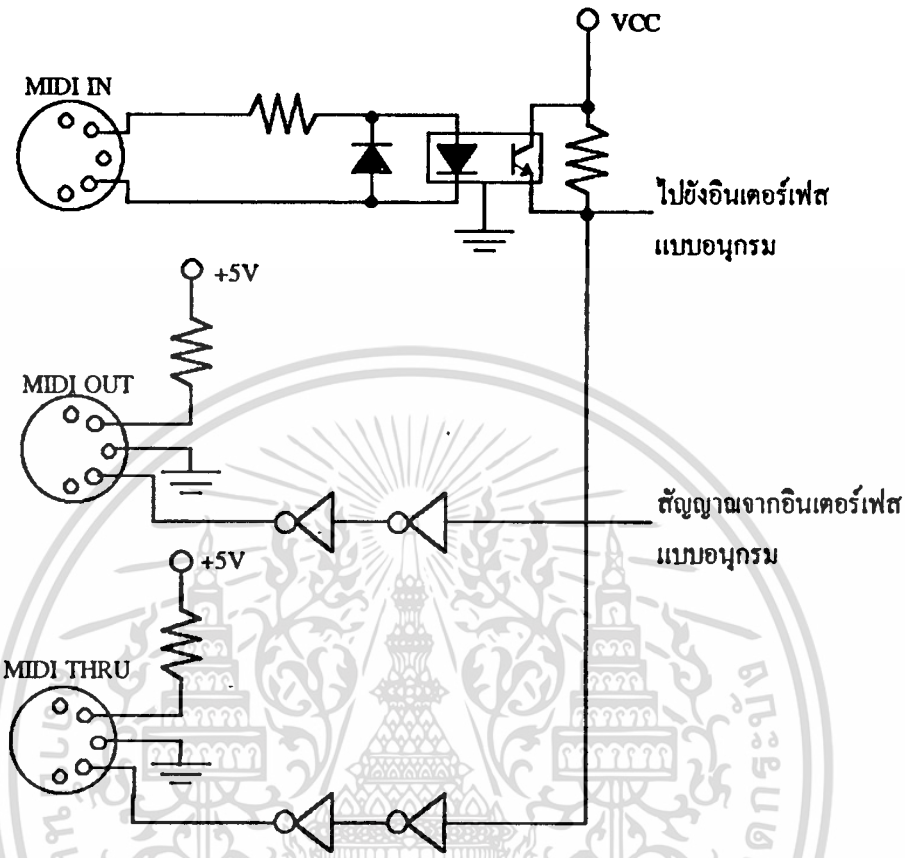
โครงสร้างของระบบมีดี้

ระบบมีดี้เป็นรูปแบบมาตรฐานการสื่อสารข้อมูลแบบหนึ่งที่ใช้กับการสื่อสารระหว่างอุปกรณ์ทางด้านดนตรีที่เป็นดิจิตอล ซึ่งก็คือโปรโตคอลในการสื่อสารแบบหนึ่ง (Communication Protocal) ที่ใช้สื่อสารระหว่างเครื่องคนตรีที่มีระบบคอมพิวเตอร์อยู่ภายใน การสื่อสารดังกล่าวใช้การสื่อสารแบบอนุกรมแบบอะซิงโครนัส (Asynchronous) ความเร็ว 31.25 Kbaud มีขนาดข้อมูล 8 บิต มีทั้งสตาร์ทบิต (Start bit) และสตอพบิต (Stop bit) เพราะฉะนั้นในแต่ละช่วงของข้อมูลจึงมีทั้งหมด 10 บิต ใช้เวลาต่อหนึ่งไบต์เท่ากับ 320 ไมโครวินาที



รูปแสดงรูปแบบของข้อมูลมีดี้มาตรฐานที่มีความเร็วในการส่งข้อมูล 31.25 kbps

พอร์ตสำหรับการสื่อสารจะใช้คอนเนกเตอร์แบบ DIN 5 ขา มีพอร์ต 3 แบบ คือ มีดี้อิน (MIDI IN), มีดี้เอาต์ (MIDI OUT) และมีดี้ทรู (MIDI THRU) โดยที่ช่องมีดี้อินใช้สำหรับรับข้อมูล ช่องมีดี้เอาต์ใช้สำหรับส่งข้อมูล และสุดท้ายช่องมีดี้ทรูใช้ในการผ่านข้อมูลจากช่องมีดี้อินต่อไปที่อุปกรณ์มีดี้ตัวอื่นๆ ทำให้เราสามารถที่จะพ่วงต่ออุปกรณ์มีดี้ได้หลายๆ ตัว



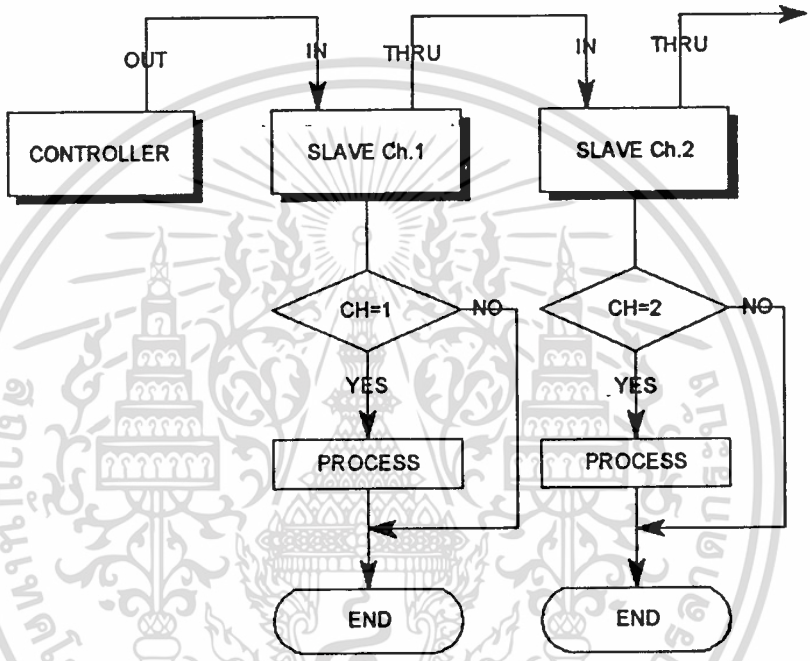
รูปแสดงวงจรอินเตอร์เฟสพื้นฐานของระบบมีดี้

ลักษณะของข้อมูลมีดี้

ข้อมูลที่ส่งออกมาจากตัวซีเควนเซอร์ (Sequencer), ซินทีไซเซอร์ (Synthesizer) หรือเครื่องคอมพิวเตอร์จะรับรู้โดยอุปกรณ์มีดี้ที่เป็นสลาฟ (slave) แต่ละตัวได้โดยการบอกแชนแนล ถ้าแชนแนลที่ตัวลูกตั้งไว้ ตรงกับข้อมูลที่ตัวซีเควนเซอร์ส่งมา สลาฟตัวนั้นก็จะรับข้อมูลดังกล่าวไปประมวลผล ถ้าไม่ใช่แชนแนลที่ตัวของมันตั้งไว้ ข้อมูลดังกล่าวจะไม่มีผลต่อสลาฟตัวนั้น แต่เพราะข้อมูลจะถูกผ่านไปทางพอร์ตทรูให้กับอุปกรณ์ทุกตัว ดังนั้นสลาฟที่มีแชนแนลตรงกับแชนแนลที่ตัวซีเควนเซอร์ต้องการติดต่อด้วยก็จะรับข้อมูลไปประมวลผลเอง แต่ข้อมูลบางประเภทที่ส่งภายใต้ระบบมีดี้ก็ต้องการจะติดต่อกับทุกๆ แชนแนลหรืออุปกรณ์ทุกตัวในระบบ เช่น เรื่องของความเร็วของเพลง เป็นต้น

แชนแนลของระบบมีดี้

ระบบมีดี้จะมีแชนแนลที่ติดต่อได้เพียง 16 แชนแนล อุปกรณ์มีดี้แต่ละตัวจะมีหมายเลขแชนแนลประจำตัวที่จะรับและส่งข้อมูล อุปกรณ์ส่วนใหญ่ในปัจจุบันสามารถจะรับและส่งในหลายแชนแนลได้พร้อมๆ กัน ซึ่งอุปกรณ์ดังกล่าวเรียกว่า Multitimbral Module ซึ่งภายในก็เปรียบเสมือนเครื่องที่มีอุปกรณ์มีดี้อยู่หลายๆ ตัว



รูปแสดงขบวนการติดต่อข้อมูลของแต่ละแชนแนลในอุปกรณ์มีดี้

ไบต์สถานะกับไบท์ข้อมูล (Status byte and Data byte)

ภายในข้อมูลมีดี้ 1 เหตุการณ์ (event) จะสามารถแบ่งได้เป็น 2 ชนิด คือ ไบต์สถานะและไบต์ข้อมูล โดยลักษณะของไบต์สถานะจะมีบิต 7 (MSB) เป็น "1" ส่วนไบต์ข้อมูลจะมีบิตที่ 7 เป็น "0" ตัวไบต์ข้อมูลจะเป็นเสมือนเฮดเดอร์ของชุดข้อมูลมีดี้ที่กำลังส่งมา (Package) เพื่อบอกว่าเป็นเหตุการณ์ใด

ข้อมูลเฉพาะแชนแนล (Channel type message)

ข้อมูลมีดี้ชนิดนี้เวลาทำการส่งจะมีไบต์ทำหน้าที่บอกแชนแนลที่ต้องการจะติดต่อด้วยแล้วจึงตามด้วยข้อมูลที่ต้องการติดต่อ คือจะประกอบด้วยไบต์สถานะแล้วตามด้วยไบต์ข้อมูล

จำนวนหนึ่งหรือสองไบต์ ไบต์สถานะจะมีค่าตั้งแต่ 80h ถึง Efh ซึ่ง 4 บิตบน (บิต 7 - บิต 4) จะบอกความหมายของคำสั่ง ส่วน 4 บิตล่าง (บิต 3 - บิต 0) จะแสดงถึงแชนแนลที่ต้องการจะส่งไปให้ ซึ่งมีคำสั่งต่างๆ ดังนี้

- Note Off

Status	Second	Third
8nH	kkH	vvH
9nH	kkH	00H

ใช้สำหรับปิดเสียงของตัวโน้ตตัวที่ kkH

- Note On

Status	Second	Third
9nH	kkH	vvH

ใช้สำหรับเปิดเสียงตัวโน้ตตัวที่ kkH ที่ระดับความดัง vvH

- Polyphonic Key Pressure

Status	Second	Third
AnH	kkH	vvH

ใช้สำหรับเร่งระดับความดังของเสียงขึ้นลงโดยยังไม่ปิดเสียง

- Program Change

Status	Second
CnH	ppH

ใช้สำหรับเปลี่ยนชนิดของเครื่องคนตรี

- Channel Pressure

Status	Second
DnH	vvH

ใช้สำหรับเลือกโหมดการวัดแรงในการกดคีย์แบบทั้งแชนแนล

- Pitch-Bend Change

Status	Second	Third
EnH	llH	mmH

ใช้สำหรับเปลี่ยนระดับความถี่เสียงขึ้นหรือลง

- All Sounds Off

Status	Second	Third
BnH	78H	00H

ใช้สำหรับปิดเสียงทั้งหมดในแชนแนลที่กำหนด

- Reset All Controllers

Status	Second	Third
BnH	79H	00H

ใช้สำหรับรีเซ็ตตัวซินทีไซเซอร์ให้เหมือนกับตอนเริ่มต้นเปิดเครื่อง

- All Notes Off

Status	Second	Third
BnH	7BH	00H

ใช้สำหรับปิดเสียงที่เกิดจากคำสั่ง Note On ทุกตัว ในแชนแนลที่กำหนด

- OMNI Off

Status	Second	Third
BnH	7CH	00H

ใช้สำหรับเลือกโหมดการทำงานแชนแนลที่กำหนดรับเฉพาะข้อมูลของตนเอง

- OMNI On

Status	Second	Third
BnH	7DH	00H

ใช้สำหรับเลือกโหมดการทำงานให้แชนแนลที่กำหนดรับข้อมูลของทุกแชนแนลได้

- MONO

Status	Second	Third
BnH	7EH	mmH

ใช้สำหรับเลือกโหมดการทำงานให้กำเนิดเสียงได้ทีละเสียงเดียว

- POLY

Status	Second	Third
BnH	7FH	00H

ใช้สำหรับเลือกโหมดการทำงานให้กำเนิดเสียงได้ทีละหลายเสียงพร้อมกัน

ข้อมูลที่ติดต่อทั้งระบบ (System type message)

ข้อมูลชนิดนี้จะไม่มีการอ้างถึงแชนแนลที่ต้องการจะติดต่อด้วย แต่จะเป็นการควบคุมโดยรวม ซึ่งมีคำสั่งต่างๆ ดังนี้

- Active Sensing

Status
FEH

ใช้สำหรับป้องกันความผิดพลาดที่เกิดจากสายสัญญาณมีดี้

- System Exclusive Message

Status	Data
F0H	iiH, ddH,.....,eeH
F7H	

ใช้สำหรับส่งข้อมูลเฉพาะสำหรับอุปกรณ์มีดี้ เช่น

⇒ GS Reset

Status	Data Byte	Status

F0H 41H, 10H, 42H, 12H, 40H, 00H, 7FH, 00H, 41H F7H

Byte	Description	
F0H	Exclusive status	
41H	Manufacturer's ID	(Roland)
10H	Device ID	(UNIT # = 17)
42H	Model ID	(GS)
12H	Command ID	(DT1)
40H	Address MSB	
00H	|	
7FH	Address LSB	
00H	Data	(GS reset)
41H	Check sum	
F7H	EOX	(End of exclusive)

⇒ Turn General MIDI System On

Status	Data Byte	Status
F0H	7EH, 7FH, 09H, 01H	F7H

Byte	Description	
F0H	Exclusive status	
7EH	ID Number	(Universal non-real time message)
7FH	ID of target device	(Broadcast)
09H	sub - ID # 1	(General MIDI message)
01H	sub - ID # 2	(General MIDI on)
F7H	EOX	(End of exclusive)

บทที่ 2

โครงสร้างและหลักการทำงานของฮาร์ดแวร์มีดี้
(Principle of MIDI Hardware)

ส่วนของฮาร์ดแวร์ในโครงงานที่ทำขึ้นนี้ มีลักษณะเป็นการ์ดสำหรับใช้เสียบลงในสล็อตของเครื่องคอมพิวเตอร์ เพื่อทำหน้าที่เป็นตัวอินเตอร์เฟสกับเครื่องดนตรีหรืออุปกรณ์มีดี้ต่างๆ ทำให้สามารถใช้เครื่องคอมพิวเตอร์เล่นดนตรีได้

หลักการทำงานของวงจร

ส่วนประกอบฮาร์ดแวร์ของโครงงานที่ทำขึ้นนี้จะประกอบด้วยสามส่วนใหญ่ๆ คือ ส่วนที่ทำหน้าที่เป็นตัวกำหนดหมายเลขพอร์ตต่างๆ ให้กับการ์ดมีดี้, ส่วนที่แปลงข้อมูลจากขนานเป็นอนุกรมหรือจากอนุกรมเป็นขนานซึ่งใช้ไอซีเบอร์ Z80-SIO เป็นตัวทำหน้าที่ในส่วนนี้ และส่วนของตัวสร้างฐานเวลาจะใช้ไอซีเบอร์ 8253

Port No.	Function
02A0h	Data port A of Z80-SIO
02A1h	Control port A of Z80-SIO
02B0h	Data port B of Z80-SIO
02B1h	Control port B of Z80-SIO
02A2h	Channal 0 of 8253
02A3h	Channal 1 of 8253
02B2h	Channal 2 of 8253
02B3h	Control port of 8253

ตารางแสดงหมายเลขพอร์ตต่างๆ บนการ์ดมีดี้

ไอซีเบอร์ Z80-SIO จะทำหน้าที่เป็นตัวรับและตัวส่งสัญญาณมีดี้ โดยที่จะใช้แชนแนล A เป็นตัวส่ง (MIDI OUT) และให้แชนแนล B เป็นตัวรับ (MIDI IN) ซึ่งสามารถที่จะควบคุม

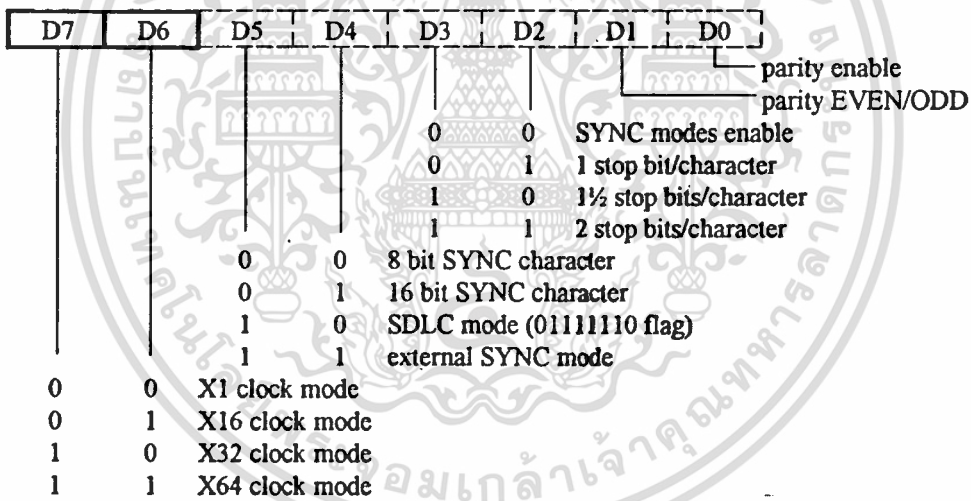
อุปกรณ์มีดี้ต่างๆ ได้จำนวน 16 แชนแนล สัญญาณในการรับส่งข้อมูลมีดี้มีมาตรฐานอยู่ที่ความเร็ว 31.25 กิโลไบต์ต่อวินาที (Kbps) และเนื่องจากความถี่สัญญาณนาฬิกาที่ป้อนให้กับ Z80-SIO เท่ากับ 2 Mhz ซึ่งสามารถคำนวนหาค่าตัวหารได้ดังนี้

$$\text{divisor} = \frac{\text{clock (Hz)}}{31.25\times10^{3}}$$

$$= \frac{2\times10^{6}}{31.25\times10^{3}}$$

$$= 64$$

เมื่อคำนวนได้ค่าของตัวหารเรียบร้อยแล้วก็ให้ใช้ค่านี้ในเวลาเขียนโปรแกรมเพื่อโปรแกรมตัว Z80-SIO ให้ได้ความถี่ที่ต้องการ ซึ่งตัว Z80-SIO จะมีรีจิสเตอร์ (register) สำหรับกำหนดค่าของตัวหารดังในตาราง



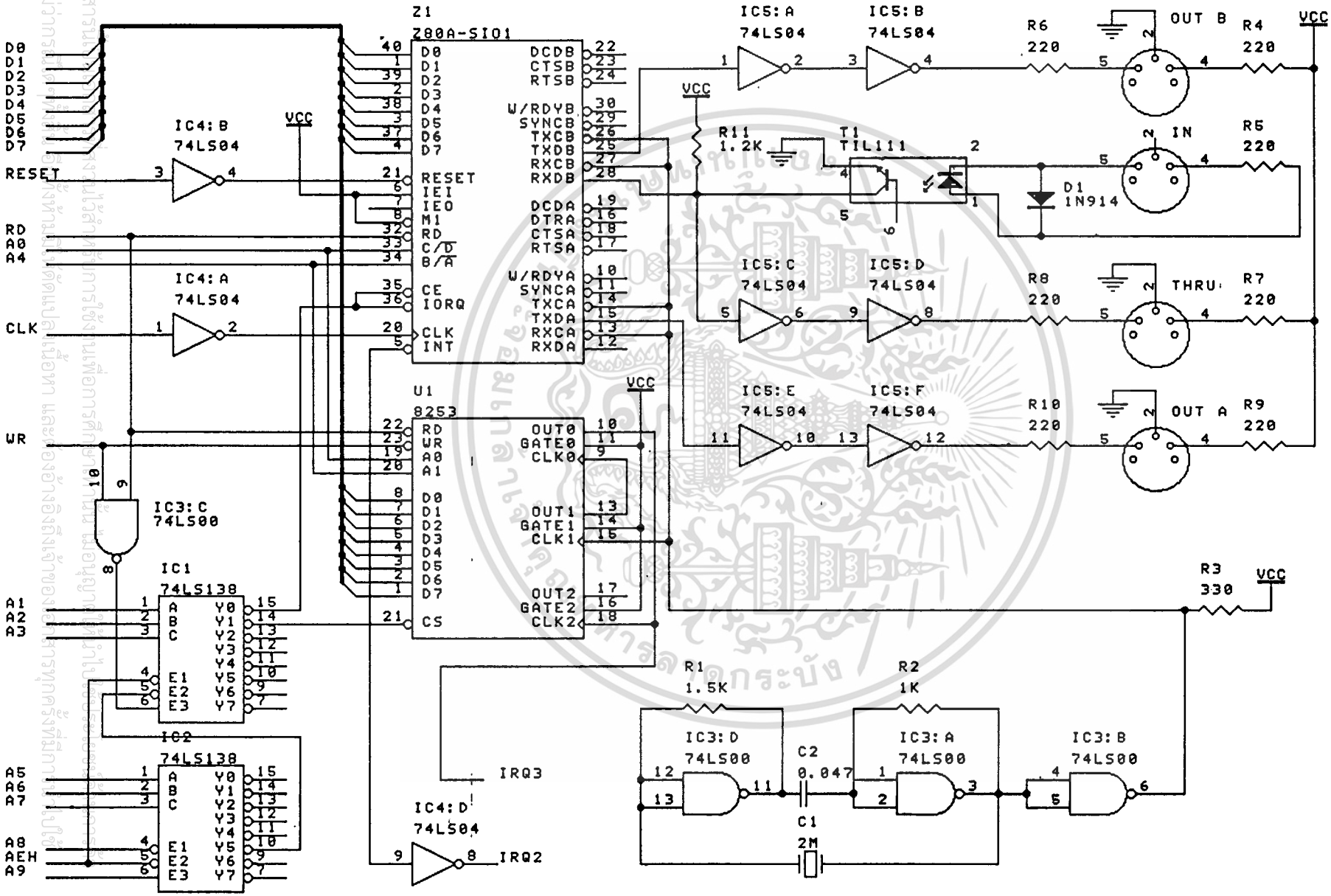
รูปแสดงรายละเอียดของ Write Register 4

* ดูรายละเอียดของรีจิสเตอร์ทั้งหมดได้ที่ภาคผนวกท้ายเล่ม

ไอซีเบอร์ 8253 จะทำหน้าที่เป็นตัวการหนดฐานเวลาที่ใช้ในการเล่นตัวโน้ต (tempo) ตัวอย่างเช่น สมมติว่าค่าความเร็วเทมโป้ (tempo) เท่ากับ 120 ก็คือจะต้องเล่นตัวโน้ตตัวดำจำนวน 120 ตัว ในเวลา 1 นาที ดังนั้นจะเห็นได้ว่า เมื่อค่าเทมโป้มากขึ้น จังหวะของเพลงก็จะเร็วขึ้น แต่ถ้ามีค่าน้อยลง จังหวะของเพลงก็จะช้าลง

8253 จะประกอบด้วยตัวหารความถี่ที่โปรแกรมได้อยู่ 3 แชนแนล (แชนแนล 0 ถึง แชนแนล 2) และมีแชนแนลที่ในการควบคุมอยู่ 1 แชนแนล จากวงจรจะต่อสัญญาณคล็อคความถี่

2 MHz เข้าไปที่ขา CLK1 ของ 8253 ซึ่งจะทำการโปรแกรมแชนแนล 1 ให้หารความถี่ด้วย 2 เหลือความถี่ 1 MHz ออกมาที่ขา OUT1 ซึ่งจะต่อไปเข้าที่ CLK0 ซึ่งที่แชนแนลที่ 0 นี้จะใช้เป็นตัวกำหนดค่าของเทมโป้ และสุดท้ายก็จะต่อสัญญาณที่ขา OUT0 ไปเข้าขา CLK2 ซึ่งที่แชนแนลที่ 2 นี้จะกำหนดให้มีค่าของตัวนับคงที่ ที่ค่า 240 ซึ่งหมายถึงมีความละเอียด 1/240 ของโน้ตตัวดำ


Z1
Z80A-SIO1
U1
8253
IC1
74LS138
IC2
74LS138
IC3:C
74LS00
IC3:D
74LS00
IC3:A
74LS00
IC3:B
74LS00
IC4:A
74LS04
IC4:B
74LS04
IC4:D
74LS04
IC5:A
74LS04
IC5:B
74LS04
IC5:C
74LS04
IC5:D
74LS04
IC5:E
74LS04
IC5:F
74LS04
T1
TIL111
D1
IN914
R1
1.5K
R2
1K
R3
330
R4
220
R5
220
R6
220
R7
220
R8
220
R9
220
R10
220
R11
1.2K
C1
2M
C2
0.047
OUT B
IN
THRU
OUT A
VCC
RESET
IRQ2
IRQ3
D0
D1
D2
D3
D4
D5
D6
D7
RD
A0
A4
CLK
WR
A1
A2
A3
A5
A6
A7
A8
AEH
A9

บทที่ 3

การออกแบบซอฟต์แวร์ซีเควนเซอร์
(Sequencer Software Designing)

ในส่วนของซอฟต์แวร์จะทำหน้าที่ควบคุมส่วนฮาร์ดแวร์ให้ทำการรับส่งข้อมูลของมีดี้ ซึ่งจะมีลักษณะการสื่อสารกันแบบอนุกรม เพื่อทำการบันทึกหรือปรับแต่งข้อมูลมีดี้ เช่น การทำจังหวะของดนตรีให้เร็วขึ้นหรือช้าลง การปรับเสียงดังค่อย การบันทึกให้เหมือนการเล่นดนตรีจริง (real time record) การบันทึกทีละตัวโน้ต (step record) ฯลฯ ซึ่งซอฟต์แวร์เหล่านี้มักจะเรียกว่า "โปรแกรมซีเควนเซอร์"

หลักการออกแบบโปรแกรม

โปรแกรมซีเควนเซอร์ที่ได้เขียนขึ้นมานี้ เขียนขึ้นโดยใช้คอมไพล์เลอร์ของบริษัทไมโครซอฟต์ คือ "Microsoft Visual C++ 1.0" โดยสามารถแบ่งออกเป็น 2 ส่วนหลักๆ ดังนี้

- *โปรแกรมส่วนที่จะทำการติดต่อกับฮาร์ดแวร์โดยตรง* ซึ่งจะเขียนเป็นไลบรารี่แบบไดนามิกลิงก์ (DLL - Dynamic Link Librarys)
- *ส่วนโปรแกรมหลัก* ที่ทำหน้าที่ในการแสดงผลออกที่หน้าจอ, อีดิตแก้ไขข้อมูล ฯลฯ

โปรแกรมส่วนที่ทำการติดต่อกับฮาร์ดแวร์โดยตรง

โปรแกรมในส่วนนี้จะเขียนเป็นไลบรารี่แบบไดนามิกลิงก์ ซึ่งจะประกอบด้วยฟังก์ชั่นต่างๆ ซึ่งจะทำหน้าที่ในการรับส่งข้อมูลมีดี้, การเซ็ตค่าเริ่มต้นให้กับการ์ด, อ่านค่าฐานเวลา ฯลฯ ดังต่อไปนี้

- void InstallMidi(void);
  ใช้สำหรับเซ็ตค่าเริ่มต้นให้กับตัวการ์ดมีดี้
- void Transmit(int count, unsigned char *data);
  ใช้สำหรับส่งชุดของข้อมูลออกไปที่พอร์ตมีดี้เอาต์
- int Receive(void);
  ใช้สำหรับรับข้อมูลที่เข้ามาทางพอร์ตมีดี้อินทีละ 1 ไบต์

- void SetTempo(unsigned tempo);

  ใช้สำหรับเซ็ตค่าความเร็วของฐานเวลา

- void SendVolume(unsigned char channel, unsigned char volume);

  ใช้สำหรับเปลี่ยนค่าความดังเสียงของแชนแนลที่ต้องการ

- void SendNote(unsigned char channel, unsigned char note, unsigned char velocity)
  ;

  ใช้สำหรับส่งตัวโน้ตที่ต้องการออกไปที่พอร์ตมีดี้เอาต์

- void SendProgram(unsigned char channel, unsigned char program);

  ใช้สำหรับเปลี่ยนชนิดของเครื่องคนตรี

- void SendPanpot(unsigned char channel, unsigned char panpot);

  ใช้สำหรับเลือกความสมดุลย์ของเสียงด้านซ้ายและขวา (balance)

- void SendPitchbend(unsigned char channel, int value);

  ใช้สำหรับเปลี่ยนความถี่เสียงให้สูงขึ้นหรือต่ำลง

- void GS_Reset(void);

  ใช้สำหรับเลือกชุดของเครื่องคนตรีตามมาตรฐาน GS format

- void GeneralMidiSystemOn(void);

  ใช้สำหรับเลือกชุดของเครื่องคนตรีตามมาตรฐาน General MIDI

- void MidiReset(void);

  ใช้สำหรับรีเซ็ตตัวซินทีไซเซอร์ให้เหมือนกับตอนเริ่มเปิดเครื่อง

- void AllNotesOff(unsigned char channel);

  ใช้สำหรับปิดเสียงที่เกิดจากคำสั่ง Note On

- void AllSoundsOff(unsigned char channel);

  ใช้สำหรับปิดเสียงทั้งหมดในแชนแนลที่ต้องการ

- void ResetAllControllers(unsigned char channel);

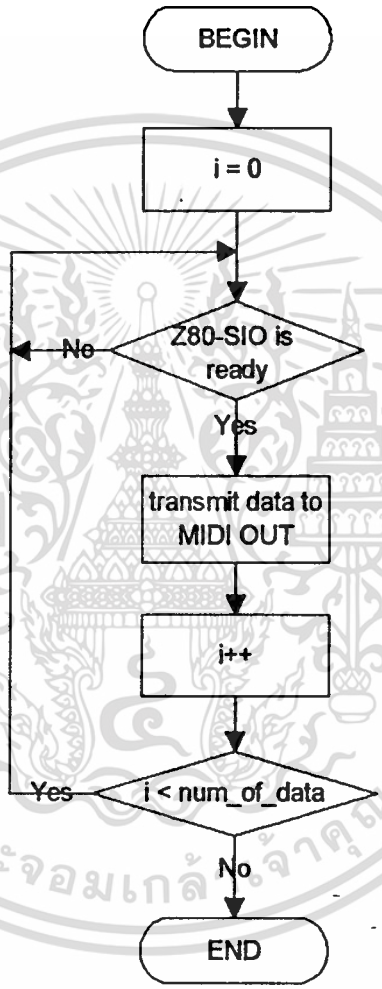
  ใช้สำหรับรีเซ็ตค่าคอนโทรลเลอร์ต่างๆ ให้เป็นค่าเริ่มต้น (เช่น Panpot, Volume)

- void ResetTime(void);

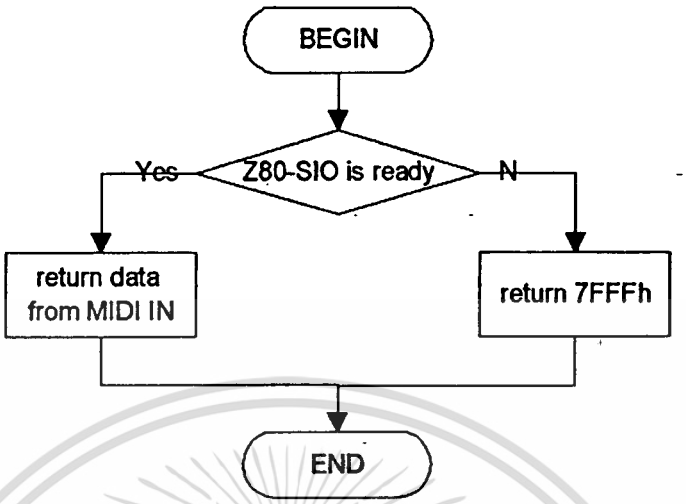
ใช้สำหรับรีเซ็ตค่าฐานเวลาให้เป็นค่าเริ่มต้น

- TIME GetTime(void);

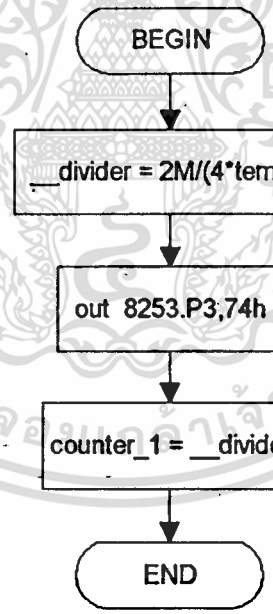
  ใช้สำหรับอ่านค่าฐานเวลาจากการ์ด



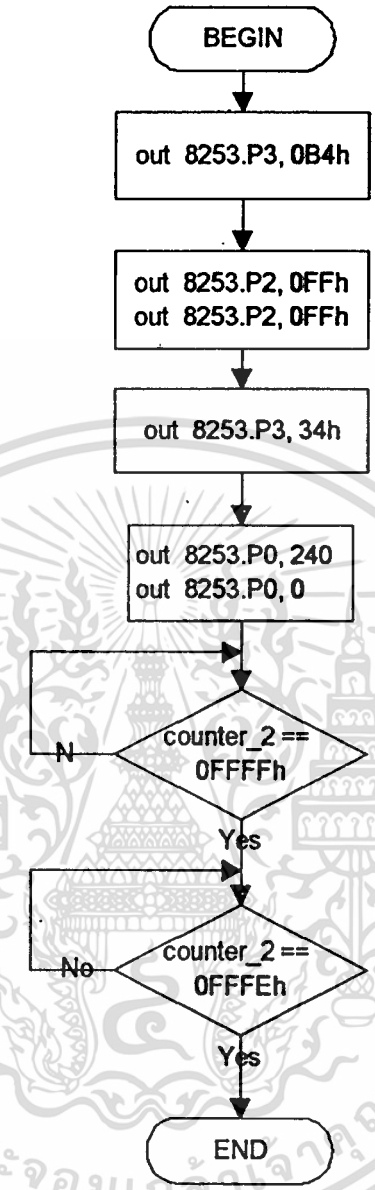
รูปผังแสดงหลักการในการส่งสัญญาณมีดี้ (transmit)



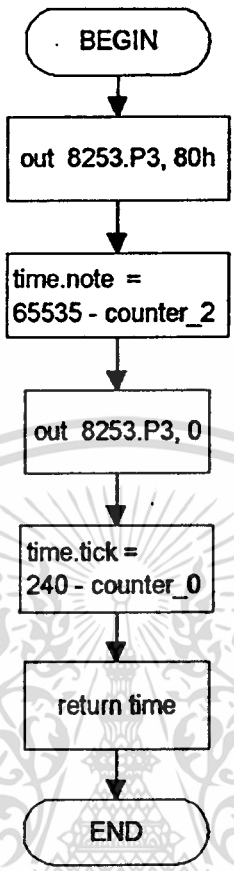
รูปผังแสดงหลักการในการรับสัญญาณมีดี้ (receive)



รูปผังแสดงหลักการเซ็ตค่าเทมโป้ (set tempo)



รูปผังแสดงหลักการเซ็ตค่าเริ่มต้นให้กับฐานเวลา (reset time)



รูปผังแสดงหลักการอ่านค่าเวลาจากฐานเวลา (get time)

ส่วนโปรแกรมหลัก (Main Program)

โปรแกรมในส่วนนี้จะเขียนด้วย MS Visual C++ 1.0 ซึ่งจะทำหน้าที่ต่างๆ เช่น การเปิดไฟล์, การแสดงผลที่หน้าจอ, การอีดิตแก้ไขข้อมูล ฯลฯ ดังที่จะได้อธิบายดังต่อไปนี้

การออกแบบโครงสร้างของข้อมูล

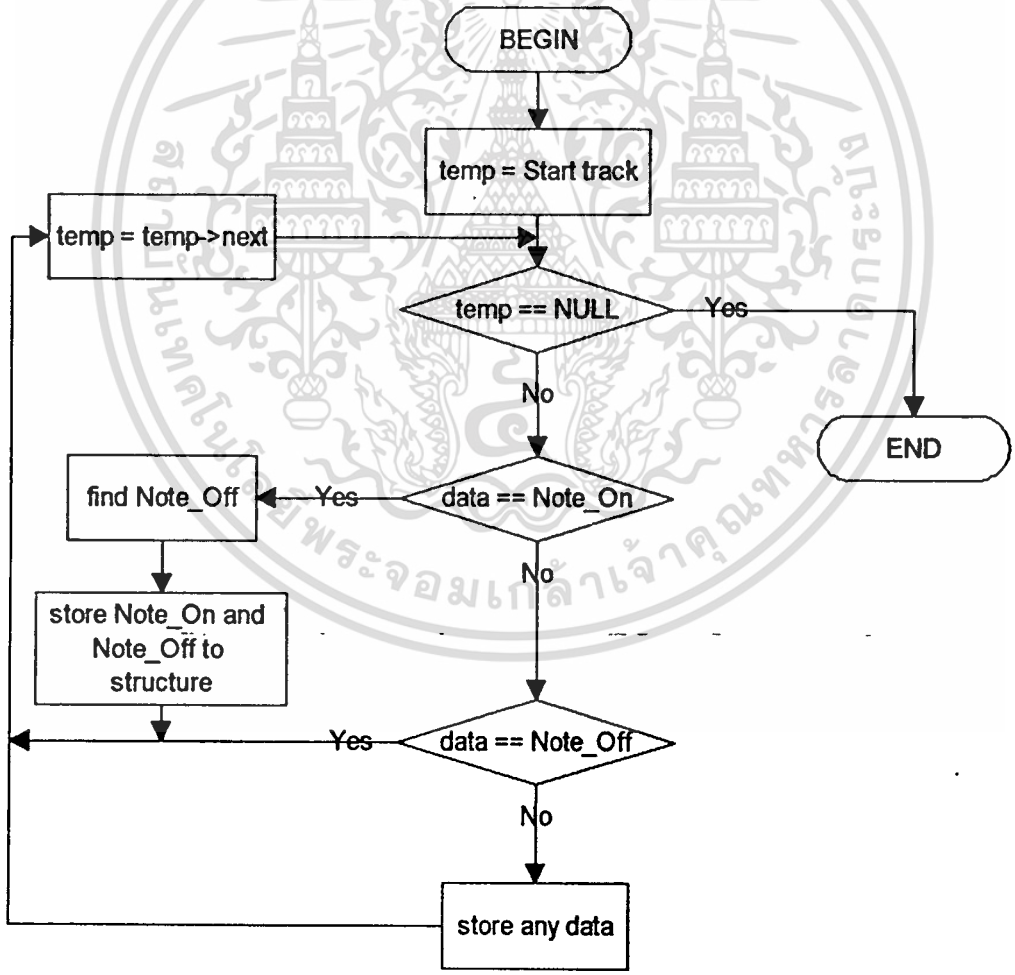
- ระบบข้อมูลมี 2 โครงสร้าง (Structure)

1. รูปแบบที่ใช้ส่งออกเอาท์พุท
2. รูปแบบที่ใช้ปรับปรุง

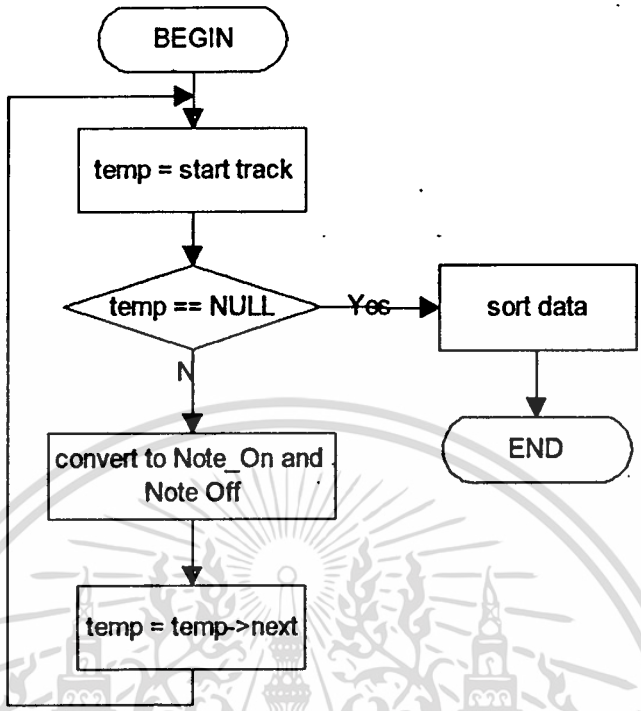
- ส่วนปรับปรุงข้อมูล

ทำการแปลงจากรูปแบบข้อมูลที่ใช้ส่งออกเอาท์พุท ให้อยู่ในรูปแบบข้อมูลที่ใช้ปรับปรุง ซึ่งรูปแบบของข้อมูลที่ใช้ปรับปรุงประกอบด้วยส่วนที่สำคัญ ๆดังนี้ เวลาเริ่มต้นของโน้ต, ข้อมูลโน้ต, เวลาสิ้นสุดของโน้ต

- การแปลงรูปแบบข้อมูลที่ใช้ส่งออกเอาท์พุทให้อยู่ในรูปแบบข้อมูลที่ใช้ปรับปรุง โดยทำการอ่านข้อมูลของรูปแบบที่ใช้ส่งออกเอาท์พุททุก ๆ โหนด (node) ถ้าข้อมูลไม่ใช่โน้ต-ออน (Note- On) หรือ โน้ต-ออฟ (Note - Off) ก็ทำการเก็บข้อมูลนั้น แล้วเลื่อนตัวชี้ข้อมูลเพื่ออ่านข้อมูลถัดไปได้เลย แต่ถ้าข้อมูลเป็นโน้ต-ออน จะต้องใช้ตัวชี้ข้อมูลอีกตัวหนึ่งเป็นตัวเลื่อนเพื่อหาตำแหน่งของโน้ต-ออฟ เมื่อได้ข้อมูลของโน้ต-ออฟแล้ว ก็จะได้ระยะเวลาในการกำเนิดเสียง จากลักษณะการเก็บข้อมูลดังกล่าวจะเห็นว่า เป็นการช่วยให้การปรับปรุงข้อมูลเป็นไปได้ง่าย การปรับปรุงที่เกี่ยวกับข้อมูลเช่น ต้องการเพิ่ม หรือ ลดค่าระยะเวลาในการเกิดเสียงของตัวโน้ตก็ทำการแก้ไขได้เลยหรือ ต้องการแก้ไขให้โน้ตมีเสียงดังขึ้น, เสียงเบาลง ต้องการแก้ไขระดับโน้ต

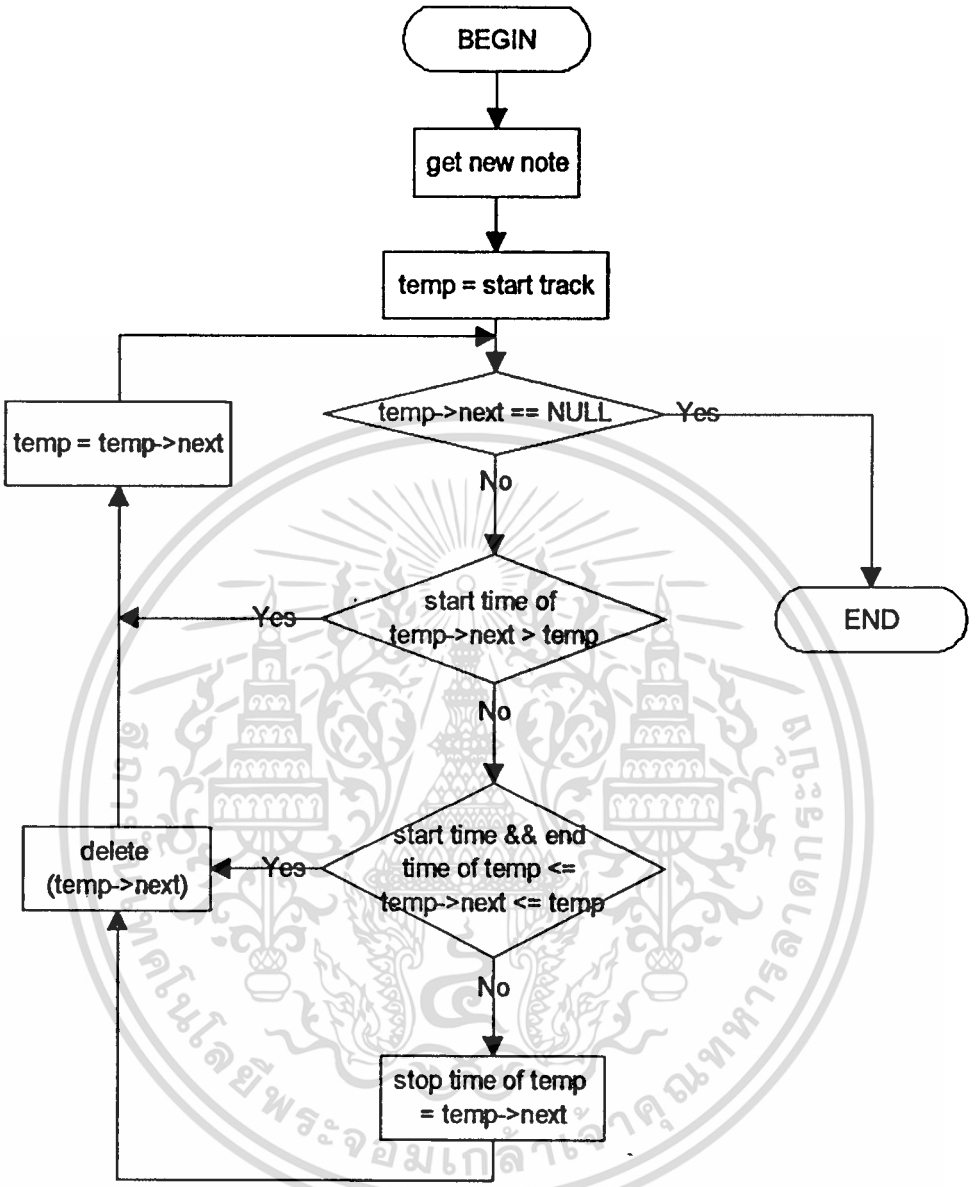


รูป Flow chart การแปลงรูปแบบข้อมูลที่ใช้ส่งออกเอาท์พุทให้อยู่ในรูปแบบข้อมูลที่ใช้ปรับปรุง



รูป Flow chart การแปลงรูปแบบข้อมูลที่ใช้ปรับปรุงให้อยู่ในรูปแบบข้อมูลที่ใช้ส่งออกเอาท์พุท

- ลักษณะการแปลงรูปแบบข้อมูลที่ใช้ปรับปรุง ให้กลับไปเป็นรูปแบบเดิม ทำโดยกำหนดตัวชี้ข้อมูลให้ชี้ที่โหนดแรก แล้วทำการแปลงข้อมูลของโหนดที่ชี้ ก่อนทำการแปลงต้องตรวจสอบว่าโหนดที่ชี้เป็นข้อมูลของโน้ต-ออน หรือ ข้อมูลอื่น ถ้าเป็นข้อมูลอื่น ๆ ก็ทำการโหลดค่าให้กับอีกรูปแบบได้เลย แต่ถ้าเป็นข้อมูลโน้ต-ออน จะต้องแปลงค่ากลับไปเป็นข้อมูลโน้ต-ออน และข้อมูลโน้ต-ออฟ หลังจากการแปลงก็ทำการเลื่อนตัวชี้ไปที่โหนดถัดไปจนหมดข้อมูล ถึงขั้นตอนนี้จะได้ข้อมูลครบเหมือนก่อนที่มีการแปลง แต่ข้อมูลขณะนี้อาจมีการเรียงลำดับค่าของเวลาที่ไม่ถูกต้อง จึงต้องมีการเรียงลำดับข้อมูลตามค่าของเวลาใหม่ ซึ่งโปรแกรมนี้ได้ใช้บับเบิลสอร์ท (Bubble Sort)



รูป Flow chart การ insert ข้อมูลโน้ต

- เมื่อผู้ใช้เลือกการเพิ่มข้อมูลโน้ต โปรแกรมก็ต้องทำการคำนวณ หาตำแหน่งเวลาเริ่มต้นของข้อมูลที่จะเพิ่ม เมื่อได้ค่าเวลาเริ่มต้นแล้ว โปรแกรมก็จะทำการจองเนื้อที่เพื่อเก็บข้อมูลใหม่นี้พร้อมทั้งกำหนดค่าเริ่มต้นให้กับตัวแปรต่าง ๆ จากนั้นก็ต้องตรวจสอบว่าถ้าเวลาเริ่มต้นของข้อมูลใหม่นี้ช้ากว่า (มีค่าเวลามากกว่า) ค่าเวลาเริ่มต้นของข้อมูลสุดท้ายในแทรคนั้น ก็นำข้อมูลใหม่นี้ต่อท้ายข้อมูลเดิมได้เลย แต่ถ้าค่าเวลาเริ่มต้นของข้อมูลใหม่นี้อยู่ระหว่างข้อมูลในแทรค ก็ต้องทำการหาตำแหน่งที่เหมาะสมเพื่อแทรกข้อมูลนี้ ก็ทำโดยกำหนดตัวชี้ข้อมูลให้ไปชี้

ที่ตำแหน่งเริ่มต้นของ ข้อมูลในแทรคนั้น แล้วทำการเลื่อนตัวชี้ข้อมูลเพื่อหาข้อมูลที่มีค่าเวลาเริ่มต้นที่ถัดจาก ข้อมูลที่ต้อง การแทรก เมื่อพบแล้วจึงทำการแทรคข้อมูลใหม่นี้

แต่ในกรณีที่การแทรกข้อมูลเกิดการทับกัน ก็จะทำให้ผลของการส่งข้อมูลที่ทับกันนั้นออกไปที่เอาท์พุทเกิดเสียงที่ผิดพลาดได้ ดังนั้นหลังจากการแทรกข้อมูลทุกครั้ง จะมีการตรวจสอบว่าเกิดการทับกัน, เหลื่อมกันหรือไม่ ถ้าพบว่ามี ก็จะทำการคำนวณหาค่าเวลาที่ครอบคลุมที่สุด แล้วทำการลบโหนดที่ไม่จำเป็นทิ้งไป หลังจากที่มีการแก้ไขข้อมูลเสร็จแล้ว ก็ต้องทำการวาดพื้นที่ใช้งานใหม่เพื่อให้ทันต่อความเป็นจริงมากที่สุด

- การแก้ไขข้อมูลโน้ต เมื่อต้องการแก้ไขข้อมูลโน้ตที่ต้องการ ก็ทำได้โดยการดับเบิ้ลคลิ้กที่ปุ่มซ้ายของเมาส์ในตำแหน่งของโน้ตที่ต้องการ จากนั้นโปรแกรมจะแสดงกรอบข้อความ(วินโดว์ชนิดหนึ่ง) พร้อมกับข้อมูลต่าง ๆ เพื่อให้ทำการแก้ไข และเพื่อป้องกันการป้อนข้อมูลที่ผิดรูปแบบโปรแกรมจึงใช้สกอร์บาร์ในการเลื่อนข้อมูลต่าง ๆ แทนการป้อนข้อมูล

หลังจากการแก้ไขข้อมูลแล้ว เพื่อความสะดวกในการทดสอบข้อมูลที่ได้ทำการแก้ไขไปแล้ว ผู้ใช้สามารถทดสอบเสียงได้โดยการกดปุ่มขวาของเมาส์ที่ตำแหน่งของข้อมูล ก็จะเป็นการส่งข้อมูลเฉพาะที่เลือกออกไปที่เอาท์พุทเพื่อกำเนิดเสียง

บทที่ 4

การเขียนโปรแกรมแบบเชิงวัตถุ (Object - Oriented)

พื้นฐานของออบเจกต์ที่สำคัญ ๆ ประกอบด้วย

- แอบสแตรก ดาต้า ไทป์ (abstract data type : ADT) เป็นการรวมไทป์ และชุดของการทำงานเข้าไว้ด้วยกัน ซึ่งการทำงานจะเป็นตัวกำหนดคุณสมบัติของแต่ละไทป์

- การนิยามคลาส (class definition) ซึ่งจะกำหนดการทำงานของ ADT สามารถทำได้ด้วยการนิยามถึงวิธีการเรียกใช้การทำงานของไทป์ การนิยาม คลาส ก็เป็นการกำหนดโครงสร้างของข้อมูลของไทป์

การทำงานของไทป์สามารถแบ่งเป็น 2 แบบคือ การทำงานแบบ พับลิค (public) และแบบไพรเวท (private) การทำงานแบบพับลิค สามารถเรียกใช้ได้ภายนอกคลาสได้ ส่วนการทำงานแบบไพรเวท จะเรียกใช้ได้เฉพาะภายในคลาส การทำงานทั้ง 2 แบบนี้รวมเรียกว่า เมทอด (method) ซึ่งก็คือฟังก์ชั่นในภาษา ออบเจกต์ โอเรียนต

ออบเจกต์ คือ ตัวแปรของคลาส โดยทำการรวมข้อมูลทุกตัวไว้ในออบเจกต์ นั่นคือการรวมข้อมูลทั้ง ไพรเวท และ พับลิค ที่ได้นิยามไว้ในคลาส วิธีเรียกใช้ เมทอด เพื่อใช้งานกับออบเจกต์ ก็สามารถทำได้โดยส่ง เมสเสจ (message)ให้ออบเจกต์ เมสเสจจะประกอบด้วยพารามิเตอร์ ต่างๆ เหมือนกับพารามิเตอร์ของฟังก์ชั่นในภาษาอื่นๆที่ไม่ใช่ภาษาออบเจกต์ โอเรียนต โดยทั่วไปแล้วเมื่อมีการเรียกใช้ เมทอด ก็จะทำให้ข้อมูลภายในออบเจกต์ เปลี่ยนแปลงไป

เอนแคปซูเรชั่น(Encapsulation) เป็นคุณสมบัติของออบเจกต์ ซึ่งเป็นการกำหนดขอบเขตให้กับออบเจกต์ กำหนดส่วนอินเตอร์เฟสให้กับออบเจกต์เพื่อการใช้งาน

การเขียนโปรแกรมแบบเชิงวัตถุ จะเน้นในเรื่อง เอนแคปซูเรชั่น และ ดาต้า ไฮดิ้ง (data hiding) เอนแคปซูเรชั่น และ ดาต้า ไฮดิ้ง จะนำดาต้าและโพรซีเยอร์มารวมกันและจะจำกัดสิทธิในการเข้าถึงดาต้าและโพรซีเยอร์ ดาต้าและโพรซีเยอร์ที่มารวมกันนี้เรียกว่า ออบเจกต์ โดยในแต่ละออบเจกต์จะมีดาต้าและโพรซีเดอร์เป็นของตนเอง ออบเจกต์เท่านั้นที่มีสิทธิในการเข้าถึงดาต้าและโพรซีเดอร์ที่เป็นไพรเวท ผู้ใช้สามารถดำเนินการกับออบเจกต์ได้โดยการส่ง เมสเสจไปให้ออบเจกต์ ดังนั้นโครงสร้างของซอฟแวร์ในแนวออบเจกต์ โอเรียนต จึงไม่ขึ้นอยู่กับดาต้า แต่จะขึ้นอยู่กับเมสเสจต่างๆ ที่สามารถส่งไปให้ออบเจกต์ได้

แนวความคิดของADT ได้กำเนิดขึ้นมาเมื่อ 25 ปีที่แล้ว แต่เพิ่งจะมาแพร่หลายเมื่อไม่กี่ปีมานี่เอง ADT คือโมเดลของซอฟต์แวร์ซึ่งประกอบด้วยเซตของค่าและเซตของตัวดำเนินการกับค่าเหล่านั้น ในภาษาโพรซีเดอร์ เช่นภาษาซีหรือปาสคาลสามารถทำ ADT ได้โดยการกำหนดโกบอลดาต้า และโพรซีเยอร์หรือฟังก์ชั่นที่ดำเนินการกับดาต้านั้น

ภาษา โพรซีเดอร์ รุ่นใหม่เช่น โมดูลา-2 (Modula-2) และ เอด้า(Ada) ได้พัฒนาแนวความคิดนี้ไปบ้าง ภาษาเหล่านี้ยอมให้โปรแกรมเมอร์แยกส่วนอินเตอร์เฟส(definition หรือ interface) กับส่วนอิมพลีเมนต์(implementation)ของ ADT ออกจากกัน ส่วนอินเตอร์เฟสจะกำหนดชื่อของ ADT และโพรซีเดอร์หรือฟังก์ชั่นที่มี และส่วนอิมพลีเมนต์จะเป็นตัวโปรแกรมการทำงานจริงๆ ของโพรซีเดอร์หรือฟังก์ชั่นนั้น โปรแกรมเมอร์ไม่สาสารถเข้าถึงดาต้าภายในได้ แต่โพรซีเดอร์หรือฟังก์ชั่นสามารถทำได้

การแยกส่วนอินเตอร์เฟสกับส่วนอิมพลีเมนต์นับได้ว่าเป็นจุดที่น่าสนใจมาก การนำดาต้าและโพรซีเยอร์หรือฟังก์ชั่นมารวมกันนี้เรียกว่า เอนแคปซูเรชั่น และการที่ไม่สามารถเข้าถึงดาต้าภายในได้เรียกว่า ดาต้า ไฮดิ้ง การใช้แนวความคิดนี้มาใช้จะช่วยลดปัญหาการบำรุงรักษาได้ ถ้าดาต้าภายในออบเจกต์เปลี่ยนไป ส่วนที่ต้องเปลี่ยนแปลงคือส่วนอิมพลีเมนต์ แต่ส่วนอินเตอร์เฟสยังคงเดิม ทำให้ส่วนที่เหลือของระบบไม่ต้องทำการแก้ไข เพราะระบบจะติดต่อกับออบเจกต์โดย ผ่านทางส่วนอินเตอร์เฟส ถ้าส่วนอินเตอร์เฟสไม่เปลี่ยนแปลง ระบบก็ไม่ต้องเปลี่ยนแปลง

- การเขียนโปรแกรมโดยที่มีส่วนอินเตอร์เฟสกับส่วนอิมพลีเมนต์อยู่รวมกัน คอมไพเลอร์จะถือว่า เมทอดนั้นเป็น อินไลน์ (Inline) นั่นคือ คอมไพเลอร์จะแทนการทำงานทั้งหมดของเมทอดลงไปเมื่อมีการเรียกใช้เมทอด โดยไม่มีการเรียกใช้ฟังก์ชั่นจริงๆซึ่งจะทำให้โปรแกรมทำงานได้เร็วขึ้น แต่ข้อเสียคือโปรแกรมจะมีขนาดใหญ่ขึ้น

- การเขียนโปรแกรมโดยแยกส่วนอินเตอร์เฟสกับส่วนอิมพลีเมนต์ออกเป็นคนละไฟล์ ก็ยังคงทำงานได้เหมือนเดิม

การใช้งาน Visual C++

ปัจจุบันการใช้งานวินโดว์เป็นไปอย่างกว้างขวาง เนื่องจากความสะดวกสบายและการเรียนรู้การใช้งานแอพพลิเคชันต่างๆเป็นไปได้โดยง่าย และพร้อมกับเครื่องมือและทูลต่างๆ ที่มีการพัฒนากันออกมาเพื่ออำนวยความสะดวกในการพัฒนาแอพพลิเคชันบนวินโดว์

จุดเด่นที่สำคัญของ วิชวล ซี ++ คือ การทำงานของ คลาส วิซาร์ด(Class Wizard) คลาส วิซาร์ด มีหน้าที่เขียนโปรแกรมการทำงานบางส่วนให้แก่แอพพลิเคชัน เช่นถ้าเราต้องการวางปุ่มกด (push button)ลงบนแอพพลิเคชัน จากนั้นเราต้องการเขียนโปรแกรมเพื่อควบคุมการทำงานของปุ่มนั้น ก็ทำได้โดยการเลือกคลาส วิซาร์ด และกำหนดการทำงานที่ต้องการ จากนั้นคลาสวิซาร์ดก็จะ เขียนโปรแกรมเพื่อตอบสนองการทำงานในส่วนนั้นให้ ขั้นตอนสุดท้ายเราจึงเขียนโปรแกรมควบคุมเอง

ดังนั้นก่อนที่จะเริ่มสร้างแอพพลิชั่นควรจะต้องรู้ถึงคลาสที่สำคัญๆก่อน

สถาปัตยกรรมของคลาสต่างๆในแอพพลิเคชัน (Application Architecture Classes)

Windows Application Class

- CWinApp เป็นคลาสที่เก็บคำสั่งการทำงานต่างๆที่ใช้ เป็นตัวเริ่มต้น, การทำงาน, จนถึงตอนจบการทำงานของแอพพลิเคชัน

Command-Related Classes

- CCmdTarget ใช้เป็นคลาสพื้นฐาน(base class) ของ คลาสของออบเจกต์ทั้งหมด ซึ่งสามารถรับและตอบสนองต่อเมสเสจได้

- CCmdUI ให้โปรแกรมอินเตอร์เฟสที่ใช้สำหรับทำการอัปเดทในส่วนของออบเจกต์ที่ทำการติดต่อกับผู้ใช้(user-interface objects) เช่น เมนู ไอเท็ม (menu items) หรือ ปุ่มการทำงานต่างๆ หรือเมื่อมีคำสั่งการทำงานต่อออบเจกต์ที่ต้องการเช่น การกำหนดให้ยอมรับคำสั่งการทำงาน,กำหนดให้ไม่ยอมรับคำสั่งในการทำงาน ออบเจกต์ที่ทำการติดต่อกับผู้ใช้ก็จะทำหน้าที่ส่งผ่านต่อไป

Document/View Classes

- CDocTemplate เป็นคลาสพื้นฐานของรูปแบบเอกสาร(document templates) รูปแบบเอกสารจะใช้เป็นตัวประสานในการสร้างออบเจกต์ต่อไปนี้ ดอกคิวเมนต์ (document), วิว (view), และ เฟรมวินโดว์

- CSingleDocTemplate เป็นรูปแบบสำหรับเอกสาร ที่ใช้ในการติดต่อกับเอกสารเดียว (single document interface), (SDI) SDI คือในการทำงานจะเปิดเอกสารเพื่อใช้งานได้เพียงเอกสารเดียว

- CMultiDocTemplate เป็นรูปแบบสำหรับเอกสาร ที่ใช้ในการติดต่อกับหลายๆ เอกสาร (multiple document interface), (MDI). MDI คือในการทำงานจะเปิดเอกสารเพื่อใช้งานได้หลายๆเอกสารพร้อมๆกัน

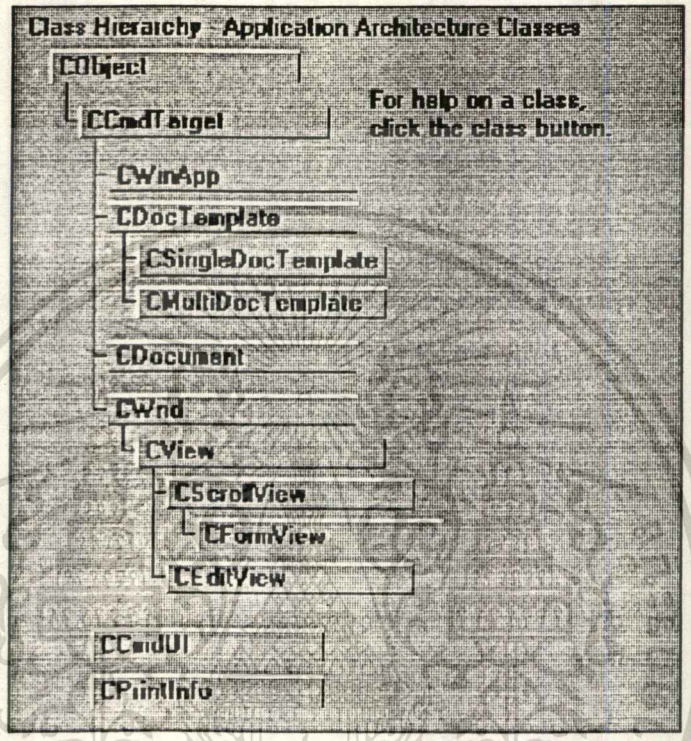
- CDocument เป็นคลาสสำหรับใช้ทำเอกสาร โดยไฟล์เอกสารของผู้ใช้จะได้รับการถ่ายทอดมาจากคลาสนี้

- Cview เป็นคลาสพื้นฐานที่ใช้ในการมองถึงข้อมูลในเอกสาร โดยวิวคลาสของของผู้ใช้จะได้รับการถ่ายทอดมาจากคลาสนี้

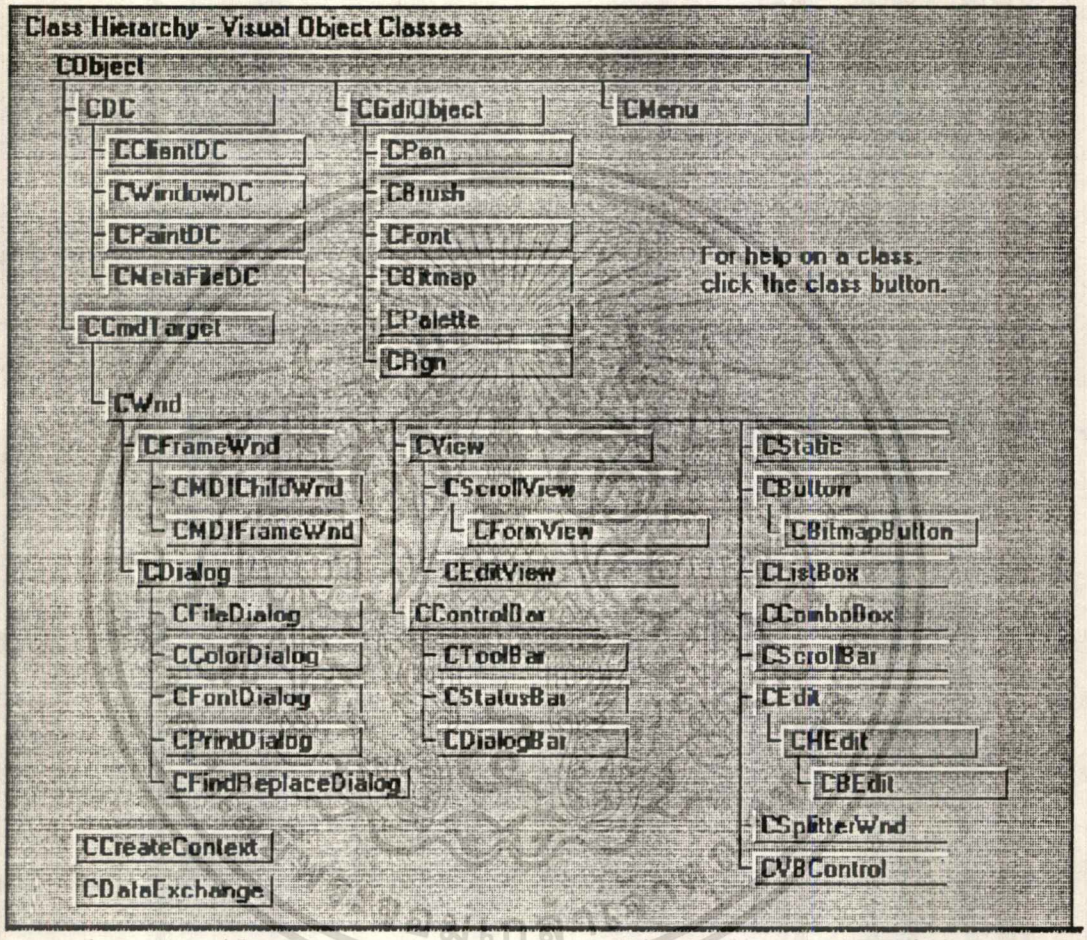
- CPrintInfo เป็นโครงสร้างที่มีการเก็บเนื้อหาเกี่ยวกับการพิมพ์ หรือการพรีวิว(preview)

- CCreateContext เป็นโครงสร้างที่รูปแบบเอกสารจะถูกส่งผ่านไปยังวินโดว์ เพื่อทำการ สร้างฟังก์ชั่นที่จะต้องประสานกับการสร้าง เอกสาร, วิว, และเฟรม วินโดว์ ออบเจกต์

Application Architecture Hierarchy



Visual Object Hierarchy



การเขียนโปรแกรมด้วย วิชวล ซี ++ แบ่งเป็น 2 ขั้นตอน

1. ขั้นตอนการออกแบบแอพพลิเคชัน (Visual programming step) เป็นการทำงานโดยอาศัย ซอฟแวร์ ทูลซึ่งอยู่ในวิชวล แพคเกจ ทูลต่างๆที่กล่าวถึง สามารถนำมาใช้ในการออกแบบแอพพลิเคชัน โดยอาศัยเมาส์ และคีย์บอร์ด ซึ่งในขั้นตอนนี้ยังไม่มีการเขียนโปรแกรมการทำงานใดๆทั้งสิ้น

2. ขั้นตอนการเขียนโปรแกรมเพื่อควบคุมการทำงานภายใน ซึ่งหลังจากที่ซอฟแวร์ ทูลซึ่งอยู่ในวิชวล แพคเกจ ได้สร้างคลาสต่างๆให้แล้ว โปรแกรมเมอร์ก็จะต้องเขียนโปรแกรมเพื่อตอบรับเมสเสจต่างๆ โดยอาศัยคลาสต่างๆที่มีมาให้แล้ว

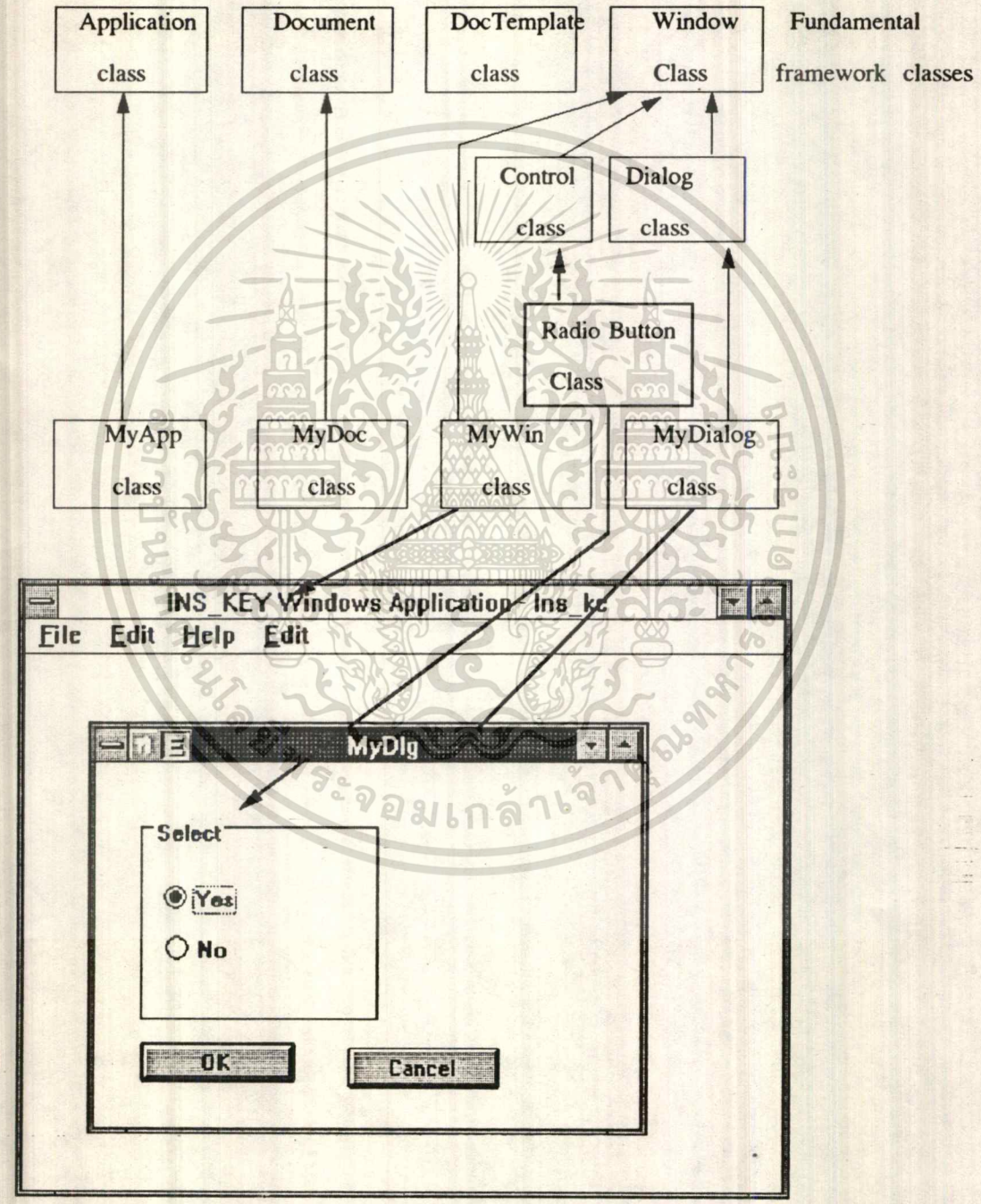
ในการพัฒนาแอพพลิเคชันบนวินโดว์ สิ่งที่สำคัญคือการใช้คลาสไลบรารีต่างๆให้ถูกต้องและเหมาะสม แอพพลิเคชัน เฟรมเวอรก์ (Application Framework) ก็เป็นคลาสไลบรารีหนึ่งที่จะช่วยอำนวยความสะดวกให้กับผู้ใช้ในการที่จะพัฒนาแอพพลิเคชันบนวินโดว์ โดยที่แอพพลิเคชัน เฟรมเวอรก์ จะช่วยจัดการในเรื่องของ เมสเสจลูป และส่วนประกอบต่างๆ ที่เป็นโครงร่างและทำหน้าที่ดำเนินงานกับ วินโดว์

แอพพลิเคชัน เฟรมเวอรก์คลาสไลบรารีที่มีคล้ายคลึงกับ วินโดว์ API แต่วินโดว์ API จะมีลักษณะแบบฟังก์ชั่นหรือโพรซีเดอร์ แต่สำหรับ แอพพลิเคชัน เฟรมเวอรก์ นั้นจะมีลักษณะการเขียนโปรแกรมในแบบของเชิงวัตถุ

ในส่วนของ MFC ได้มีการใช้งาน เมสเสจแมพ (message maps) ที่ช่วยในการจัดการเมสเสจภายในวินโดว์ ที่ฟังก์ชั่นที่เป็นสมาชิกของคลาสได้ออกแบบไว้ต้องการตอบสนองต่อ

สร้างแอพพลิเคชันโดยใช้ แอพพลิเคชัน เฟรมเวอร์ก์(Application Framework)

Application Framework Architecture



รูปแสดงโครงสร้างของแอพพลิเคชันที่สร้างโดย MFC Application Framework

จากรูปได้แสดงถึงโครงสร้างของ วินโดว์แอพพลิเคชันที่สร้างขึ้นโดยใช้ แอพพลิเคชัน เฟรมเวอรก์ ซึ่งจากรูปจะเห็นว่าส่วนของคลาสพื้นฐานของไลบรารีนั้นจะประกอบไปด้วย Application Class, Document Class, DocTemplate Class, Window Class

- Application Class เป็น class ที่ทำหน้าที่ในส่วนของการรับส่ง และดำเนินการกับเมสเสจต่างๆ คล้ายกับการทำงานในส่วนของ WinMain

- จากนั้นเราก็ต้องทำการสร้าง object โดยใช้ class ที่ได้จาก แอพพลิเคชัน เฟรมเวอร์ก

- วินโดว์คลาส ซึ่งเป็นส่วนที่ทำหน้าที่ในการกำหดการกระทำของหน้าต่างแต่ละหน้าต่าง โดยที่ วินโดว์สามารถที่จะทำงานบนหน้าต่างเหล่านั้นได้ซึ่งMFC จะสร้างวินโดว์คลาส ต่างๆจากคลาสที่มีรายละเอียดต่างๆซึ่งเก็บอยู่ใน ไลบรารี เช่น คอนโทรล คลาส , กรอบข้อความ คลาส

- แอพพลิเคชัน เฟรมเวอรก์ ได้เตรียม ไดอะล็อกคลาสไว้ให้กับผู้ใช้เพื่อสะดวกในการเรียกใช้ได้เลย โดยที่ ไดอะล็อกคลาส นี้จะได้มาจากวินโดว์คลาส เพราะว่าไดอะล็อกบ็อกซ์เปรียบเสมือนหน้าต่างหน้าต่างหนึ่งบนวินโดว์ แต่มีลักษณะการทำงานที่พิเศษต่างจากหน้าต่างอื่นๆเท่านั้น ในการใช้งานเกี่ยวกับไดอะล็อกบ็อกซ์ ก็ทำได้โดยรับการสืบทอดจาก วินโดว์คลาส ไปสู่ ไดอะล็อก คลาส ของผู้ใช้ แล้วกำหนดค่าในส่วนของการควบคุมและการตอบสนองการทำงานต่างๆ

- ขั้นสุดท้ายเป็นการตรวจสอบค่าที่จะหยุดการทำงานของไดอะล็อกบ็อกซ์

สรุปจะเห็นว่าในการสร้างแอพพลิเคชันโดย แอพพลิเคชัน เฟรมเวอรก์ นี้ วิชวล ซี++ ได้เตรียมทูลไว้แล้ว โดยในขั้นตอนการสร้างโครงสร้างและเซตของของคลาสที่ต้องการใช้งานในแอพพลิเคชันนี้ก็สามารถใช้ AppWizard และจากนั้นถ้าต้องการเพิ่มเติม, ลด คลาส หรือ ฟังก์ชั่นในคลาสก็สามารถใช้ ClassWizard เพื่อช่วยในการอ้างอิงได้

ตัวอย่างการทำงานของ แอพพลิเคชัน เฟรมเวอร์ก

เมื่อทดลองสร้างแอพพลิเคชันขึ้นมาโดยอาศัย AppWizard วิชวล ซี++ จะทำการสร้างคลาสขึ้นมา 5 คลาส ซึ่งจะแยกระหว่างส่วน อินเตอร์เฟส กับ ส่วนอิมพลีเมนต์ โดยจะถูกเก็บไว้ในไฟล์นามสกุล .H และ .CPP ซึ่งรวมทั้งหมด 10 ไฟล์ และยังมีไฟล์ที่เก็บรีซอร์ซที่ใช้ในแอพพลิเคชัน ซึ่งเก็บอยู่ในไฟล์นามสกุล .RC และสำหรับไฟล์ .DEF ก็จะทำหน้าที่เก็บไฟล์ที่กำหนดลักษณะของแอพพลิเคชัน

คลาสที่ถูกสร้างจากทูลประกอบด้วย

1. CTestApp : Test.h

   Test.cpp

2. CTestDoc : TestDoc.h

   TestDoc.cpp

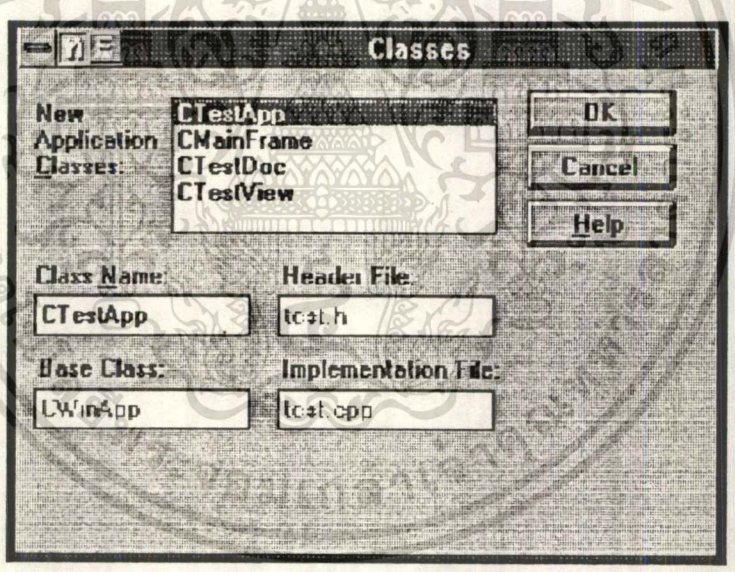
3. CTestView : TestView.h

   TestView.cpp

4. CMainFrame :mainfrm.h

   mainfrm.cpp

5. CAbout



การส่งเมสเสจในแอพพลิเคชันเฟรมเวอร์ก (Message maps of Application Framework)

การทำงานที่สำคัญอีกส่วนหนึ่งของ แอพพลิเคชัน เฟรมเวอร์ก ก็คือส่วนของ แผนผังการส่งเมสเสจ ซึ่งมนส่วนนี้จะทำหน้าที่คอยให้ความช่วยเหลือแอพพลิเคชันในการรับส่งเมสเสจต่างๆ ภายในระบบของวินโดว์ ซึ่งสำหรับจุดประสงค์หลักของ แผนผังการส่งเมสเสจ ก็คือการ

จัดการกับฟังก์ชันให้ได้รับเมสเสจตามที่ต้องการ ซึ่งในการทำงานของแผนผังการส่งเมสเสจ ก็จะดำเนินการสองสิ่งคือ

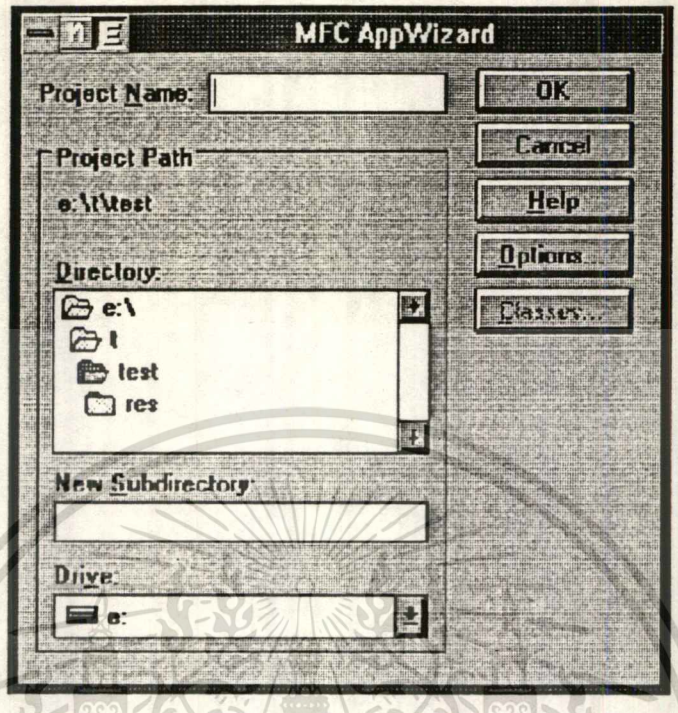
- ทำการเปลี่ยนชุดของเมสเสจของวินโดว์ ให้เป็นพารามิเตอร์ wParam และ lParam สำหรับฟังก์ชันที่เป็นสมาชิกของคลาสนั้นๆ
- ค้นหาเส้นทางของเมสเสจที่ถูกต้องในการส่งเข้าสู่แต่ละระดับชั้นของคลาส

เมื่อเมสเสจของวินโดว์ ถูกส่งมาถึง แอพพลิเคชัน เฟรมเวอร์ก จะทำการค้นหาฟังก์ชันที่ทำหน้าที่ดำเนินการกับเมสเสจในแผนผังการส่งเมสเสจของคลาส ณ ที่คลาสระดับต่ำสุดที่มีการสืบมา และเฟรมเวอร์กก็จะส่งเมสเสจที่ได้รับนี้ให้กับแผนผังการส่งเมสเสจ เพื่อค้นหาฟังก์ชันการทำงานที่เป็นสมาชิกของคลาสที่ดำเนินการกับเมสเสจนี้

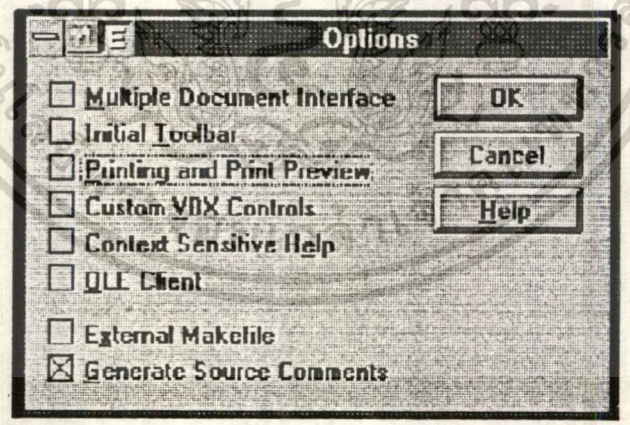
เมื่อพบแล้ว เฟรมเวอร์ก ก็จะทำการแยกเอา wParam และ lParam เพื่อส่งค่าพารามิเตอร์ทั้งสองให้กับฟังก์ชันที่ดำเนินการกับเมสเสจที่ได้รับ แต่ถ้าหากว่าค้นหาไม่พบแล้วแผนผังการส่งเมสเสจก็จะทำการค้นหาขึ้นไปในระบบถัดไปของคลาสนั้นไปจนถึงคลาสพื้นฐานของคลาสนั้นๆ

ขั้นตอนการสร้างแอพพลิเคชัน

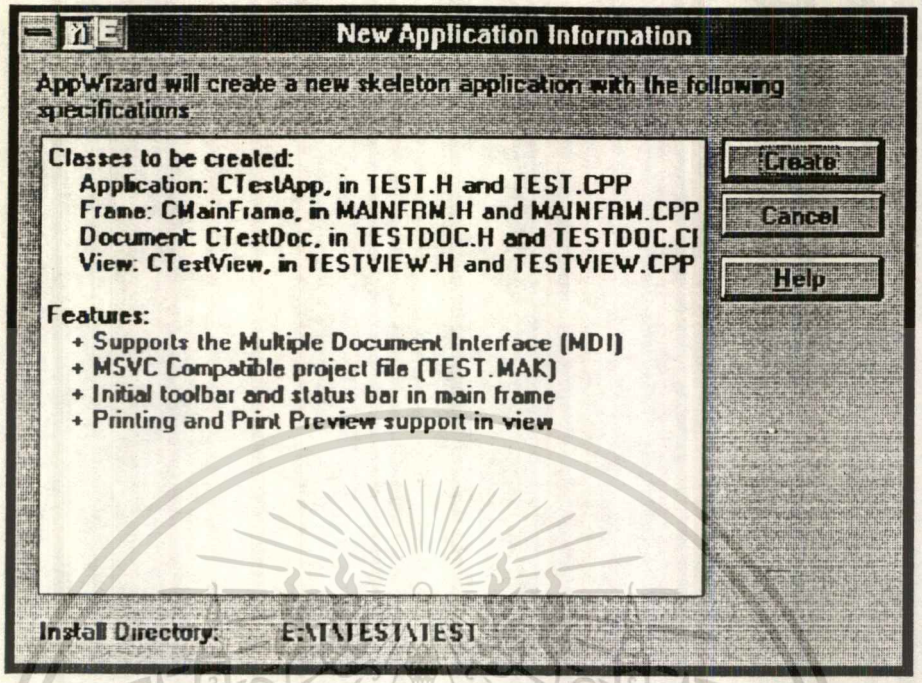
1. การสร้างแอพพลิเคชันโปรเจก
   - เลือกเมนูโปรเจก ---> AppWizard
   - กำหนดชื่อโปรเจก



- เลือกออปชัน ---> เลือกเฉพาะ Generate Source Comment



ขณะนี้การสร้างรูปแบบของแอพพลิเคชันเสร็จแล้ว ถ้ารูปแบบตรงตามความต้องการแล้ว ก็ทำการเลือก ปุ่ม OK ตามด้วย Create แล้วเลือกเมนูโปรเจก ---> Rebuilld all เพื่อ compile



2. การออกแบบระบบเมนูของแอพพลิเคชัน

- เลือกเมนูทูล ---> AppStudio ---> เปิด file .RC
- ถ้าต้องการออกแบบ ระบบเมนูให้เลือกไทพ (จากระบบเมนู ไอเท็ม)
- กดดับเบิลคลิกที่ IDR-MAINFRAME item

ขณะนี้สามารถทำการเพิ่ม, ลดระบบเมนูได้โดยใช้ปุ่ม insert, delete

3. การออกแบบวินโดว์แบบกรอบข้อความ (Dialog Box)

- เลือกเมนูทูล ---> AppStudio ---> Resource ---> Dialog ---> New

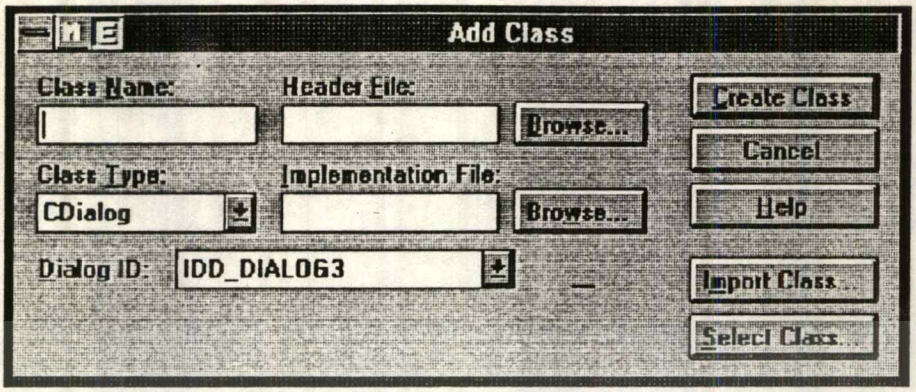
การกำหนดคลาสให้กับ Dialog Box

- เลือกเมนูทูล ---> AppStudio ---> Resource ---> ClassWizard

กำหนดชื่อคลาส :

กำหนด(ส่วน implement)ชื่อไฟล์ .CPP:

กำหนด(ส่วน interface)ชื่อไฟล์ .H:



4. ออกแบบส่วนควบคุม(คอนโทรล ต่างๆ เช่น อีดิต บอกซ์, ปุ่มกด)

5. เขียนโปรแกรมการทำงานของ แต่ละ เมนู ไอเท็ม
(เช่น เมื่อ เลือกเมนูไอเท็ม ก็จะไปทำงานที่วินโดว์อื่น
การทำให้ออบเจกของหน้าต่างแบบกรอบข้อความเป็นข้อมูลที่เป็นสมาชิก(data member) ของ วิว คลาส ทำได้โดยประกาศ ออบเจกของกรอบข้อวามไว้ที่ view.h ก็จะทำให้ทุกๆ ฟังก์ชันของ วิว สามารถอ้างถึงออบเจกนั้นได้)

โปรแกรมที่ทำหน้าที่ส่งข้อมูล, แก้ไขข้อมูล ประกอบด้วย วินโดว์แบบกรอบข้อความ 2 วินโดว์ ใหญ่ๆ 1. Trak Edit 2. Step Edit
ขั้นตอนการทำงานทั้งหมด

- สร้าง วินโดว์แบบกรอบข้อวาม ชื่อ "แทรคอีดิต"(Track Edit)

กำหนดตัวควบคุมต่างๆ ที่ต้องการไว้ในวินโดว์ ประกอบด้วย ปุ่มต่างๆ และเมื่อวินโดว์นี้ทำงานจะตรวจสอบว่ามีการโหลดข้อมูล .MID เข้ามาหรือไม่ ถ้าเมนูไอเท็ม ที่วิวคลาสได้รับเมสเสจในการอ่านข้อมูล จะส่งผลให้วินโดว์นี้ทำงาน คือ วินโดว์นี้จะตรวจสอบว่าในโครงสร้างข้อมูลมีข้อมูลโน๊ตในห้องเพลงที่ต้องการวาดหรือไม่ ถ้ามีก็จะวาดบิตแมปที่มีสี ถ้าไม่มีก็ทำการวาดบิตแมปว่างเปล่า

วินโดว์ก็จะรอรับเมสเสจต่างๆ เช่น ปุ่ม หยุดเพลงชั่วคราว, หยุดเพลงถาวร เล่นเพลง, ถอยหลัง, เดินหน้า, เล่นเพลง, อัดเพลง

เมสเสจที่วินโดว์รอรับอีกคือ การกดปุ่ม ซ้ายของเมาส์ดับเบิลคลิ๊ก วินโดว์นี้ก็จะรับเมสเสจพร้อมกับ พิกัดที่เมาส์ชี้อยู่ จากนั้นฟังก์ชันการทำงานของวินโดว์นี้จะทำการตรวจสอบว่าพิกัดที่ผู้ใช้เลือกอยู่ในขอบเขตของวินโดว์ ก็ทำการเรียกใช้ วินโดว์ สเต็บ อีดิต

- สร้าง วินโดว์แบบกรอบข้อวาม ชื่อ "สเต็บ อีดิต"(Step Edit)

กำหนดตัวควบคุมต่างๆ ที่ต้องการไว้ในวินโดว์ ประกอบด้วย ปุ่มต่างๆ วินโดว์นี้จะตรวจสอบว่าในโครงสร้างข้อมูลมีข้อมูลโน๊ตในห้องเพลงที่ต้องการวาดหรือไม่ ถ้ามีก็จะวาดบิตแมปที่มีสี

วินโดว์ก็จะรอรับเมสเสจต่างๆ เช่น ปุ่ม เพิ่มเติมข้อมูลโน๊ต, ลบข้อมูลโน๊ต

วินโดว์รอรับเมสเสจอีกคือ การกดปุ่ม ซ้ายของเมาส์ดับเบิลคลิ๊ก วินโดว์นี้ก็จะรับเมสเสจพร้อมกับ พิกัดที่เมาส์ชี้อยู่ จากนั้นฟังก์ชันการทำงานของวินโดว์นี้จะทำการตรวจสอบว่า พิกัดที่ผู้ใช้เลือกมีข้อมูลโน๊ตหรือไม่ ถ้ามีก็ทำการเรียกใช้ วินโดว์แบบกรอบข้อความขนาดเล็ก กรอบข้อความขนาดเล็กก็จะนำข้อมูลต่างๆที่เกี่ยวกับโน๊ตมาแสดง และผู้ใช้สามารถที่จะแก้ไขข้อมูลเหล่านั้นได้ โดยการรับข้อมูลอินพุทจากผู้ใช้จะไม่รับคีย์ แต่จะรับการป้อนข้อมูลผ่านเมาส์ โดยผู้ใช้ต้องเลือกทางตัวเลื่อน

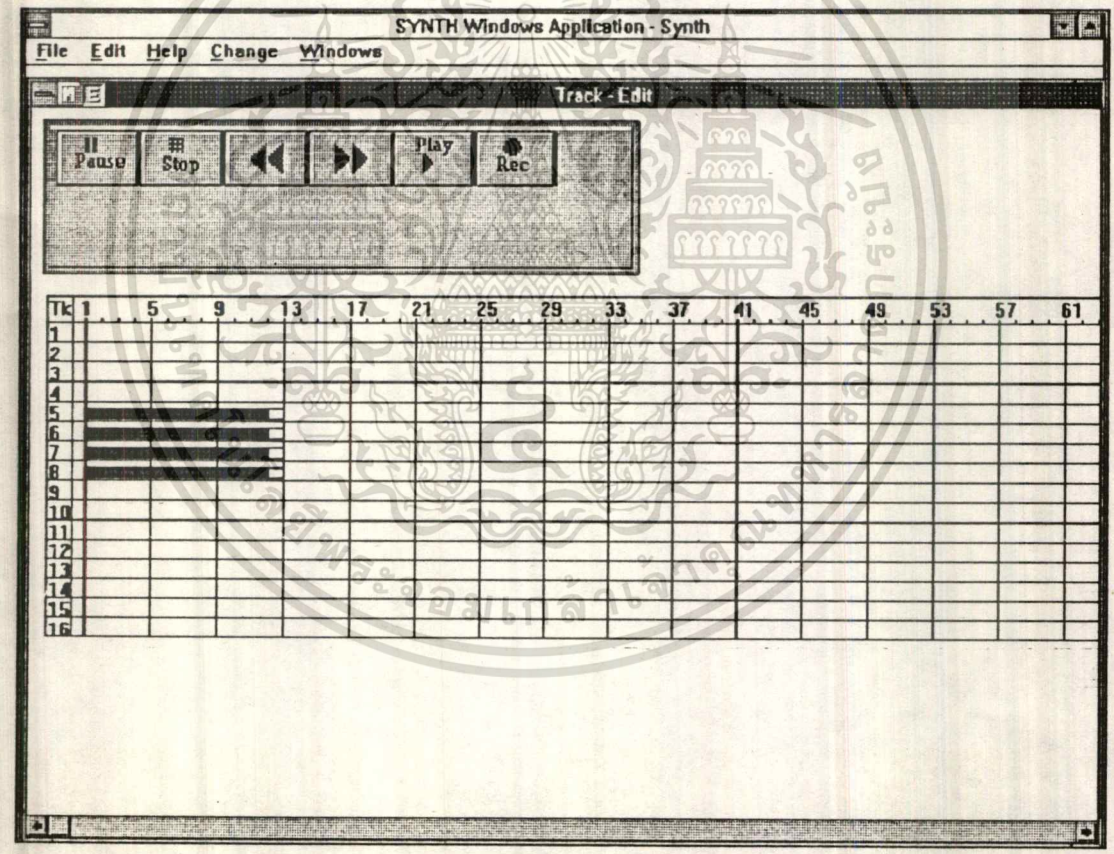
วินโดว์รอรับเมสเสจอีกคือ การกดปุ่ม ขวาของเมาส์ วินโดว์นี้ก็จะรับเมสเสจพร้อมกับพิกัดที่เมาส์ชี้อยู่ จากนั้นฟังก์ชันการทำงานของวินโดว์นี้จะทำการตรวจสอบว่า พิกัดที่ผู้ใช้เลือกมีข้อมูลโน๊ตหรือไม่

บทที่ 5

ผลการทดลองการใช้งาน

หน้าต่างแทร็กอีดิต (Track edit window)

หน้าต่างแทร็กอีดิตนี้จะทำหน้าที่แสดงข้อมูลของแต่ละแทร็กว่าห้องเพลงใด (measure) มีข้อมูลอยู่ โดยถ้าห้องใดเป็นช่องทึบก็แสดงว่าห้องนั้นมีข้อมูลอยู่ แต่ถ้าห้องใดมีลักษณะเป็นกรอบว่างๆ ก็แสดงว่าห้องนั้นไม่มีข้อมูลอยู่ดังรูป



รูปแสดงหน้าต่างแทร็คอีดิต

จากรูปจะเห็นว่ามีปุ่มอยู่ 6 ปุ่ม ดังนี้

- ปุ่ม 

ใช้สำหรับหยุดการเล่นเพลงชั่วคราว

- ปุ่ม 

ใช้สำหรับหยุดการเล่นเพลง
- ปุ่ม  ใช้สำหรับย้อนกลับไปเล่นที่ต้นเพลง (Rewind)
- ปุ่ม  ใช้สำหรับเลื่อนเพลงไปทีละ 1 ห้องเพลง (Fast Forward)
- ปุ่ม 

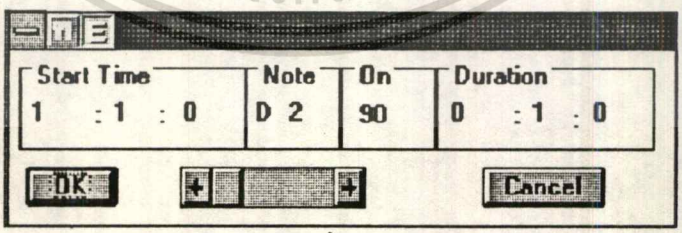
ใช้สำหรับเล่นเพลง
- ปุ่ม 

ใช้สำหรับบันทึกข้อมูลมีดี้ที่ส่งมาทางพอร์ตมีดี้อิน

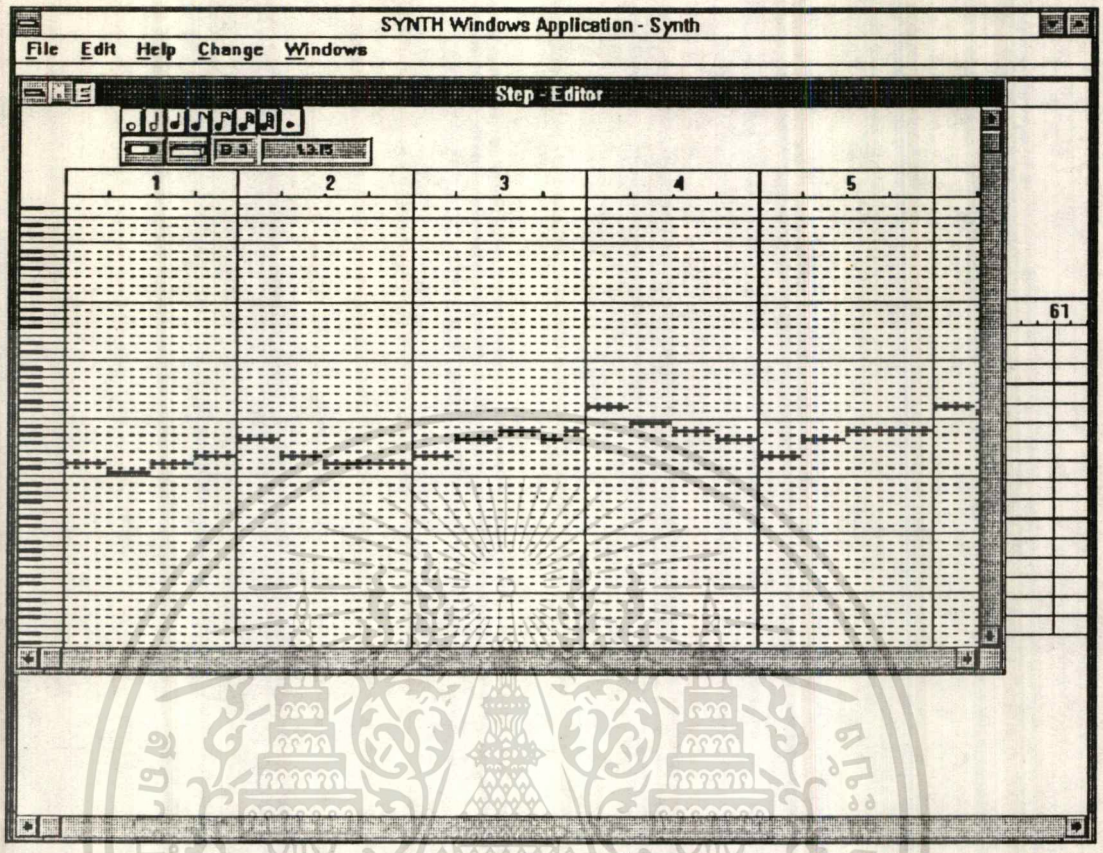
หน้าต่างสเต็ปอีดิต (Step edit window)

หน้าต่างสเต็ปอีดิตจะทำหน้าที่แสดงแถบเส้นของตัวโน้ตออกที่จอภาพ โดยสามารถที่จะทำการแก้ไขข้อมูลมูลตัวโน้ตได้ โดยจะมีปุ่มรูปดินสอซึ่งจะใช้ในการใส่ตัวโน้ตเพิ่มเข้าไป และปุ่มรูปยางลบจะใช้สำหรับลบตัวโน้ตออก ส่วนปุ่มรูปตัวโน้ตจะเป็นการเลือกความยาวของตัวโน้ตที่จะทำการใส่เพิ่มเข้าไป นอกจากนี้ที่หน้าต่างสเต็ปอีดิตก็จะมีการแสดงค่าของเวลาตรงตำแหน่งที่เคอร์เซอร์ปรากฏอยู่ เพื่อที่เวลาใส่ตัวโน้ตจะทำให้ใส่ได้ถูกตำแหน่ง

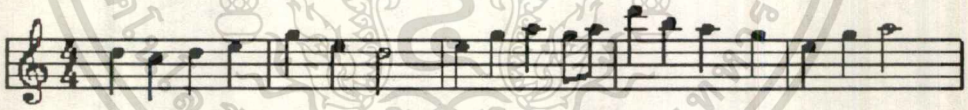
สังเกตุจากหน้าต่างสเต็ปอีดิตในรูป ทางซ้ายมือสุดจะมีรูปคีย์บอร์ด ซึ่งมีลักษณะเหมือนกับคีย์ของเปียนโนทำหน้าที่ในการแสดงลำดับเสียงของตัวโน้ต โดยจะมีตัวโน้ตปรากฏอยู่เป็นเส้นสีเทา และเมื่อดับเบิ้ลคลิกที่แถบตัวโน้ตก็จะแสดงไดอะล้อกขึ้นมาให้สามารถแก้ไขข้อมูลต่างๆ ของตัวโน้ตได้ ดังรูป



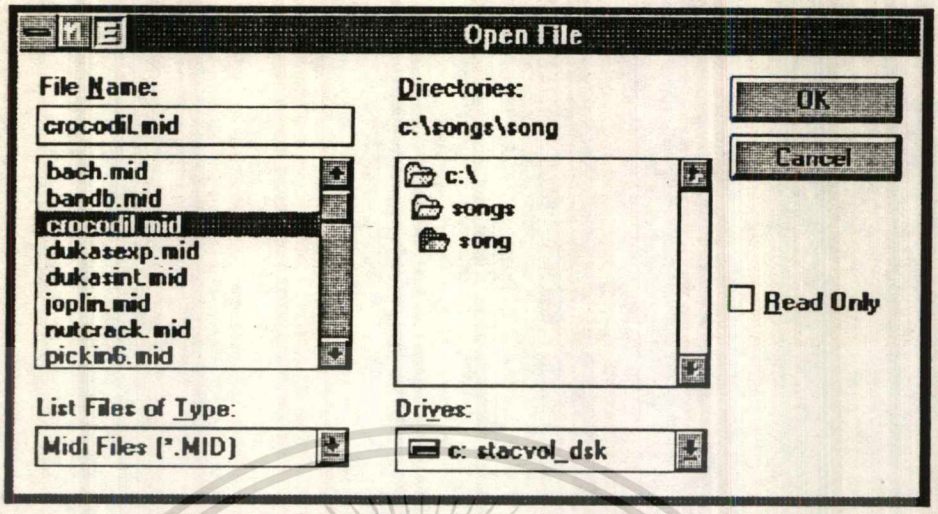
รูปแสดงไดอะล้อกที่ใช้ในการอีดิตตัวโน้ต



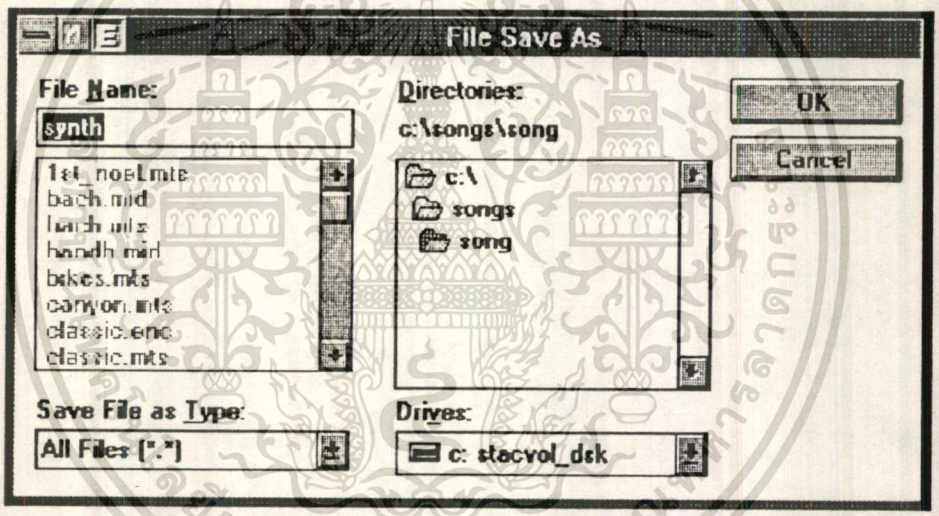
รูปแสดงหน้าต่างสเต็ปอีดิต



รูปแสดงเปรียบเทียบตัวโน้ตที่ได้จากรูปบน



รูปแสดงไดอะล้อกสำหรับเปิดไฟล์



รูปแสดงไดอะล้อกสำหรับเซฟไฟล์

บทที่ 6

บทวิจารณ์และสรุป

1. การออกแบบ ออบเจก จะค่อนข้างยาก

- จะใช้การออกแบบ ออบเจก โอเรียนเตด ผสมกับ การทำงานแบบฟังก์ชัน
- โดยส่วนที่ต้องมีการ อ่าน, เขียน จะทำงานแบบ เป็น โกบอล
- ส่วนที่ต้องแสดงผล, และรอรับ เมสเสจ จากผู้ใช้จะทำงานตามลักษณะ ซี++

2. วิชวลซี พลัส พลัส เป็นภาษาใหม่ที่ต้องการใช้เวลาในการเรียนรู้

-ข้อจำกัดต่าง ๆ (รวมถึงปัญหา) ของ วิชวล ซี พลัส พลัส เวอร์ชัน 1

- การเขียนโปรแกรม เพื่อเรียกใช้ ฟังก์ชัน ไดนามิก ลิงค์ ไลบรารี (Dynamic Link Library) จะต้องเลือกรูปแบบของหน่วยความจำ ให้เป็นหน่วยความจำ แบบ large

- การอ่านข้อมูล, การจองหน่วยความจำ ชนิด fmalloc() จะต้องเลือกโหมดการคอมไพล์เป็นโหมด ดีบัก

ส่งผลให้ ขนาดของโปรแกรมมีขนาดใหญ่กว่าการคอมไพล์ในโหมดรีลีส

ส่งผลให้ การคอมไพล์ ก็ใช้เวลานานกว่าการคอมไพล์ในโหมดรีลีส

- การตรวจสอบการจบประโยคของโปรแกรม

ตัวอย่างโปรแกรมข้างล่างนี้มีการเขียนที่ผิดรูปแบบของภาษาซี แต่ คอมไพล์เลอร์ตรวจสอบไม่ได้

```
void Cadlg::OnOK()
{
    // TODO: Add extra validation here
```

```
        int x,y;
        x=1,
        y=9,
        y = x,
        MessageBox("TEST"),
        CDialog::OnOK();
}
```

- ส่วนที่ไม่ใช่ข้อมูลของโปรแกรม (comment) โปรแกรมเมอร์ไม่ควรไปยุ่งเกี่ยว เพราะอาจมีผลกระทบต่อ การอ้างถึง คลาสวิซาร์ด

ตัวอย่างของรูปแบบที่ถูกต้อง

```
class Cadlg : public CDialog
{
// Construction
public:
        Cadlg(CWnd* pParent = NULL);        // standard constructor
    BOOL ml_state;
// Dialog Data
        //{{AFX_DATA(Cadlg)
        enum { IDD = IDD_DIALOG2 };
                // NOTE: the ClassWizard will add data members here
        //}}AFX_MSG
        . . . . . .
        . . . . . .
}
```

ตัวอย่างของรูปแบบที่ไม่ถูกต้อง

```
class Cadlg : public CDialog
{
```

```
// Construction
public:
        Cadlg(CWnd* pParent = NULL);        // standard constructor
    BOOL ml_state;
// Dialog Data
        //{{AFX_DATA(Cadlg)
        enum { IDD = IDD_DIALOG2 };
                // NOTE: the ClassWizard will add data members here
```

** ถ้ามีการลบข้อมูลของบรรทัดถัดไปนี้ จะส่งผลกระทบต่อคลาส แต่การคอมไพล์จะไม่มีปัญหา **

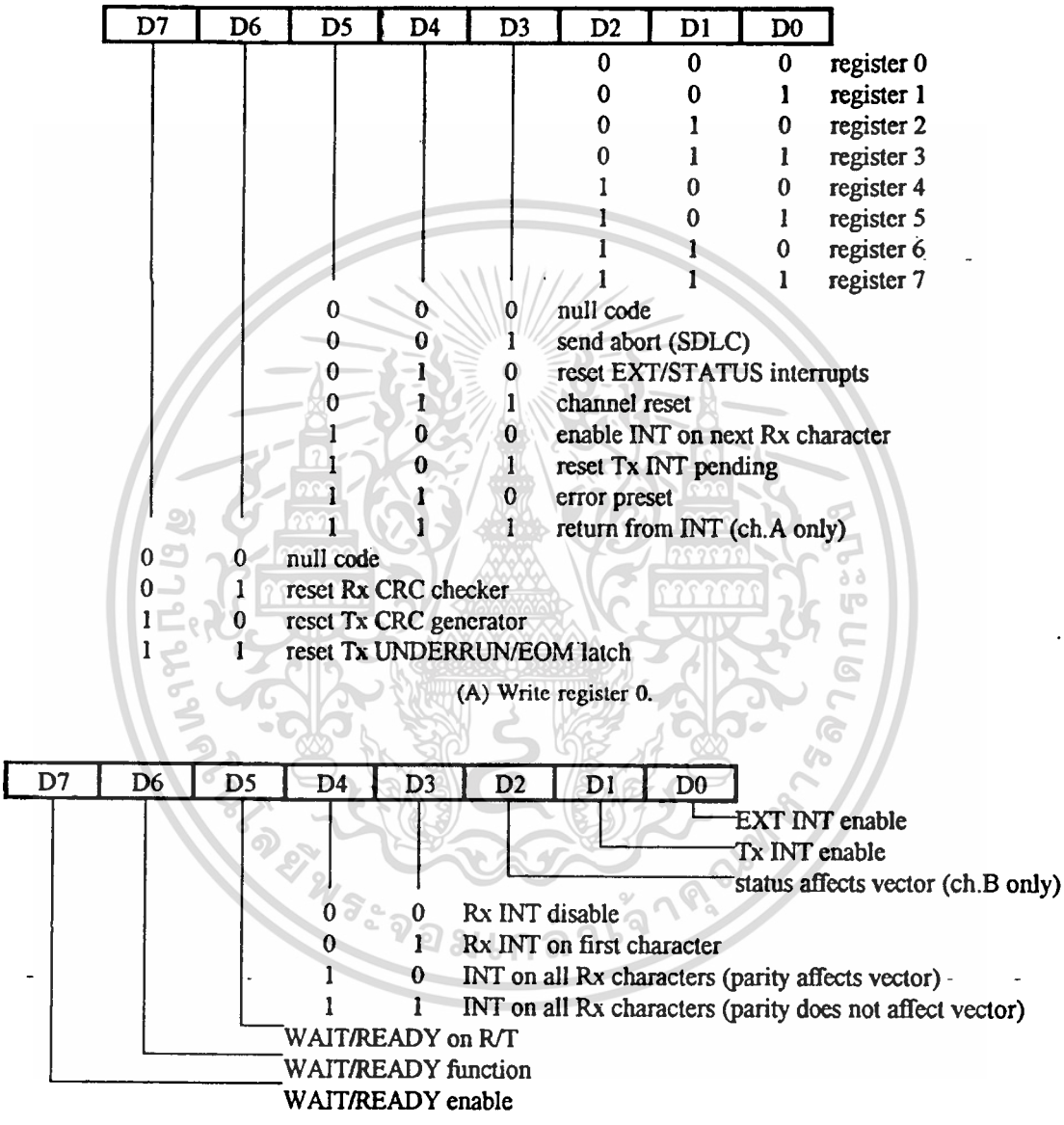
```
        //}}AFX_MSG
        ..............
        ..............
}
```

- การประกาศตัวแปร ที่เป็นตัวชี้ (pointer) ภายในระหว่างส่วนที่ไม่ใช่ข้อมูลของโปรแกรมจะส่งผลกระทบต่อคลาส แต่การคอมไพล์จะไม่มีปัญหา

```
        //{{AFX_DATA(Cadlg)
        enum { IDD = IDD_DIALOG2 };
                // NOTE: the ClassWizard will add data members here
         char *st;
        //}}AFX_MSG
```

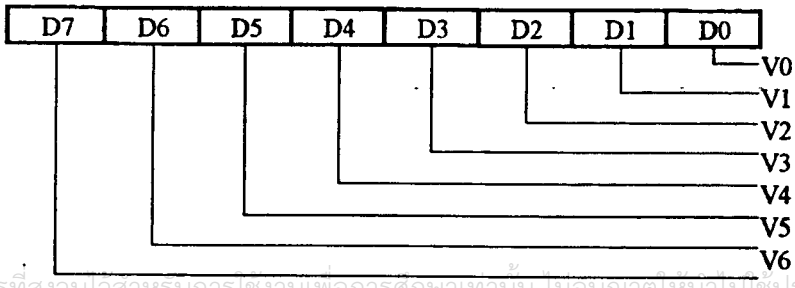
ภาคผนวก ก. รีจิสเตอร์ต่างๆ ของไอซี Z80-SIO

Z80-SIO write registers



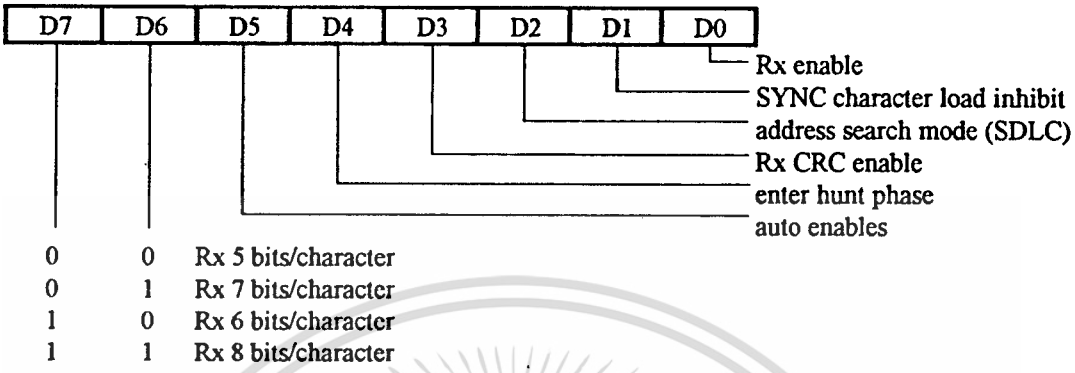
(A) Write register 0.

(B) Write register 1.

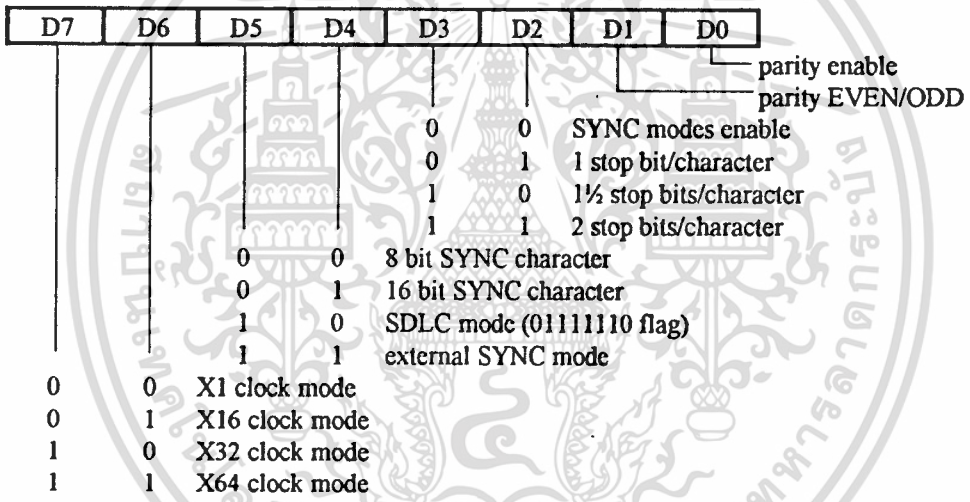


V7

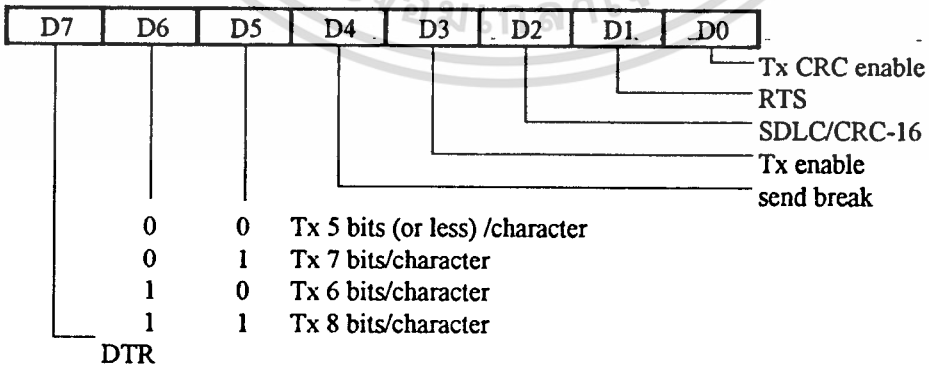
(C) Write register 2.



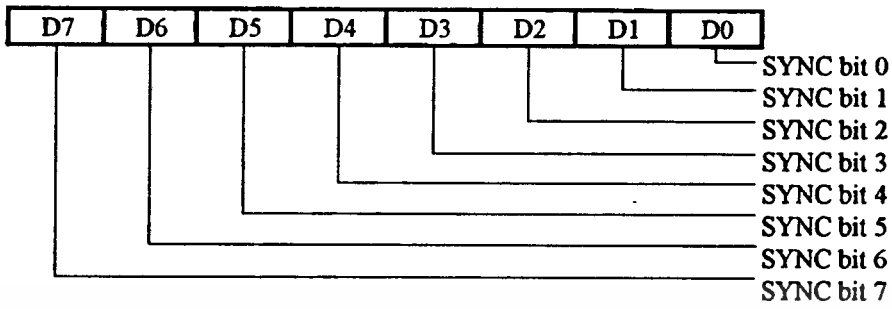
(D) Write register 3.



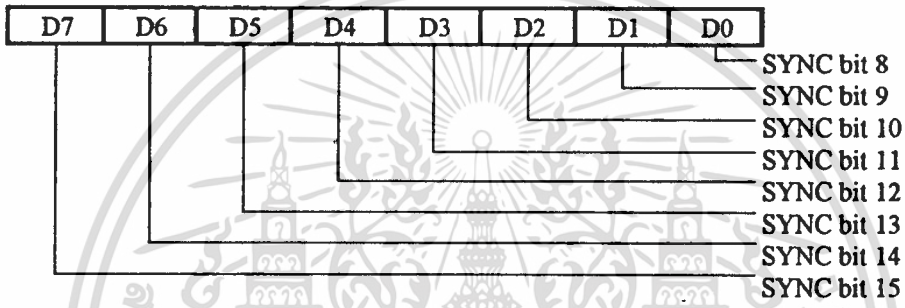
(E) Write register 4.



(F) Write register 5.

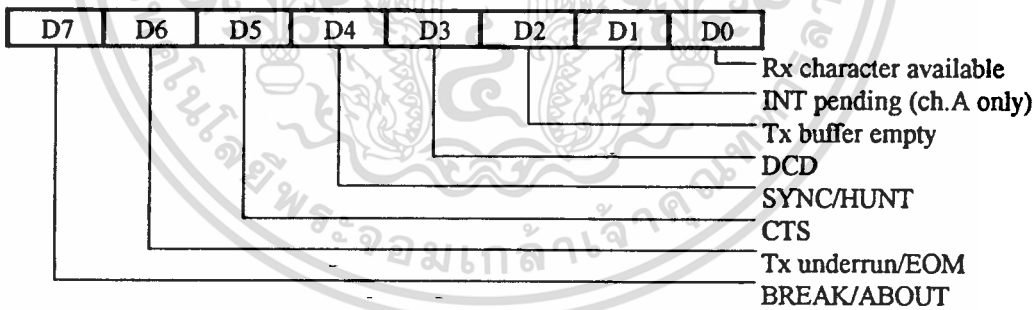


(G) Write register 6.

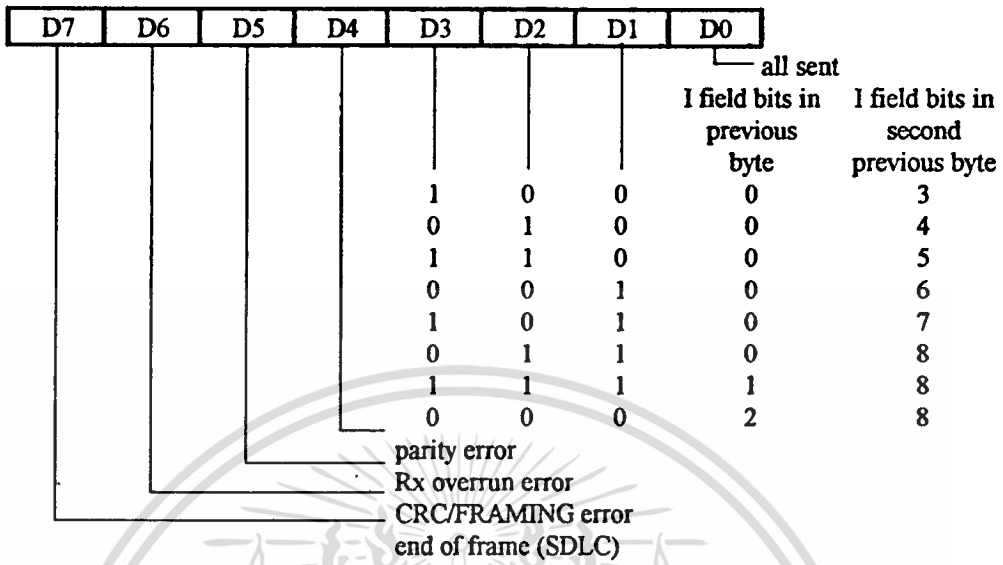


(H) Write register 7.

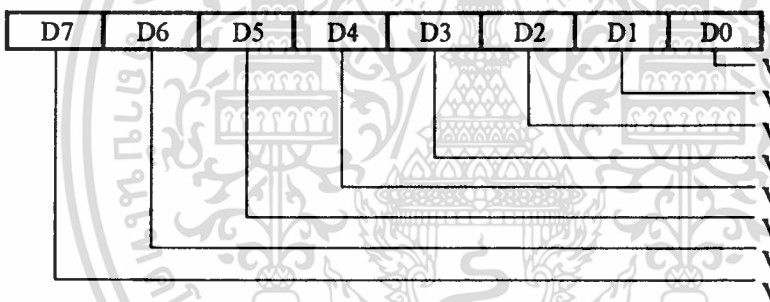
Z80-SIO read registers



(A) Read register 0.



(B) Read register 1.



(C) Read register 2.

ภาคผนวก ข. การโปรแกรมค่ารีจิสเตอร์ให้กับไอซี 8253

ไอซีเบอร์ 8253 เป็นชิพซัพพอร์ทที่ทำหน้าที่ในการสร้างฐานเวลาต่างๆ ให้กับระบบ และภายใน 8253 จะประกอบไปด้วยแชนแนลที่ทำงานเป็นอิสระต่อกันอยู่ 3 แชนแนล ซึ่งการที่แชนแนลทั้ง 3 ของ 8253 จะทำงานตามที่เราต้องการได้นั้น เราจะต้องทำการโปรแกรมแชนแนลเหล่านี้เสียก่อน

สำหรับการโปรแกรม 8253 นั้น จะทำได้โดยใช้คำสั่ง out ส่งข้อมูล (คำสั่ง) ให้กับรีจิสเตอร์ต่างๆ ของ 8253 ซึ่งรีจิสเตอร์ของ 8253 จะแบ่งออกได้เป็น 2 ประเภท คือ รีจิสเตอร์โหมดคอนโทรล (Mode Control) และรีจิสเตอร์เคาน์เตอร์ (Counter) ดังนั้น ก่อนที่จะศึกษาถึงวิธีการโปรแกรม 8253 จึงจำเป็นต้องทำความเข้าใจกับหน้าที่ของบิตต่างๆ ภายในรีจิสเตอร์ทั้ง 2 นี้เสียก่อน

รีจิสเตอร์เคาน์เตอร์ (Counter Register)

ภายใน 8253 จะมีรีจิสเตอร์ Counter ซึ่งเป็นรีจิสเตอร์ขนาด 16 บิทอยู่ 3 ตัว โดยข้อมูลในรีจิสเตอร์แต่ละตัวจะควบคุมช่วงเวลาของเอาท์พุทในแต่ละแชนแนล

เมื่อ 8253 เริ่มต้นการทำงานในแชนแนลใดแล้ว ข้อมูลในรีจิสเตอร์ Counter ของแชนแนลนั้นจะถูกลดค่าลง 1, 2 หรือ 3 ขึ้นอยู่กับโหมดการทำงาน เมื่อแชนแนลนั้นได้รับสัญญาณคล็อกแต่ละลูก ข้อมูลในรีจิสเตอร์เคาน์เตอร์ จะถูกลดลงจนมีค่าเป็น “0” (Terminal Count) จากนั้น จึงจะเกิดการเปลี่ยนแปลงทางด้านเอาท์พุทและการทำงานของแชนแนลนั้น

รีจิสเตอร์โหมดคอนโทรล (Mode Control Register)

สามารถโปรแกรมได้โดยใช้คำสั่ง out ส่งข้อมูลไปยังพอร์ทของ 8253 ที่มีแอดเดรส A0 และ A1 เป็น “1” ทั้งคู่ ซึ่งแต่ละบิทจะมีหน้าที่ต่างๆ ดังนี้

บิท	D7	D6	D5	D4	D3	D2	D1	D0
หน้าที่	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

บิท D7 - D6 : ข้อมูลในบิททั้งสองนี้จะใช้สำหรับเลือกแชนแนลที่ต้องการ ซึ่งจะแสดงได้ดังนี้ (เนื่องจาก 8253 มีเพียง 3 แชนแนล ดังนั้นกรณีที่ข้อมูลในบิท SC1 และ SC0 เป็น "1" ทั้งคู่จึงไม่มีผลในการเลือกแชนแนลใดๆ)

SC1	SC0	แชนแนลที่ถูกเลือก
0	0	0
0	1	1
1	0	2
1	1	-

บิท D5 - D4 : ใช้ในการกำหนดไบท์ของข้อมูลที่ต้องการจะอ่านจากรีจิสเตอร์ Counter

RL1	RL0	หน้าที่
0	0	ทำการแลทช์ค่าในรีจิสเตอร์ Counter
0	1	อ่าน/เขียนเฉพาะข้อมูล 8 บิทล่าง
1	0	อ่าน/เขียนเฉพาะข้อมูล 8 บิทบน
1	1	อ่าน/เขียนเฉพาะข้อมูล 16 บิท (เริ่มจาก 8 บิทล่างก่อน)

บิท D3 - D1 : ข้อมูลในบิทนี้ใช้สำหรับเลือกโหมดการทำงานให้กับแชนแนลที่เรากำหนดในบิท SC1 และ SC0

M2	M1	M0	โหมดการทำงาน
0	0	0	โหมด 0 : Interrupt on Terminal Count
0	0	1	โหมด 1 : Programmable One-Shot
0	1	0	โหมด 2: Rate Generator
0	1	1	โหมด 3: Square Wave Generator
1	0	0	โหมด 4: Software Triggered Strobe
1	0	1	โหมด 5: Hardware Triggered Strobe

ภาคผนวก ค. รูปแบบไฟล์มีดี้มาตรฐาน

รูปแบบไฟล์มาตรฐานของมีดี้ (Standard MIDI File Format) ถูกกำหนดขึ้นในปี ค.ศ. 1988 โดย Dave Oppenheimer เพื่อแก้ปัญหาการใช้ไฟล์เพลงร่วมกันไม่ได้ ทั้งนี้เพราะในช่วงก่อนที่จะมีการกำหนดรูปแบบไฟล์มาตรฐานของมีดี้ขึ้นมานั้น ซอฟต์แวร์ซีเควนเซอร์ที่ทำกันออกมาจะมีการกำหนดรูปแบบของไฟล์เพลงกันเอง ทำให้ซอฟแวร์แต่ละตัวไม่สามารถใช้งานไฟล์ร่วมกันได้หรือทำได้ไม่สะดวกนัก

ไฟล์มีดี้จะประกอบด้วยกลุ่มของข้อมูลหลายๆ กลุ่ม ซึ่งกลุ่มของข้อมูลที่ประกอบขึ้นเป็นไฟล์มีดี้จะมีอยู่ 2 ชนิด คือ CHUNK และ TRACK_CHUNK ไฟล์มีดี้ส่วนใหญ่จะประกอบด้วย CHUNK เพียงอันเดียวซึ่งเป็นตัวบอกค่าต่างๆ ของไฟล์เพลง เช่น ความเร็วของเพลง, ฐานเวลาของเพลง (Time Signature), ค่าของความดัง ฯลฯ ส่วน TRACK_CHUNK จะมีหลายอันซึ่งเป็นตัวบอกข้อมูลมีดี้ เช่น โน้ตออน, โน้ตออฟ, ชนิดเครื่องดนตรี ฯลฯ

CHUNK จะเริ่มต้นด้วยอักขระ 4 ตัว คือ MThd และตามด้วยไบต์อีก 4 ไบต์ที่เป็นตัวบอกขนาดของข้อมูล ซึ่งส่วนใหญ่จะมีค่าเท่ากับ 6 โดยสองไบต์แรกจะหมายถึงชนิดของไฟล์ (format 0 หรือ format 1) สองไบต์ถัดมาจะบอกจำนวนของแทร็คของเพลง และสองไบต์สุดท้ายจะเป็นค่าของตัวหาร

TRACK_CHUNK จะเริ่มต้นด้วยอักขระ 4 ตัว คือ Mtrk ตามด้วยข้อมูล 4 ไบต์ซึ่งเป็นตัวบอกความยาวของข้อมูลที่อยู่ในแทร็คหลังจากนั้นก็จะเป็นข้อมูลอีเวนต์ (event) ต่างๆ ที่เกิดขึ้นในเพลง

```
Displacement ------------------ Hex codes ------------------  ASCII value
 0000(0000)  4D 54 68 64 00 00 00 06 00 01 00 09 01 E0 4D 54  MThd  ♠ ☺ ☺ ☺ ☺MT
 0016(0010)  72 6B 00 00 00 25 00 FF 01 05 53 65 71 2D 31 00  rk   %  ☺ ☺Seq-1
 0032(0020)  FF 54 05 60 00 00 00 00 00 FF 58 04 04 02 18 08   T♣`      X♦♦☺↑◘
 0048(0030)  00 FF 51 03 0D 44 BD 00 FF 2F 00 4D 54 72 6B 00    Q♥♪D╜  / MTrk
 0064(0040)  00 00 16 00 FF 01 0E 28 43 29 20 26 20 28 50 29    ▬  ☺♫(C) & (P)
 0080(0050)  20 31 39 39 32 00 FF 2F 00 4D 54 72 6B 00 00 00   1992  / MTrk
 0096(0060)  19 00 FF 01 11 52 6F 79 61 6C 74 79 20 4D 75 73  ↓  ☺◄Royalty Mus
 0112(0070)  20 49 6E 74 6C 2E 00 FF 2F 00 4D 54 72 6B 00 00   Intl.  / MTrk
 0128(0080)  00 1B 00 FF 01 13 43 61 6C 6C 20 66 6F 72 20 6D  ←  ☺!!Call for m
 0144(0090)  6F 72 65 20 69 6E 66 6F 21 00 FF 2F 00 4D 54 72  ore info!  / MTr
 0160(00A0)  6B 00 00 00 16 00 FF 01 0E 31 2D 36 31 37 2D 32  k  _  ☺♫1-617-2
 0176(00B0)  35 34 2D 39 31 30 39 00 FF 2F 00 4D 54 72 6B 00  54-9109  / MTrk
 0192(00C0)  00 01 85 00 FF 01 0A 53 68 61 6B 75 68 61 63 68   ☺à  ☺◙Shakuhach
 0208(00D0)  69 00 C0 4D 00 90 4A 5A 83 60 80 4A 5A 00 90 48  i └M ÉJZâ`ÇJZ ÉH
 0224(00E0)  5A 83 60 80 48 5A 00 90 4A 5A 83 60 80 4A 5A 00  Zâ`ÇHZ ÉJZâ`ÇJZ
 0240(00F0)  90 4C 5A 83 60 80 4C 5A 00 90 4F 5A 83 60 80 4F  ÉLZâ`ÇLZ ÉOZâ`ÇO
```

รูปแสดงไบต์ข้อมูลไฟล์มีดี้ตัวอย่าง (ชื่อไฟล์ JAPAN.MID)

เมตาอีเวนต์ (META EVENT)

เมตาอีเวนต์ คือ อีเวนต์จำพวก คำสั่ง และข้อมูลต่างๆ มีไว้สำหรับให้ซอฟต์แวร์ใช้เป็นข้อกำหนดในการเล่นไฟล์มีดี้

ไบต์แรกของเมตาอีเวนต์จะมีค่าเท่ากับ 0FFh และถัดมาในไบต์ที่สองจะเป็นตัวบอกชนิดของเมตาอีเวนต์ ไบต์ที่สามจะบอกความยาวของข้อมูลที่จะใช้เป็นเมตาอีเวนต์

second byte	type of META EVENT
1	text meta event
3	sequence/track name
4	instrument name
20h	MIDI channel prefix
2Fh	end of track
51h	set tempo
58h	time signature

ตารางแสดงความสัมพันธ์ระหว่างค่าในไบต์ที่สองกับชนิดของเมตาอีเวนต์

กิตติกรรมประกาศ

ในการจัดทำโครงงานพิเศษเรื่องนี้ ทางคณะผู้จัดทำได้รับความอนุเคราะห์ช่วยเหลือในเรื่องต่างๆ ตั้งแต่เริ่มต้น จนกระทั่งสำเร็จลุล่วงไปด้วยดี จากบุคคลต่างๆ ดังรายนามต่อไปนี้

- อ.สมศักดิ์ มิตะถา
- อ.ประสาร ตังติสานนท์
- อ.วัชระ ฉัตรวิริยะ
- อ. กฤตวัน เครือตราชู
- เพื่อนๆ ทุกคนที่ให้ความช่วยเหลือและคำปรึกษา

หนังสืออ้างอิง

1. จิรพัฒน์ จันทร์เจิด และ วีระ นพนิราพาธ, "เขียนโปรแกรมบนไมโครซอฟต์วินโดวส์", ซีเอ็ด, พ.ศ. 2536
2. ประสิทธ์ วรฉัตราวณิช และ อุทัย เกียรติวิกรัย, "มีดี้ตำนานบทใหม่ของโลกดนตรี", วารสารคอมพิวเตอร์รีวิว, ปีที่ 10, ฉบับที่ 102, พ.ศ. 2536, หน้า 175-186
3. สมศักดิ์ อัศวกุลไพบูลย์, "C++ Application Frameworks", วารสารพีซีแมกกาซีน, ปีที่ 2, ฉบับที่ 21, พ.ศ. 2537, หน้า 165-170
4. Jim Conger, "MIDI Sequencing in C", M&T BOOKS, 1989.
5. Ori Gurewich & Nathan Gurewich, "Master Visual C++ 1.5", Sams Publishing, 1994.