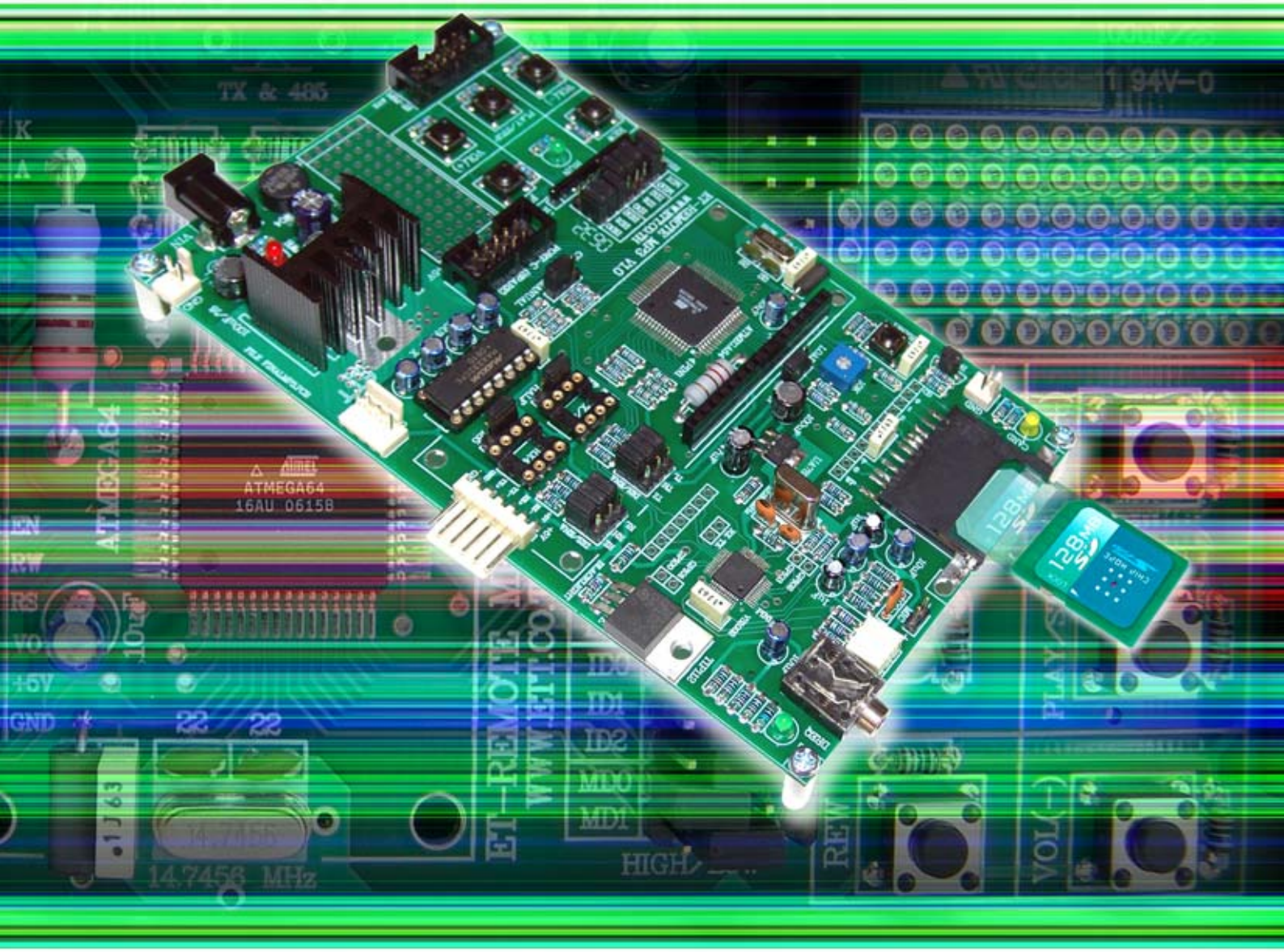
คู่มือการใช้งานบอร์ด

ET-REMOTE MP3 V1.0



ETT
www.ett.co.th
บริษัท อีทีที จำกัด
1112/96-98 ถนนสุขุมวิท แขวงพระโขนง เขตคลองเตย กรุงเทพฯ 10110 http:/www.etteam.com
1112/96-98 Sukhumvit Rd., Phrakanong Klongtoey Bangkok 10110 http:/www.ett.co.th
Tel : 02-7121120 Fax : 02-3917216 email : sale@etteam.com

สารบัญ

ET- REMOTE MP3. V1.0

1. คุณสมบัติของบอร์ด ET-REMOTE MP3.

1.1 เป็นบอร์ดสำหรับเล่นไฟล์ MP3 หรือ WAV เท่านั้น ไม่สามารถ Record เสียงได้

1.2 ใช้ SD/MMC Card เป็นตัวเก็บข้อมูล ; ให้สัญญาณเสียงทาง Output แบบ Stereo

1.3 มีโหมดการทำงานให้เลือก 2 โหมด คือ –Manual Mode:ควบคุมการเล่นโดยผ่านสวิตช์ 5 ตัว บนบอร์ด และ -Control Mode: ควบคุมการเล่นโดยส่ง เป็น Command ผ่านทาง RS232 หรือ RS422 และ RS485 แบบ 4 เส้น

1.4 มีPort สำหรับต่อ LCD แสดงผลการทำงานขณะอยู่ในโหมด Manual ซึ่ง LCD จะแสดงหมายเลขเพลง , ชื่อและนามสกุลของเพลงที่กำลังเล่นอยู่ และแสดงสถานะของสวิตช์แต่ละตัวที่กด ,แสดงเวลาขณะเล่น

1.5 ไม่สามารถแสดงชื่อไฟล์ที่เป็นภาษาไทยบนจอ LCDได้ แต่สามารถนำไฟล์นั้นมาเล่นได้

1.6 ชื่อไฟล์ที่แสดงบนจอ LCD จะแสดงได้ไม่เกิน 8 ตัวอักษร และลำดับเพลงจะแสดงได้ตั้งแต่ 001- 999

1.7 รูปแบบการควบคุม :เดินหน้า1เพลง ,ถอยหลัง1เพลง ,ปรับ Volume ,ปรับ BASS ,PLAY/STOP,เล่นซ้ำ1เพลง (R1) ,เล่นซ้ำทั้งหมด(<->) , เล่นเพลงทั้งหมด 1รอบ(->) , เล่นเพลงเดียวจบ(x1)

1.8 มี LED แสดงสถานะ การใส่การ์ด และแสดงสถานะ Error บนจอ LCD ถ้าไม่ได้ใส่การ์ด

1.9 ในโหมด Command Control สามารถที่จะเลือกเรียงไฟล์ให้เล่นได้ตามความต้องการ โดยเรียงตามชื่อไฟล์ที่จะเล่นได้ประมาณ 50 ไฟล์(หรือมากกว่าถ้าชื่อไฟล์สั้นลง) หรือเรียงตามหมายเลขลำดับไฟล์ที่จะเล่นได้ 110 ไฟล์

1.10 ในโหมด Command Control ในระหว่างที่เล่นไฟล์นั้นๆอยู่ สามารถส่งไฟล์ใหม่ให้เข้าไปเล่นแทรกได้ เมื่อไฟล์ใหม่ที่ส่งไปเล่นจบ ก็จะกลับมาเล่นไฟล์ที่ค้างอยู่ ต่อได้

1.11 กำหนด Baud Rate ในการส่ง Command ได้ 8 ค่า คือ 1200,2400,4800,9600,19200,28800,38400,57600

1.13 แสดงเวลาการเล่นออกที่จอ LCD และส่งเป็น Command ออกมาทาง RS232/422/485 รวมทั้ง สามารถกำหนด ID Board ได้ 32 ID ตั้งแต่ ‘00’-‘31’

1.14 เมื่อเล่นจบไฟล์แต่ละไฟล์แล้วจะส่ง Command End of File ออกมาทาง RS232/422/485 รวมทั้งส่งเป็นพัลซ์ (Logic‘0’) ออกมาที่ขั้วต่อ EOF ที่มีอยู่บนบอร์ดด้วย

2. ข้อจำกัดการใช้งานบอร์ด ET-REMOTE MP3.

2.1 SD/MMC Card ที่นำมาใช้ควรจะมีความเร็วในการอ่านเที่สูงๆ ยี่ห้อที่แนะนำให้ใช้เช่น Kington,RIDATA ส่วนของ Sandisk ไม่แนะนำให้ใช้กับบอร์ดนี้เพราะมักจะมีปัญหาในการอ่าน

2.2 SD/MMC Card ที่จะนำมาเก็บเพลงนั้น จะต้องทำการ Format ระบบไฟล์เป็นแบบ FAT16 เท่านั้น

2.3 บอร์ดนี้จะเล่นไฟล์ที่อยู่ภายนอก Folder เท่านั้น ไม่สามารถเล่นไฟล์ที่เก็บอยู่ใน Folder ได้

2.4 File Support : File Type = MP3,WAV

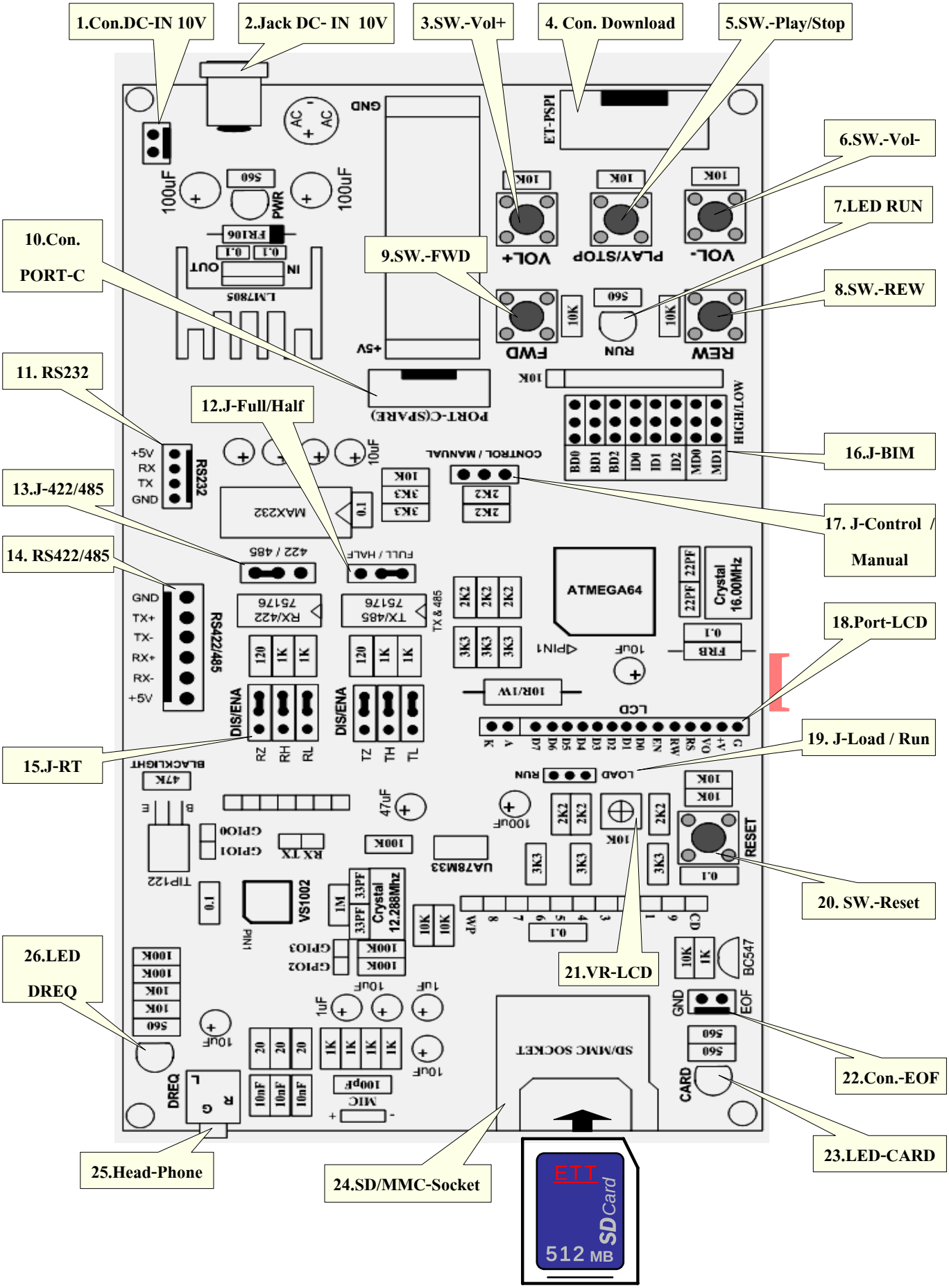
BitRate = MP3 ~ 8Kbps – 192 Kbps , WAV ~176 Kbps(Format PCM,Sample Rate 11Khz)

2.5 การตั้งชื่อไฟล์เป็นไปตามมาตรฐานทั่วไป แต่ห้ามใช้อักษรที่เหมือนกับ Command ได้แก่ $ * # @ ; : =

2.6 ความยาวชื่อไฟล์จะแสดงให้เห็นได้ไม่เกิน 8 ตัวอักษร(กรณีชื่อไฟล์ยาวเกิน8ตัวอักษร) ตามมาตรฐานFAT16

2.7 เมื่อบอร์ดถูก Reset การทำงานหรือการ Set ค่าต่างๆ จะไม่สามารถเก็บจำไว้ได้ จะกลับมา Set ที่ค่า Default เสมอ

3. ลักษณะของบอร์ด ET-REMOTE MP3.



รูปแสดงโครงสร้างของบอร์ด ET-REMOTE MP3.

หมายเลข 1 ,2 Connector DC-IN, Jack DC-IN (7-12V) เป็นขั้วต่อไฟเลี้ยงให้กับบอร์ด โดยจะรับแรงดัน VDC 7-12 V สามารถสลับขั้วได้เนื่องจากมีบริดจ์ไดโอดป้องกันอยู่

หมายเลข 3 SW.-VOL+ เป็นสวิตช์สำหรับเพิ่มความดังเสียง(สำหรับ Manual Mode)

หมายเลข 4 Connector Download Connector นี้จะใช้สำหรับ Download code ลงใน MCU AVR Mega64,128 เมื่อต้องการจะ Up date Software ตัวใหม่

หมายเลข 5 SW.-PLAY/STOP เป็นสวิตช์สำหรับสั่ง เล่นเพลง และ หยุดเล่นเพลง (สำหรับ Manual Mode)

หมายเลข 6 SW.-VOL- เป็นสวิตช์สำหรับลดความดังเสียง(สำหรับ Manual Mode)

หมายเลข 7 LED RUN คือ LED แสดงสถานะ ซึ่งใน Manual Mode ถ้าบอร์ดอยู่ในสภาวะพร้อม LED จะติด จากนั้นเมื่อสั่ง Play LED จะกระพริบเพื่อแสดงสถานะ การเล่น และเมื่อสั่ง Stop LED จะดับลง , ถ้าอยู่ใน Control Mode LEDจะกระพริบเมื่อมีการรับCommand จากภายนอกเข้ามาทางRS232/422/485 เท่านั้น

หมายเลข 8 SW.-REW(Back) เป็นสวิตช์สำหรับเลื่อนเพลงถอยหลังครั้งละ 1 เพลง ซึ่งจะต้องกดภายใน 5 วินาที หลังจากเพลงเริ่มเล่น เพลงก็จะถอยหลังให้ ถ้าเกินเวลานี้จะเป็นการเริ่มต้นเพลงที่เล่นอยู่ขณะนั้น ใหม่

หมายเลข 9 SW.-FWD(Next) เป็นสวิตช์สำหรับเลื่อนเพลงไปข้างหน้าครั้งละ 1 เพลง

หมายเลข 10 Connector PORT C สำหรับPort นี้จะจำลองเป็น VU ไฟวิ่ง 2รูปแบบเพื่อความสวยงาม โดยผู้ใช้สามารถนำ LED 8 ดวง ต่อในลักษณะ Common Anode มาต่อเข้าที่ Port นี้ได้ เมื่อสั่งเล่นเพลง LED ก็จะวิ่งตามรูปแบบที่ทางเราได้กำหนดไว้

หมายเลข 11 RS232 เป็น Connector สำหรับต่อสายสัญญาณ RS232 เมื่อต้องการจะสื่อสารหรือ ส่ง Command จากภายนอกเข้ามาควบคุมบอร์ด ET-REMOTE MP3.

หมายเลข 12 Jumper Full/Half เป็น Jumper สำหรับกำหนดให้ IC Line Driver 75176 ทำงาน 1 ตัว หรือ 2 ตัว ซึ่งเมื่อจะใช้การสื่อสารแบบ RS422 ให้ Set มาทางด้าน Full (IC ทำงาน 2 ตัว) ถ้าใช้การสื่อสารแบบ RS485 ก็ให้ Set Jumper มาทางด้าน Full ด้วยเช่นกัน เนื่องจากในบอร์ดนี้ได้กำหนดให้ การสื่อสารแบบ RS485 นั้นจะ ต้องใช้สายสัญญาณ 4 เส้น

หมายเลข 13 Jumper 422/485 เป็น Jumper สำหรับเลือกรูปแบบการสื่อสารแบบ RS422 หรือ RS485

หมายเลข 14 RS422/485 เป็น Connector สำหรับต่อสายสัญญาณ RS422/485 เมื่อต้องการจะสื่อสารหรือส่ง Command จากภายนอกเข้ามาควบคุมบอร์ด ET-REMOTE MP3.

หมายเลข 15 Jumper -RT เป็น Jumper สำหรับเพิ่มระยะการสื่อสารแบบ RS422/485 ให้ได้ไกลขึ้นกว่าปกติ โดยให้ Set Jumper RZ,RH,RL,TZ,TH,TL มาทางด้าน Enable

หมายเลข 16 Jumper J-BIM เป็น Jumper สำหรับเลือก Baud Rate(BD0-BD2) ,เลือก ID (ID0-ID2,MD0-MD1) ใน Manual Mode MD1 จะถูกใช้สำหรับเลือกฟังก์ชันการควบคุม สวิตช์ ส่วน MD0 จะไม่ถูกใช้งานในโหมดนี้ รายละเอียดในการ Set jumper BIM ดูได้ในตารางด้านล่าง

ตารางแสดงการ Set Jumper BIM :

-ตารางที่3.1 การ Set Function Switch (MD0-MD1) Use For Manual Mode

MD0 = ไม่ใช้งาน
MD1 = 0 : กำหนดให้ SW. PLAY/STOP ทำหน้าที่ Play และเป็น key Shift ใช้งานร่วมกับ SW. อื่นๆ **MD1** = 1 : กำหนดให้ SW. PLAY/STOP ทำหน้าที่ Play และ Stop การเล่น ในปุ่มเดียวกัน

- ตารางที่ 3.2 การ Set Baud Rate(BD0-BD2)

BD0	BD1	BD2	BAUD RATE
0	0	0	1200
0	0	1	2400
0	1	0	4800
0	1	1	9600
1	0	0	19200
1	0	1	28800
1	1	0	38400
1	1	1	57600

- ตารางที่3.3 การ Set ID Board(ID0-ID2,MD0-MD1)

ID0	ID1	ID2	MD0	MD1	ID CODE (ASCII 2 Character)
0	0	0	0	0	‘00’
0	0	0	0	1	‘01’
0	0	0	1	0	‘02’
0	0	0	1	1	‘03’
0	0	1	0	0	‘04’
0	0	1	0	1	‘05’
0	0	1	1	0	‘06’
0	0	1	1	1	‘07’
0	1	0	0	0	‘08’
0	1	0	0	1	‘09’
0	1	0	1	0	‘10’
0	1	0	1	1	‘11’
0	1	1	0	0	‘12’
0	1	1	0	1	‘13’
0	1	1	1	0	‘14’

- ตารางที่3.3 การ Set ID Board(ID0-ID2,MD0-MD1) ต่อ

ID0	ID1	ID2	MD0	MD1	ID CODE (ASCII 2 Character)
0	1	1	1	1	‘15’
1	0	0	0	0	‘16’
1	0	0	0	1	‘17’
1	0	0	1	0	‘18’
1	0	0	1	1	‘19’
1	0	1	0	0	‘20’
1	0	1	0	1	‘21’
1	0	1	1	0	‘22’
1	0	1	1	1	‘23’
1	1	0	0	0	‘24’
1	1	0	0	1	‘25’
1	1	0	1	0	‘26’
1	1	0	1	1	‘27’
1	1	1	0	0	‘28’
1	1	1	0	1	‘29’
1	1	1	1	0	‘30’
1	1	1	1	1	‘31’

หมายเลข 17 Jumper Control / Manual เป็น Jumper สำหรับใช้เลือกโหมดควบคุมการเล่นของบอร์ด โดยถ้า Set มาทางด้าน Manual จะเป็นการเลือกควบคุมการเล่นผ่านทาง SW. ทั้ง 5 บนบอร์ด และถ้า Set มาทางด้าน Control จะเป็นการเลือกควบคุมการเล่นด้วยการส่ง Command ผ่านทาง RS232/422/485

หมายเลข 18 Port LCD เป็น Port สำหรับต่อ LCD ขนาด 16 ตัวอักษร 2 แถว(16x2) เพื่อแสดงผลการทำงานใน Manual Mode โดยจะต้องต่อขาเรียงให้ถูกต้องตามบอร์ด

หมายเลข 19 Jumper Load / Run เป็น Jumper สำหรับเลือก Download Software ที่จะ Update ลงใน MCU โดยเมื่อจะ Update Software ต้อง Set มาด้าน Load เมื่อจะใช้งานบอร์ดตามปกติให้ Set มาด้าน Run

หมายเลข 20 SW. – Reset เป็นสวิตช์ สำหรับ Reset MCU ให้เริ่มทำงานใหม่ เมื่อเกิด กรณีบอร์ด แฮงค์

หมายเลข 21 VR-LCD เป็น VR ปรับค่า ใช้สำหรับปรับความเข้มของตัวอักษรที่จอ LCD ที่นำมาต่อใช้งาน

หมายเลข 22 Connector-EOF เป็น Connector สำหรับส่งสัญญานพัลซ์ Logic ‘0’ ออกมา 1 ลูก เมื่อมีการเล่นเพลงจบลง(จะส่งพัลซ์เมื่อมีการเล่นเพลงจบแล้วเท่านั้น ถ้าเป็นการเลื่อนเพลงจะไม่มีการส่งพัลซ์ออกมา) ในสภาวะปกติ ที่ Pin EOF นี้จะมีสถานะเป็น Logic ‘1’ อยู่เสมอ

หมายเลข 23 LED-CARD เป็น LED แสดงสถานะของ Card คือ เมื่อใส่ Card LED จะติด ถ้าไม่มี Card LED จะดับ

หมายเลข 24 SD/MMC-Socket เป็น Socket สำหรับใส่ SD/MMC Card เมื่อจะนำเพลงมาเล่นกับบอร์ด

หมายเลข 25 Head-Phone เป็นจุดต่อสัญญาณเสียง Output (Stereo) ออกไปใช้งาน โดยถ้าเป็นชุดหูฟังสามารถนำมาต่อฟังได้ทันที แต่ถ้าเป็นลำโพงจะต้องผ่านชุดขยายเสียงก่อน อย่านำมาต่อโดยตรง จะทำให้ Chip MP3 เสียหายได้

หมายเลข 26 LED-DREQ เป็น LED แสดงสถานะ การรับส่งข้อมูลระหว่าง Chip MP3 กับ MCU ถ้าไม่มีการรับส่งข้อมูล(Stop) LED นี้จะดับอยู่ ถ้ามีการรับส่งข้อมูลเกิดขึ้น(Play) LED นี้จะกระพริบ

4. สภาวะเริ่มต้นของบอร์ด ET-REMOTE MP3.หลังจาก Reset

4.1 Baud Rate ในการสื่อสารแบบ Serial Port(RS232/422/485) จะขึ้นอยู่กับ Jumper BD0-BD2(16) ที่ผู้ใช้กำหนด

4.2 รูปแบบการเล่นจะถูกกำหนดไว้ที่ Normal คือ เล่นเพลงทั้งหมด 1รอบแล้วจบ

4.3 โหมดการเล่นจะถูกเลือกตาม Jumper(17) ที่ Set ว่าอยู่ในโหมด Manual(SW.) หรือ Control(RS232/422/485)

4.4 ID Board และ Baud Rate นั้นจะถูกกำหนดหลังจากมีการรีเซตบอร์ดเพียงครั้งเดียว ดังนั้นถ้ามีการเปลี่ยน Baud Rate หรือ ID Board ใหม่ทุกครั้ง จะต้อง Reset Board ET-REMOTE MP3. ใหม่เสมอถึงจะมีผลตามที่ Set ใหม่

4.5 ระดับความดังเสียงจะถูกกำหนดระดับเริ่มต้นไว้ที่ค่า 14 (Scale 0-20) สำหรับ Manual Mode และ 225(000-255) สำหรับControl Mode

4.6 BASS จะถูกปรับไว้ที่ OFF สำหรับเสียง เบสนี้จะตอบสนองได้เต็มที่เมื่อปรับระดับความดังของเสียงจากบอร์ดอยู่ที่ค่า 14 หรือ 225 ถ้าปรับระดับความดังเกินจากนี้เสียงเบสจะลดต่ำลง ;Delay ของ VU จะถูก Set ที่ค่า 12

4.7 เวลาในการเล่น จะยังไม่ถูกส่งออกทาง RS232/422/485 ถ้าต้องให้ส่งเวลาออกมาจะต้องส่ง Command 08 เพื่อ Enable ให้บอร์ดส่งเวลาในการเล่นออกมาทาง RS232/422/485 (ดูลายละเอียดที่ Command 08)

5. การใช้งานบอร์ด

5.1 การใช้งานใน Manual Mode (PLAY SW.MODE)

การใช้งานในโหมดนี้จะเป็นการควบคุมการเล่นผ่านสวิตช์ทั้ง 5 ตัวบนบอร์ด ได้แก่ SW-PLAY/STOP , SW-VOL(-) , SW-VOL(+) , SW-REW , SW-FWD โดยหน้าที่และรูปแบบการใช้งานของสวิตช์แต่ละตัวดูได้ในตารางด้านล่าง

เริ่มต้นการใช้งานในโหมดนี้จะต้องทำการ Set Jumper Control/Manual(17) มาทางด้าน Manual และ Jumper Load/Run(19) มาทางด้าน Run เสมอ จากนั้นต่อ LCD (ถ้ามี) เข้ากับบอร์ดที่ Port-LCD(18) ตามขั้วที่กำหนด หรือจะต่อสาย สัญญาณ RS232 จาก PC เข้ากับขั้วต่อ RS232 ของบอร์ด และใช้ โปรแกรม Hyperterminal หรือ Procomm เพื่อดูผลการทำงานของบอร์ดแทน LCD โดยกำหนด Baud Rate ของโปรแกรม ให้ตรงกับที่ผู้ใช้กำหนดจาก Jumper บนบอร์ด จากนั้น ทำการใส่ SD/MMC Card ที่เก็บ File MP3 หรือ Wave ไว้แล้ว เข้ากับ Socket ของบอร์ดจะเห็น LED Card สีเขียวติด จากนั้นตัวบอร์ดก็จะทำการตรวจสอบ Card ถ้าบอร์ด Initial Card ผ่าน ที่จอ LCD จะ แสดงดังนี้

“ ET-REMOTE MP3 V1 ”

“CARD OK. ”

ข้อความด้านบนนี้จะค้างอยู่ชั่วขณะ แล้วจะมีข้อความขึ้นมาที่จอ LCD ใหม่ดังนี้

“PL-MANUAL MODE ”

“ET-MP3. OK. ”

เมื่อข้อความนี้ปรากฏขึ้น LED RUN(7)สีเขียวก็จะติดด้วย(ติดเฉพาะใน Manual Mode) แสดงว่าบอร์ดพร้อมเล่นไฟล์เพลงแล้ว ให้ผู้ใช้กด SW. PLAY /STOP เพื่อทำการเล่นเพลง LED RUN สีเขียว(7)ก็จะกระพริบเมื่อมีการเล่น โดยลำดับเพลงที่จะนำมาเล่นนั้นจะเริ่มจาก ไฟล์แรกสุดที่ผู้ใช้ทำการ Copy ใส่ ลงใน SD/MMC Card ซึ่งจะเรียงลำดับการเล่นไปเรื่อยๆ

ถ้าดูการทำงานผ่าน RS232 เมื่อการ์ด Initial ผ่านแล้ว ที่หน้าต่างของ Procomm หรือ Hyperterminal จะแสดง ลำดับไฟล์,ชื่อ,นามสกุล และ ขนาดของไฟล์ MP3,WAV ที่มีอยู่ใน Card ออกมาให้เห็นทั้งหมด(เฉพาะไฟล์ที่อยู่นอกโฟลเดอร์) ถ้าเป็นไฟล์ นามสกุลอื่นๆ หรือไฟล์ที่อยู่ในโฟลเดอร์จะไม่ถูกแสดงให้เห็นและจะไม่ถูกนำมาเล่น

ในขณะที่เล่นเพลงอยู่จะแสดงเวลาในการเล่นออกที่จอ LCD และส่งเป็น Command (9 Byte)ออกมาทาง RS232/422 ด้วย โดย Timer Command มีรูปแบบดังนี้

@ xx = mn : ss <----------Timer Command 9 Byte

เมื่อ xx = ID ของบอร์ด แสดงเป็น ASCII 2 หลัก (00-31)

mn = เวลาเป็น นาที แสดงเป็น ASCII 2 หลัก (00-59)

ss = เวลาเป็น วินาที แสดงเป็น ASCII 2 หลัก (00-59)

เมื่อเล่นเพลงจบในแต่ละเพลงบอร์ดก็จะส่งพัลซ์ ‘0’ ออกมาที่ Connector EOF(22) และส่งเป็นCommand(5Byte) ออกมาทาง RS232/422 เพื่อแสดงว่าเพลงจบแล้ว โดย End of file Command มีรูปแบบดังนี้

$xx:E < ----------End of file Command 5 Byte

เมื่อ xx = ID ของบอร์ด แสดงเป็น ASCII 2 หลัก (00-31)

- หน้าที่และการใช้งาน SWITCH On Board .

ตารางที่5.1 แสดงหน้าที่ของสวิตช์ เมื่อ MD1 = 1

SWITCH	***FUNCTION***	***LCD DISPLAY***	***PROCOMM DISPLAY***
PLAY/STOP	ใช้เมื่อต้องการ เล่นเพลง หรือหยุดเล่นชั่วขณะ โดยการกดซ้ำเพื่อเปลี่ยนการทำงาน	แสดงชื่อไฟล์เมื่อเริ่มเล่น และสถานะ ของ SW. คือ (PLAY=> หรือ PAUSE □)	แสดงชื่อไฟล์เมื่อเริ่มเล่น ตอนแรกเท่านั้น
VOL(+)	เพิ่มระดับความดังเสียง	VOL.= OFF-20	00-20<=Volume
VOL(-)	ลดระดับความดังเสียง	VOL.=20-OFF	20-00<=Volume
FWD	เลื่อนเพลงไปข้างหน้า 1 เพลง	NEXT -->	ชื่อเพลงเปลี่ยนเป็นเพลงใหม่
REW	เลื่อนเพลงถอยหลัง 1เพลง โดยจะต้องกด SW.ภายใน 5 วินาที เมื่อเพลงเริ่มเล่น เพลงจึงจะถอยหลัง ถ้าเกินจากนี้จะเป็นการกลับมาเริ่มเล่นที่ต้นเพลงของเพลงนั้นอีกครั้ง	BACK <--	ชื่อเพลงเปลี่ยนเป็นเพลงใหม่

ตารางที่5.2 แสดงหน้าที่ของสวิตช์ เมื่อ MD1 = 0

SWITCH	*FUNCTION*	*LCD DISPLAY*	*PROCOMM DISPLAY*
PLAY/STOP	ใช้เมื่อต้องการ เล่นเพลง หรือ ใช้เป็น Key Shift โดยใช้กดร่วมกับ SW.อื่นๆอีก 4 ตัว	แสดงชื่อไฟล์เมื่อเริ่มเล่นและ สถานะ ของ SW. (PLAY=>)	แสดงชื่อไฟล์เมื่อเริ่มเล่นตอนแรกเท่านั้น
VOL(+)	เพิ่มระดับความดังเสียง	VOL.= OFF-20	00-20<=Volume
VOL(-)	ลดระดับความดังเสียง	VOL.=20-OFF	20-00<=Volume
FWD	เลื่อนเพลงไปข้างหน้า 1 เพลง	NEXT -->	ชื่อเพลงเปลี่ยนเป็นเพลงใหม่
REW	เลื่อนเพลงถอยหลัง 1เพลง โดยจะต้องกด SW.ภายใน 5 วินาที เมื่อเพลงเริ่มเล่น เพลงถึงจะถอยกลับให้ ถ้าเกินจากนี้ จะเป็นการกลับมาเริ่มเล่นที่ต้นเพลงของเพลงนั้นอีกครั้ง	BACK <--	ชื่อเพลงเปลี่ยนเป็นเพลงใหม่
SHIFT+VOL(+)	เพิ่มเสียงเบส(จะตอบสนองเสียงเบสได้เต็มที่เมื่อ Set ระดับ Vol. ไม่เกินระดับ14)	BAS+=OFF-15	-
SHIFT+VOL(-)	ลดเสียงเบส	BAS-=15-OFF	-
SHIFT+FWD	เพิ่ม Delay ให้กับไฟวิ่ง VU (PORT C)	Dlay=01-35	-
SHIFT+REW	ลด Delay ให้กับไฟวิ่ง VU (PORT C)	Dlay=35-01	-
SHIFT+VOL(-) + REW	หยุดเล่นเพลงชั่วขณะ เมื่อต้องการจะกลับมาเล่นก็ให้กด SW. PLAY/STOP เพียงปุ่มเดียว	PAUSE □	-
SHIFT+FWD+ REW	ใช้เปลี่ยนรูปแบบไฟวิ่ง VU ระหว่าง MOD1 หรือ MOD2 โดยการกด SW. ทั้ง 3 ซ้ำเพื่อเปลี่ยนโหมด	VU-MOD1 หรือ VU-MOD2	-

หมายเหตุ ในการใช้งาน SW. มากกว่า 1 SW. ให้กด key shift (PLAY/STOP) ค้างไว้ก่อนแล้วจึงกด SW. ที่เหลือซึ่งจะเป็น SW.อะไรก่อนก็ได้ตามไป ถึงจะมีผลใช้งานได้ รูปแบบของสวิตช์ทุกหน้าที่การใช้งาน จะเป็นแบบ กดปล่อย ถ้ากดแช่จะมีผลครั้งเดียว ต้องปล่อยก่อนแล้วกดใหม่จึงจะมีผลอีก

ในระหว่างที่เล่นเพลงอยู่นั้นเมื่อผู้ใช้กด SW. PLAY/STOP(MD1=1) หรือ SHIFT +VOL(-)+REW (MD1=0) เพื่อหยุดการเล่นชั่วขณะ ผู้ใช้จะสามารถเข้ามากำหนดรูปแบบการเล่นเพลงได้อีก 4 รูปแบบด้วย SW. ดังตาราง5.3

ตารางที่5.3 แสดงหน้าที่ของสวิตช์เมื่ออยู่ในสภาวะหยุดเล่นเพลงชั่วขณะ (PAUSE)

SWITCH	*FUNCTION*	*LCD DISPLAY*
VOL+	เล่นเพลงซ้ำ 1 เพลง ไปเรื่อยๆ	R1
VOL-	เล่นเพลง 1 เพลง แล้วจบ	x1
RR	เล่นเพลงทีมีใน SD/MMC ทั้งหมด 1 รอบแล้วจบ	-->
FF	เล่นเพลงที่มีใน SD/MMC ซ้ำทั้งหมด ไปเรื่อยๆ	<-->

- *ความหมายของข้อความที่แสดงบนจอ LCD*

#Baud Rate= = อัตราการรับส่งข้อมูล สำหรับสื่อสารผ่าน RS232/422/485 (กำหนดที่ BD0-BD2)

#ID= = ID ของบอร์ด = 00-31 (กำหนดที่ ID0-ID2,MD0-MD1)

#Device=0 = หมายเลขชนิดของอุปกรณ์ โดยบอร์ด MP3. นี้จะกำหนดไว้ = 0 เสมอ

ET-REMOTE MP3 V1 = ชื่อรุ่นของบอร์ดที่ใช้งาน

Error! = ข้อความนี้จะกระพริบเมื่อไม่มี SD/MMC Card ใส่ไว้ที่ Socket

CARD OK. = สามารถทำการ Initial Card ผ่าน

Card Fail! Reset = ไม่สามารถ Initial Card ได้ ให้ทำการ Reset MCU ใหม่

BPB Error! Reset = ไม่สามารถอ่านข้อมูลจากตำแหน่ง Boot Record ของ Card ได้ ให้ Reset MCU ใหม่

File Not Found = ไม่พบไฟล์ MP3 หรือ WAV ใน Card

PL-MANUAL MODE. = เข้าสู่โหมดการทำงาน MANUAL MODE

PL-CONTROL MODE. = เข้าสู่โหมดการทำงาน CONTROL MODE

ET-MP3.OK. = บอร์ดอยู่ในสภาวะพร้อมรับคำสั่งเพื่อทำงาน

5.2 การใช้งานในโหมด Control (PLAY UART MODE)

การใช้งานในโหมดนี้จะต้อง Set Jumper มาทางด้าน Control ซึ่งในโหมดนี้จะใช้วิธีการควบคุมการเล่นของเพลงโดยการส่งเป็น Command จากภายนอกเข้ามาผ่านทาง RS232/RS422 และ RS485แบบ 4 เส้น ซึ่งการส่ง Command นั้นจะใช้คอนโทรลเลอร์ตระกูลใดๆเป็นตัวส่งก็ได้ หรือจะส่งผ่านทาง Hyperterminal หรือ Procomm ด้วยวิธีการพิมพ์จาก Keyboard ก็ได้ เมื่อผู้ใช้ทำการส่ง Command ให้กับบอร์ด ตัวบอร์ดก็จะยิง Echo ของ Command นั้นๆกลับออกมาให้เมื่อ Command ที่ส่งนั้นถูกต้อง ในกรณีที่Set บอร์ดให้ทำงานในโหมด RS485 นั้น ตัวบอร์ดจะส่งข้อความ Respond ออกมา 3 อย่างคือ Echo Command ,Timer Command(ถ้า Enable) และ End of file Command เท่านั้น ส่วนข้อความอื่นๆหรือรายชื่อ List File จะไม่ถูกพิมพ์ออกมาให้เห็นทาง Procomm หรือ Hyperterminal เหมือนกับการใช้ RS232/422 ในการสื่อสาร

Baud Rate ในการรับส่งข้อมูลสามารถกำหนดได้จาก Jumper BD0-BD2 บนบอร์ด ซึ่งดูการ Set ได้จากตารางที่3.2 ค่า Baud Rate และ ID ที่ถูก Set จะแสดงทาง LCD และ Procomm ทุกครั้งเมื่อบอร์ด MP3. ถูก Reset .

รูปแบบและการใช้งานของ Command ต่างๆ ที่บอร์ด ET-REMOTE MP3.รองรับ

รูปแบบของ Command ที่ใช้นั้นจะอยู่ในรูปของ ASCII Code ทั้งหมด ซึ่งสามารถแทนด้วยสัญลักษณ์เช่น '*' หรือแทนด้วย Code ซึ่งก็จะเท่ากับ 0x2A , '0'=0x30 , 'A' = 0x41 เป็นต้น ในกรณีที่ส่งเป็นตัวอักษร A-Z สามารถส่งเป็นตัวพิมพ์เล็กหรือพิมพ์ใหญ่ก็ได้ ซึ่งตัวบอร์ดจะทำการแปลงเป็นตัวพิมพ์ใหญ่ให้เองอัตโนมัติ

สำหรับตัวอย่างการส่ง Command ที่ให้มานั้นจะใช้ภาษา C ซึ่งสนับสนุน MCU Z8Encore! ซึ่งผู้ใช้สามารถนำตัวอย่างที่ให้ ไปดัดแปลงใช้งานกับ MCU เบอร์อื่นๆได้โดยไม่ยาก

COMMAND '00' (PLAY by Name)

คำสั่ง เล่นเพลงตามชื่อไฟล์ ซึ่งจะเล่นเพลงที่ส่งไปทันที สามารถเลือกเพลงต่อกันได้ประมาณ 50 เพลง(ขึ้นอยู่กับความยาวของชื่อไฟล์ ถ้าชื่อไฟล์สั้นก็จะได้มากกว่านี้แต่ไม่เกิน 110ไฟล์) ต่อ 1 คำสั่ง ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	File1 .MP3 /WAV	Mark4	File50.MP3 /WAV	Mark5	END
1 Byte	1 Byte	1Byte	2 Byte	1 Byte	2 Byte	1 Byte	-	1 Byte	-	1 Byte	1 Byte
*	0	:	00-31	:	00	=	xxx.MP3	;	xxx.MP3	;	0x0D (enter)
Echo Command จากบอร์ด											
#	0	:	00-31	:	R00	=	PFN				

-PFN = Play File by name (now)

COMMAND '01' (PLAY by Name)

คำสั่งเล่นเพลงตามชื่อไฟล์ ซึ่งจะรอจนกว่าเพลงที่เล่นอยู่ในขณะนั้นจบสมบูรณ์เสียก่อน แล้วถึงจะเล่นเพลงที่ส่งด้วยคำสั่งนี้ให้ สามารถเลือกเพลงต่อกันได้ประมาณ 50 เพลง (ขึ้นอยู่กับความยาวของชื่อไฟล์ ถ้าชื่อไฟล์สั้นก็จะได้มากกว่านี้แต่ไม่เกิน 110ไฟล์) ต่อ 1 คำสั่ง ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	File1 .MP3 /WAV	Mark4	File50.MP3 /WAV	Mark5	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	-	1 Byte	-	1 Byte	1 Byte
*	0	:	00-31	:	01	=	xxx.MP3	;	xxx.MP3	;	0x0D (enter)
Echo Command จากบอร์ด											
#	0	:	00-31	:	R01	=	PFW				

-PFW = Play File by name (wait end File)

-Echo Command ที่ตอบกลับจากบอร์ดจะเป็น ASCII Code ทุกตัว(13Byte)

-xxx = ชื่อไฟล์ ASCII ไม่เกิน 8 byte/1 File (8Charecter/1File)

Ex.1 การส่งคำสั่ง Play ด้วย Command '00' กำหนดให้ ID = 02

```
#include <stdio.h>
#include  <ez8.h>
main()
{
  unsigned char enter = 0x0D ;
  printf("*0:02:00 =SONG01.MP3;SONG02.MP3;SONG03.WAV",enter) ;   // Sent Command 00
 while(1){;}
}
```

ตัวอย่างที่1 นี้จะเป็นการส่งคำสั่ง '00' ให้เริ่มเล่นเพลงที่ชื่อ SONG01.MP3 ตามด้วย SONG02.MP3 และเพลงสุดท้ายคือ SONG03.WAV หลังจากส่งคำสั่งเสร็จโปรแกรมจะมาวนหยุดอยู่ที่ while ส่วนตัวบอร์ด ET-REMOTE MP3 ก็จะทำการเล่นเพลงที่ส่งมาทันทีจนครบทุกเพลงที่ส่งไป ไม่ว่าก่อนหน้านี้จะเล่นหรือไม่ได้เล่นเพลงใดๆอยู่ก็ตามคำสั่งจะถูกกระทำทันที

Ex.2 การส่งคำสั่ง Play ด้วย Command '01' กำหนดให้ ID = 01

```
#include <stdio.h>
#include  <ez8.h>
main()
{   unsigned char enter = 0x0D ;
    printf("*0:01:01 =SONG02.MP3;SONG01.MP3;SONG03.WAV",enter) ;     // Sent Command 01
    while(1){;}
}
```

ตัวอย่างที่2 นี้จะส่งคำสั่ง '01' ให้เริ่มเล่นเพลงที่ชื่อ SONG02.MP3 ก่อน ตามด้วย SONG01.MP3 และเพลงสุดท้ายคือ SONG03.WAV หลังจากส่งคำสั่งเสร็จโปรแกรมจะมาวนหยุดอยู่ที่ while ส่วนตัวบอร์ด ET-REMOTE MP3. ก็จะทำการเล่นเพลงที่ส่งมาหลังจากเพลงที่เล่นอยู่ก่อนหน้าจบลง หรือเริ่มเล่นทันทีถ้าไม่มีการเล่นเพลงใดๆอยู่ก่อนหน้านี้ จนครบทุกเพลง

การดูชื่อไฟล์เพื่อจะนำมาใส่ใน Command นั้นถ้าชื่อไฟล์ที่เราตั้งขึ้นยาวไม่เกิน 8 ตัวอักษร สามารถนำชื่อที่เราตั้งมาใส่ได้เลยแต่ถ้าเกิน 8 ตัวอักษรจะต้องดูรายชื่อไฟล์จาก List File ที่ตัวบอร์ดส่งผ่านออกมาทาง RS232/422 ซึ่งจะแสดงให้เห็นโดยใช้ Hyperterminal หรือ Procomm เป็นตัวแสดงผล หลังจาก Reset Board ทุกครั้ง ซึ่งจะแนะนำวิธีการดูในท้ายคู่มือ

COMMAND '02' (PLAY Auto)

คำสั่ง เล่นเพลงแบบอัตโนมัติ ซึ่งจะเล่นเพลงที่มีอยู่ใน SD/MMC Card ทั้งหมด โดยจะเล่นทันทีที่ส่งคำสั่งไป ไม่ว่าในขณะนั้นบอร์ดจะเล่นเพลงใด หรือไม่ได้เล่นเพลงใดๆ อยู่ก็ตาม ซึ่งเพลงแรกที่ถูกเล่นก็คือเพลงแรกที่ผู้ใช้ Copy ลงใน SD/MMC Card เรียงกันไปตามลำดับจนครบทุกเพลงที่มีอยู่ใน SD/MMC Card ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	1 Byte
*	0	:	00-31	:	02	=	0x0D (enter)
Echo Command จากบอร์ด							
#	0	:	00-31	:	R02	=	PAN

- PAN = Play File Auto (now)

Ex.3 การส่งคำสั่ง Play ด้วย Command '02' แบบเล่นอัตโนมัติ กำหนดให้ ID = 03

```
#include <stdio.h>
#include  <ez8.h>
main()
 {
   unsigned char enter = 0x0D ;
   printf("*0:03:02 =",enter) ;       // Sent Command 02
  while(1){;}
}
```

ตัวอย่างที่3 นี้จะเป็นการส่งคำสั่ง '02' ให้กับบอร์ด ET-REMOTE MP3. เมื่อบอร์ดได้รับคำสั่งนี้ ก็จะทำการเริ่มเล่นเพลงที่มีอยู่ใน SD/MMC Card ทันทีไม่ว่าก่อนหน้านี้จะเล่นเพลงอะไรค้างอยู่ก็ตามจะไม่สนใจ โดยเพลงที่เริ่มเล่นนั้นตัวบอร์ดจะทำการเล่นเพลงที่ถูกเก็บไว้ในแอดเดรสเริ่มต้นของ SD/MMC Card ก่อนเสมอและเรียงไปจนครบทุกเพลง

COMMAND '02' (PLAY by Number)

คำสั่งเล่นเพลงตามหมายเลขลำดับของเพลง โดยเพลงที่ส่งด้วยคำสั่งนี้จะถูกเล่นทันที สามารถเลือกหมายเลขลำดับเพลงให้เล่นต่อกันได้ประมาณ 110 เพลง ต่อ 1 คำสั่ง และหมายเลขลำดับเพลงอ้างได้ตั้งแต่ '001'-'999' (ASCII 3 Byte) ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	Number (1)	Mark4	Number (110)	Mark5	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	3 Byte	1 Byte	3 Byte	1 Byte	1 Byte
*	0	:	00-31	:	02	=	001-999	;	001-999	;	0x0D (enter)
Echo Command จากบอร์ด											
#	0	:	00-31	:	R02	=	PNN				

-PNN = Play file by Number (now) ; 001-999 คือ หมายเลขลำดับของเพลงที่แสดงใน List File

COMMAND '03' (PLAY by Number)

คำสั่งเล่นเพลงตามหมายเลขลำดับของเพลง โดยเพลงที่ส่งด้วยคำสั่งนี้จะถูกเล่นหลังจากเพลงที่เล่นอยู่ในขณะนั้นจบลงสมบูรณ์เสียก่อน หรือ เริ่มเล่นทันทีถ้าไม่มีการเล่นเพลงใดๆอยู่ก่อน สามารถเลือกหมายเลขลำดับเพลงให้เล่นต่อกันได้ประมาณ 110 เพลง ต่อ 1 คำสั่ง และหมายเลขลำดับเพลงสามารถอ้างได้ตั้งแต่ '001'-'999' (ASCII 3 Byte) ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	Number (1)	Mark4	Number (110)	Mark5	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	3 Byte	1 Byte	3 Byte	1 Byte	1 Byte
*	0	:	00-31	:	03	=	001-999	;	001-999	;	0x0D (enter)
Echo Command จากบอร์ด											
#	0	:	00-31	:	R03	=	PNW				

-PNW = Play file by number (wait) ; 001-999 คือ หมายเลขลำดับของเพลงที่แสดงใน List File

การดูหมายเลขลำดับเพลงสามารถดูได้จาก List File ที่ตัวบอร์ดส่งผ่านออกทาง RS232/422 ซึ่งจะแสดงให้เห็นโดยใช้ Hyperterminal หรือ Procomm เป็นตัวแสดงผลหลังจาก Reset Board ทุกครั้ง ซึ่งจะแนะนำวิธีการดูในท้ายคู่มือ

Ex.4 การส่งคำสั่ง Play ด้วย Command '02' แบบเล่นตามหมายเลขลำดับเพลง กำหนดให้ ID = 07

```
#include <stdio.h>
#include  <ez8.h>
main()
 {   unsigned char enter = 0x0D ;
     printf("*0:07:02 =002;044;250;",enter) ;      // Sent Command 02
     while(1){;}
 }
```

ตัวอย่างที่4 นี้จะเป็นการส่งคำสั่ง '02' ให้กับบอร์ด ET-REMOTE MP3. เมื่อบอร์ดได้รับคำสั่งนี้ ก็จะทำการเริ่มเล่นเพลงตามหมายเลขเพลงที่ส่งไปทันที ไม่ว่าก่อนหน้านี้จะเล่นเพลงอะไรค้างอยู่ก็ตามจะไม่สนใจ โดยจะเริ่มเล่นเพลงในลำดับ 002 ก่อน ตามด้วย 044 และ 250 เป็นเพลงสุดท้าย

Ex.5 การส่งคำสั่ง Play ด้วย Command '03' กำหนดให้ ID = 07

```
#include <stdio.h>
#include  <ez8.h>
 main()
   {   unsigned char enter = 0x0D ;
       printf("*0:07:03 =210;",enter) ;       // Sent Command 03
       while(1){;}
   }
```

ตัวอย่างที่5 นี้จะเป็นการส่งคำสั่ง '03' ให้กับบอร์ด ET-REMOTE MP3. เมื่อบอร์ดได้รับคำสั่ง จะยังไม่เล่นเพลงที่ส่งออกไปทันที แต่จะรอจนกว่าเพลงที่บอร์ดเล่นอยู่ในขณะนั้นจบลงเสียก่อนถึงจะเริ่มเล่นเพลงลำดับที่ 210 หรือ ถ้าไม่มีการเล่นเพลงใดๆอยู่ก่อน เพลงที่ 210 ก็จะถูกเล่นทันทีหลังจากที่ได้รับคำสั่ง

ในส่วนของคำสั่ง Play ที่กล่าวไปข้างต้นทั้งหมดนั้นในการส่งคำสั่งไปแต่ละครั้งตัวบอร์ด ET-REMOTE MP3 จะตอบสนองต่อคำสั่งที่ส่งออกไปล่าสุดเสมอ จะไม่จำคำสั่งเดิมก่อนหน้าไว้ ไม่ว่าจะเล่นคำสั่งก่อนหน้าสมบูรณ์หรือยังก็ตาม เช่น ถ้าผู้ใช้ส่งคำสั่ง Play ให้เล่นเพลงตามที่กำหนดเรียงกัน 3 เพลง ระหว่างที่เล่นเพลงใดเพลงหนึ่งอยู่นั้น ผู้ใช้เกิดส่งคำสั่ง Play ให้เล่นเพลงอีกชุดหนึ่ง ตัวบอร์ดก็จะทำการเล่นเพลงของคำสั่งล่าสุดทันที (Command 00,02) หรือรอให้เพลงที่เล่นอยู่ปัจจุบันจบก่อนแล้วจึงทำการเล่นเพลงของคำสั่งล่าสุด(Command 01,03) โดยจะไม่สนใจว่าจะเล่นเพลงของคำสั่งก่อนหน้าครบทุกเพลงหรือไม่

ในกรณีที่ผู้ใช้ส่งคำสั่งให้มีการเล่นซ้ำ เมื่อบอร์ดเล่นครบทุกเพลงที่รับเข้ามาแล้วบอร์ดก็จะวนกลับมาเริ่มต้นเล่นในเพลงแรกที่ส่งเข้ามาอีก เป็นลักษณะนี้ไปเรื่อยๆ จะสังเกตเห็นว่าในการส่งคำสั่ง Play ออกไปแต่ละครั้ง ในส่วนของชื่อไฟล์หรือหมายเลขลำดับไฟล์แต่ละไฟล์จะต้องถูกปิดด้วยเครื่องหมาย ' ; ' เสมอ และสิ้นสุดด้วย 0x0D เมื่อจบ Command นั้นๆ ซึ่งจะใส่นอกเหนือจากนี้ไม่ได้ มิฉะนั้นจะทำให้ Command นั้นๆ ไม่ทำงานตามที่เราต้องการได้

เมื่อสั่ง Play ด้วย Command ใด Command หนึ่ง ใน 5 Command นี้(Command 00,01,02 ,03) ถ้าบอร์ดเล่นเพลงจบในแต่ละเพลงที่ส่งมาอย่างสมบูรณ์ บอร์ดก็จะส่ง Response Command End of File ออกมาทาง RS232/422/485 ให้ รวมทั้งจะส่งพัลล์ลอจิก '0' 1 ลูก ออกมาที่ขั้วต่อ EOF ที่อยู่บนบอร์ดด้วย ถ้ามีการเปลี่ยนเพลงขณะที่ยังเล่นไม่จบเพลง เช่น มีการใช้คำสั่ง Forward ,Reverse หรือ สั่ง Play ด้วยคำสั่งที่มีการเล่นทันที(Command 00,02) จะไม่มีการส่ง End of File ออกมาให้

รูปแบบ Response End of File Command เป็นดังนี้ :

$xx:E < ---------- End of file Command 5 Byte

เมื่อ xx = ID ของบอร์ด แสดงเป็น ASCII 2 หลัก (00-31)

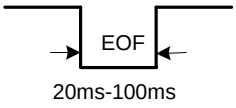
Ex.6 ตัวอย่างการตรวจสอบ End of File ที่ส่งมาจากบอร์ด ET-REMOTE MP3. ผ่านทาง RS232/422/485 สมมุติให้ ID ของบอร์ด =01

```
        //---------------- Respond End song Main ----------------
void EOF_M(char id_H,char id_L)
 {
    unsigned char buf_eof [4] ;
    unsigned char ch , e ;
    id_H = id_H+0x30 ;          // Convert ID ที่ส่งผ่านเข้ามาให้เป็น ASCII Code
    id_L = id_L+0x30  ;
    do{
       do{                      // Check Byte Start
          Rx0();
          ch = U0D ;           // Recive data from ET-REMOTE MP3
         }while(ch !='$') ;    // Check Byte start ของ EOF Command ('$'=0x24);
      for(e=0;e<4 ; e++)      //รับ EOF Command อีก 4 Byte ที่เหลือ
        {
          Rx0();
          buf_eof[e] = U0D ;
        }
     }while((buf_eof[0] != id_H )&& buf_eof[1] != id_L)&& (buf_eof[3] != 'E'));
                                        //ตรวจสอบ EOFว่าเป็นของบอร์ด ID ที่เราต้องการหรือไม่
 }
```

จากตัวอย่างนี้จะเป็นฟังก์ชันย่อยสามารถนำไปเขียนและเรียกใช้งานที่โปรแกรมหลักได้ โดยการเรียกใช้ก็อ้างเพียง EOF_M(0, 1) ; เท่านั้น โดยค่า id_H และ id_L ที่รับเข้ามาเป็นฐาน 10 หรือ 16 ก็ได้ จะเห็นว่าในโปรแกรมย่อยจะทำการ

Convert ให้เป็น ASCII เอง เพื่อจะนำไปเปรียบเทียบกับค่า ID ที่เป็น ASCII ที่ได้รับมาจากบอร์ด MP3. ซึ่งจะถูกเก็บอยู่ในตัวแปร buf_eof[0] และ buf_eof[1] ตามลำดับ ในการทำงานของโปรแกรมตรวจสอบ EOF นี้เริ่มต้นจะเช็ค Byte Start ที่ส่งมาจากบอร์ด MP3 ว่าใช่ '$' หรือไม่ ถ้าใช่ถึงจะรับข้อมูลใน Byte ที่เหลือ เมื่อรับข้อมูลครบ ก็จะตรวจสอบว่าเป็น Command EOF ที่ส่งมาจากบอร์ด ID ที่เราต้องการหรือไม่ ถ้าใช่ก็จะออกจาก Loop ได้ ถ้าไม่ใช่ ก็จะกลับไปรับข้อมูลมาตรวจสอบอีก ในกรณีที่ใช้เพียงบอร์ด MP3 บอร์ดเดียว ในคำสั่งwhile อาจจะเช็คตัวอักษร E อย่างเดียวก็ได้(สื่อสารด้วย RS232/422) ถ้ามีการใช้หลายบอร์ดควรจะ Set การสื่อสารของบอร์ด MP3.แบบ RS485 และตรวจสอบ ID ด้วย ส่วนฟังก์ชัน Rx0() นั้นจะเป็นฟังก์ชัน ตรวจสอบความพร้อมของบัพเฟอร์ในการรับข้อมูลเข้ามา ซึ่งดูได้ในตัวอย่างของการประยุกต์ใช้งาน โดย MCU แต่ละตัวจะมีวิธีการตรวจสอบความพร้อมของบัพเฟอร์ต่างกันไป

นอกจากจะตรวจสอบ EOF ที่ส่งมาในรูปแบบ Command ผ่านทาง RS232/422 แล้ว ยังสามารถตรวจสอบจากสัญญาณพัลซ์ที่ส่งออกมาที่ขั้ว EOF บนบอร์ดก็ได้ ซึ่งจะส่ง Logic '0' ออกมาเมื่อเกิดสัญญาณ EOF ขึ้น โดยสัญญาณ EOF นี้จะมีความกว้างอยู่ที่ประมาณ 20ms-100 ms ซึ่งขึ้นอยู่กับคำสั่ง Play ที่ใช้เล่น และสามารถเขียนโปรแกรมตรวจสอบได้ ดังแสดงในตัวอย่างที่7



รูปแสดง สัญญาณพัลซ์ EOF

Ex.7 ตัวอย่างการตรวจสอบสัญญาณพัลซ์ End of file (EOF) ที่ส่งมาจากบอร์ด MP3 ซึ่งตัวอย่างนี้ จะกำหนดให้รับสัญญาณเข้ามาทางขา PC0 ของ MCU Z8Encore!

```
//------------------------------- Respond EOF Pulse ---------------------------------
void EOF_Pul()
 {
   unsigned char ch ;
   do{
       ch  = PCIN ;          //รับสัญญาณเข้ามาทาง PC0
       ch &= 0x01 ;          // Clear บิตที่ไม่ใช้งาน
     }while(ch != 0x00);    //ถ้า ch = 1 ยังไม่มี EOF ส่งมากลับไปตรวจสอบต่อ
 }
```

ในตัวอย่างนี้เริ่มต้นจะอ่านสัญญาณเข้ามาทาง PC0 ถ้าไม่มีสัญญาณ EOF ที่ขานี้จะเป็น 1 เสมอ ทำให้อ่านค่าเข้ามาเก็บไว้ในตัวแปร ch เป็น 1 เสมอ แต่เมื่อมีสัญญาณ EOF ส่งมา ค่าในตัวแปรก็จะกลายเป็น 0 และออกจาก Loop ได้

COMMAND '04' (STOP)

เป็นคำสั่ง STOP การเล่น เพื่อหยุดเล่นเพลง ซึ่งจะออกจากโหมดการเล่นเพลงทันที เพื่อมาอยู่ในสภาวะเริ่มต้น พร้อมรับคำสั่งอื่นๆต่อไป ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	1 Byte
*	0	:	00-31	:	04	=	0x0D (enter)
Echo Command จากบอร์ด							
#	0	:	00-31	:	R04	=	STP

- STP = Stop

COMMAND '05' (PAUSE)

เป็นคำสั่ง หยุดการเล่นเพลงไว้ชั่วขณะ(PAUSE) เมื่อต้องการให้กลับมาเล่นเพลงในตำแหน่งที่หยุดไว้อีกครั้งหนึ่ง ก็ให้ส่งคำสั่งนี้ซ้ำอีกครั้ง ในขณะที่หยุดเล่นเพลงไว้ชั่วขณะนี้ ถ้าผู้ใช้ส่ง Command Play(00,01,02,03) , STOP(04) , Forward(06) ,Reverse(07) ออกมา จะทำให้เพลงที่เล่นค้างอยู่นั้นถูกเปลี่ยนแปลงไป ตาม Command ที่ส่งมา เมื่อส่ง Command นี้ซ้ำอีกครั้งหนึ่งเพื่อจะเล่นต่อ จะไม่สามารถกลับมาเล่นเพลงต่อจากตำแหน่งเพลงเดิมที่หยุดไว้ได้อย่างถูกต้อง มีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	1 Byte
*	0	:	00-31	:	05	=	0x0D (enter)
Echo Command จากบอร์ด							
#	0	:	00-31	:	R05	=	PAU หรือ PAY

- PAU = Pause : หยุดชั่วขณะ ; -PAY = Play : กลับมาเล่นต่อจากเดิม

Ex.8 การส่งคำสั่ง STOP และ PAUSE ด้วย Command '04' และ '05' ตามลำดับ กำหนดให้ ID = 01

```
#include <stdio.h>
#include  <ez8.h>
main()
{  unsigned char enter = 0x0D ;
   printf("*0:01:04 =",enter)   ;      // Sent Command 04   STOP
   delay(300)                   ;
   printf("*0:01:05 =",enter)   ;      // Sent Command 05  PAUSE
}
```

COMMAND '04' (Insert Song by File Name)

เป็นคำสั่งแทรกเพลงโดยส่งเป็นชื่อไฟล์และนามสกุลของไฟล์ที่ต้องการจะแทรก เพลงที่ส่งมาแทรกนั้นสามารถส่งมาให้เล่นต่อกันได้ถึง 50 เพลง (ขึ้นอยู่กับความยาวของชื่อไฟล์ ถ้าชื่อไฟล์สั้นก็จะได้มากกว่านี้แต่ไม่เกิน 110ไฟล์) โดย Command นี้จะใช้ Command เดียวกับคำสั่ง STOP เพียงแต่ ถ้าพิมพ์ชื่อไฟล์และนามสกุลของเพลงนั้นต่อท้ายเครื่องหมาย = ก่อนที่จะจบด้วย 0x0D ก็จะเป็นการใช้งานคำสั่งนี้แทน รูปแบบคำสั่งแสดงในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	File1 .MP3 /WAV	Mark4	File50.MP3 /WAV	Mark5	END
1 Byte	1 Byte	1Byte	2 Byte	1 Byte	2 Byte	1 Byte	-	1 Byte	-	1 Byte	1 Byte
*	0	:	00-31	:	04	=	xxx.MP3	;	xxx.MP3	;	0x0D (enter)
Echo Command จากบอร์ด											
#	0	:	00-31	:	R04	=	INF				

-INF = Insert song by File name ; xxx = ชื่อไฟล์ไม่เกิน8 ตัวอักษร

COMMAND '05' (Insert Song by Number)

เป็นคำสั่งแทรกเพลงโดยจะส่งเป็นหมายเลขลำดับของเพลงที่ต้องการจะแทรก เพลงที่ส่งมาแทรกนี้สามารถเลือกหมายเลขลำดับเพลงให้เล่นต่อกันได้ประมาณ 110 เพลง ต่อ 1 คำสั่ง และหมายเลขลำดับเพลงอ้างได้ตั้งแต่ '001'-'999' (ASCII 3 Byte) ซึ่ง Command นี้จะใช้ Command เดียวกันกับคำสั่ง PAUSE จะต่างตรงที่ หลังเครื่องหมาย '=' จะตามด้วยลำดับของเพลงที่จะแทรก ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	Number (1)	Mark4	Number (110)	Mark5	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	3 Byte	1 Byte	3 Byte	1 Byte	1 Byte
*	0	:	00-31	:	05	=	001-999	;	001-999	;	0x0D (enter)
Echo Command จากบอร์ด											
#	0	:	00-31	:	R05	=	INN				

-INN = Insert song by Number ; 001-999 คือ หมายเลขลำดับของเพลงที่แสดงใน List File

การทำงานของคำสั่งแทรกเพลง ก็คือ ขณะที่เล่นเพลงใดๆอยู่ เมื่อบอร์ดได้รับคำสั่งนี้ เพลงที่เล่นอยู่จะถูกหยุดไว้ชั่วขณะ และเพลงที่ถูกส่งมาแทรกจะถูกเล่นทันที เมื่อบอร์ดเล่นเพลงที่แทรกจนครบทุกเพลงที่ส่งมา ก็จะกลับออกไปเล่นเพลงหลักที่ถูกพักไว้ในตอนแรกต่อจากตำแหน่งการเล่นที่จากมาให้ทันที ในระหว่างที่มีการเล่นเพลงที่ส่งมาแทรกอยู่นั้น บอร์ดจะไม่ตอบสนองต่อคำสั่งใดๆทั้งสิ้นที่ส่งเข้ามา จะต้องรอให้เพลงที่ส่งมาแทรกนั้นเล่นจบจนครบทุกเพลงที่ส่งมาเสียก่อน(ออกจากการแทรกเพลงแล้ว) ตัวบอร์ดถึงจะยอมรับคำสั่งได้ตามปกติ

ในการเล่นเพลงตามปกตินั้นเมื่อเล่นจบในแต่ละเพลงบอร์ดจะส่ง Command EOF และ พัลซ์ลอจิก 0 ออกมาให้ดังที่กล่าวไปแล้วข้างต้น ในส่วนของการแทรกเพลงด้วย Command 04 และ 05 นี้ก็เช่นกันบอร์ดก็จะส่ง Command EOF และ พัลซ์โลจิก '0' ออกมาให้เช่นกัน แต่จะส่งออกมาเพียงครั้งเดียวหลังจากเล่นเพลงที่ส่งมาแทรกจนครบทุกเพลงแล้ว ถึงจะส่ง EOF ออกมาต่อ 1 คำสั่งแทรก(04 หรือ 05) ก่อนที่จะกลับไปเล่นเพลงที่ค้างอยู่ต่อไป ซึ่งรูปแบบ Command แสดงดังนี้

รูปแบบ Response End of File (Insert) Command เป็นดังนี้ :

$xx:e < ---------- End of file(Insert) Command (5 Byte)

เมื่อ xx = ID ของบอร์ด แสดงเป็น ASCII 2 หลัก (00-31)

สำหรับตัวอย่างโปรแกรมการตรวจสอบ Response Command EOF ที่เกิดจากคำสั่งแทรกนั้นสามารถดูได้จากตัวอย่างที่ 6 และ 7 ซึ่งจะเหมือนกัน โดยในตัวอย่างที่ 6 ให้เปลี่ยนแค่ตัว 'E' ที่อยู่ในคำสั่ง while มาเป็นตัว 'e' แทนเท่านั้น

COMMAND '06' (Forward)

เป็นคำสั่งสำหรับเลื่อนเพลงไปข้างหน้าครั้งละ 1 เพลง(FWD) คือส่งคำสั่ง 1 ครั้งเพลงก็จะถูกเลื่อนไป 1 เพลง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	1 Byte
*	0	:	00-31	:	06	=	0x0D (enter)
Echo Command จากบอร์ด							
#	0	:	00-31	:	R06	=	NEX

- NEX = Next

COMMAND '07' (Reverse)

เป็นคำสั่งสำหรับเลื่อนเพลงถอยหลังครั้งละ 1 เพลง(REW) คือส่งคำสั่ง 1 ครั้งเพลงก็จะถอยกลับมา 1 เพลง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	1 Byte
*	0	:	00-31	:	07	=	0x0D (enter)
Echo Command จากบอร์ด							
#	0	:	00-31	:	R07	=	BAC

- BAC = Back

สำหรับคำสั่ง Forward หรือ Reverse นี้จะต้องส่งขณะที่มีการเล่นเพลงอยู่ถึงจะมีผลการเปลี่ยนแปลงให้เห็น เมื่อส่ง Command เลื่อนเพลงไปจนถึงเพลงสุดท้าย เพลงก็จะเลื่อนวนกลับมาเริ่มต้นที่เพลงแรกอีก การเลื่อนเพลงนี้จะเป็นในลักษณะ Close loop ไม่ว่าจะเป็นการเลื่อนไปข้างหน้า หรือ ถอยหลังก็ตาม

Ex.9 การส่งคำสั่ง Forward และ Reverse ด้วย Command '06' และ '07' ตามลำดับ กำหนดให้ ID = 01

```
#include <stdio.h>
#include  <ez8.h>
main()
 {   unsigned char enter = 0x0D ;
     printf("*0:01:06 =",enter)    ;      // Sent Command 06  เลื่อนเพลงไปข้างหน้า 1เพลง
     delay(300)                    ;
     printf("*0:01:07 =",enter)    ;      // Sent Command 07  เลื่อนเพลงถอยหลัง 1เพลง
 }
```


COMMAND '08' (Control Mode)

เป็นคำสั่งสำหรับควบคุมรูปแบบการเล่น และ การแสดงผลของเวลาในการเล่นออกทาง RS232/422/485 โดยจะมีอยู่ด้วยกัน 4 Command ย่อย ซึ่งจะใช้ตัวอักษร R,N,T,S สำหรับเป็นตัวแยกการทำงานของ Command 08 นี้ โดยถ้าหลังเครื่องหมาย '=' เป็น R (Repeat) หมายถึง เล่นเพลงที่ส่งมาซ้ำทั้งหมด เช่น ถ้ามีการส่งเพลงมาให้เล่น 5 เพลง เมื่อเล่นครบแล้วก็จะวนกลับมาเล่นเพลงแรกใหม่เป็นต้น

N (Normal) หมายถึง เล่นเพลงที่ส่งมาทั้งหมดเพียง 1 รอบแล้วจบ ซึ่งค่า default ของบอร์ดจะถูกกำหนดไว้ที่โหมดนี้เสมอ

T (Sent Time) หมายถึง ให้บอร์ด MP3 ส่งเวลาในการเล่นออกทาง RS232/422/485 โดยรูปแบบเวลาที่ส่งออกทาง RS232/422/485 นั้นจะเป็นลักษณะของ Command ASCII Code ซึ่งมีรูปแบบดังนี้(ดูตัวอย่างที่ 11 การรับ Command Play Time)

รูปแบบ Response Play Time Command เป็นดังนี้

@ xx = mm:ss < ---------- Play Time Command (9 Byte)

เมื่อ xx = ID ของบอร์ด แสดงเป็น ASCII 2 หลัก (00-31)

mm = เวลาเป็น นาที แสดงเป็น ASCII 2 หลัก (00-59)

ss = เวลาเป็น วินาที แสดงเป็น ASCII 2 หลัก (00-59)

S (Stop Sent Time) หมายถึง ไม่ให้บอร์ด MP3 ส่งเวลาในการเล่นออกไปที่ RS232/422/485 ซึ่งค่า default ของบอร์ดจะถูกกำหนดไว้ที่ค่านี้เสมอ

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	Sub-Command	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	1 Byte	1 Byte
*	0	:	00-31	:	08	=	R or N or T or S	0x0D (enter)
Echo Command จากบอร์ด								
#	0	:	00-31	:	R08	=	M:R or M:N or M:T or M:S	

- M:R = Mode Play Repeat ; M:N = Mode Play Normal

- M:T = Mode Sent Play Time on RS232/422/485 ; M:S = Mode Stop Sent Play Time on RS232/422/485

สำหรับ Command 08 นี้ สามารถส่งก่อนที่จะมีการสั่ง Play หรือส่งขณะที่ Play อยู่ก็ได้ คำสั่งจะมีผลเช่นเดียวกัน และค่าเริ่มต้นของบอร์ดนั้นจะถูกกำหนดไว้ที่ Mode Normal และ Mode Stop Sent Play Time เสมอ ถ้าผู้ใช้ต้องการให้ทำงานใน Mode Repeat และ ให้มีการส่งเวลาในการเล่นออกทาง RS232/422/485 ด้วยก็จะต้องส่ง Command 08 ที่มี Sub-Command เป็น R และ T ออกไปให้กับบอร์ด MP3 ซึ่งจะแสดงในตัวอย่างที่ 10 เวลาที่ส่งออกไปนั้นจะมีการเพิ่มขึ้น Step ละ 1 วินาที และสิ้นสุดตามความยาวของเพลงที่เล่น โดยเวลาจะแสดงได้ตั้งแต่ค่า 00:00 ถึง 59:59(0 นาที : 0 วินาที ถึง 59 นาที:59 วินาที)

Ex.10 การส่งคำสั่ง Control Mode ด้วย Command '08' เมื่อต้องการกำหนดให้มีการเล่นซ้ำเพลงทั้งหมด และให้ส่งเวลาในการเล่นเพลงออกทาง RS232/422/485 โดยกำหนดให้ ID = 01

```
#include <stdio.h>
#include  <ez8.h>
main()
 {  unsigned char enter = 0x0D  ;
    printf("*0:01:08 =R",enter)   ;      // Sent Command 08 Mode เล่นพลงซ้ำทั้งหมด
    delay(300)                    ;
    printf("*0:01:08 =T",enter)   ;      // Sent Command 08 Mode ส่งเวลาออกทาง RS232/422/485
 }
```

Ex.11 ตัวอย่างการรับ Command Play Time ที่ส่งมาจากบอร์ด ET-REMOTE MP3. ผ่านทาง RS232/422/485 สมมุติให้ ID ของบอร์ด =01

```
                    //----------------- Check Respond Timer Play  -----------------
void Chk_Time(char T_mH,char T_mL,char T_sH,char T_sL)
{  unsigned char buf_time[8] ;
    unsigned char t , ch ;
   T_mH  = T_mH + 0x30   ;          //Convert ASCII Minit ( mm)
   T_mL  = T_mL + 0x30   ;
  T_sH    = T_sH + 0x30    ;       //Convert ASCII Sec (ss)
  T_sL    = T_sL + 0x30    ;
   do{
       do{
            Rx0()                  ;
            ch = U0D               ;
           }while(ch != '@')       ;  // Check Mark  of  Timer ( '@'=0x40 )
        for(t=0;t<8;t++)              // if ch =@ then  Keep ID and  Time 8 byte ( x x =00:00)
           {
             Rx0()                 ;
             buf_time[t] = U0D     ;
           }
     }while((buf_time[3]!= T_mH)||(buf_time[4]!=T_mL)||(buf_time[6]!= T_sH)||(buf_time[7]!= T_sL));
}
```

สำหรับตัวอย่างที่ 11 นี้จะเป็นฟังก์ชันย่อยในการรับค่าเวลาการเล่นมาตรวจสอบว่าเพลงได้เล่นมาถึงค่าเวลาที่เราต้องการหรือยัง โดยรูปแบบการเรียกใช้งานก็เพียงแค่เรียกใช้ฟังก์ชันและกำหนดค่าเวลาที่ต้องการตรวจสอบดังนี้ สมมุติว่าต้องการเช็คเวลาการเล่นที่ 2นาที 30 วินาที ก็เรียกฟังก์ชันดังนี้ Chk_Time(0,2,3,0) ซึ่งการทำงานของตัวอย่างนี้จะรับค่าเวลาที่ผู้ใช้ส่งผ่านเข้ามาเป็น ฐานสิบ หรือ ฐานสิบหกก็ได้ จากนั้นก็จะนำค่านี้ไป Convert เป็น ASCII เก็บไว้ในตัวแปร T_mH,T_mL,T_sH,T_sL ต่อมาก็จะทำการรับ Byte Start ของ Command Play Time ซึ่งก็คือ '@' มาตรวจสอบถ้าถูกต้อง ก็ถึงจะทำการรับ Command Play time อีก 8 byte ที่เหลือเข้ามาเก็บไว้ยังตัวแปร buf_time[0-7] เพื่อนำไปเปรียบเทียบกับค่าของเวลาที่ผู้ใช้ได้กำหนดไว้จากการส่งผ่านค่าเข้ามาในฟังก์ชัน ว่าตรงกันหรือยัง ถ้ายังก็จะวนกลับไปรับค่าเวลามาเปรียบเทียบต่อจนกว่าจะเท่ากับค่าที่ผู้ใช้กำหนด ถึงจะออกจาก Loop while ได้ จากตัวอย่างนี้ค่า นาที ที่อ่านได้จากบอร์ด MP3 แต่ละครั้งจะถูกเก็บไว้ที่ buf_time[3-4] ส่วน วินาทีเก็บไว้ที่ buf_time[6-7] ส่วน ID จะถูกเก็บไว้ที่ buf_time[0-1] แต่ในตัวอย่างนี้ เราไม่ได้นำค่า ID มาตรวจสอบ ถ้าต้องการตรวจสอบ ID ผู้ใช้สามารถเพิ่มเงื่อนไขเข้าไปได้ในคำสั่ง while และเพิ่มตัวแปรที่ใช้สำหรับส่งผ่านค่า ID อีก 2 ตัวแปร ซึ่งการตรวจสอบ ID นี้ จะเหมาะกับการสื่อสารด้วย RS485

COMMAND '09' (VOLUME & BASS)

Command นี้จะใช้สำหรับปรับความดังของเสียง(Volume) และ เพิ่มระดับของ Bass โดยถ้าต้องการปรับความดังเสียง หลังเครื่องหมาย '=' ให้ตามด้วย ASCII (ตัวเลข) 3หลัก ตั้งแต่ 000 คือไม่มีเสียง ถึงค่า 255 คือเสียงดังสุด แต่ถ้าต้องการเพิ่มเสียง Bass หลังเครื่องหมาย '=' ให้ตามด้วย ASCII 3หลักโดยเริ่มตั้งแต่ B00 คือ Bass off ถึงค่า B15 คือ เสียง Bass สูงสุด ทั้งนี้เสียง Bass จะตอบสนองได้ดีก็ต่อเมื่อค่าของ Volume จะต้องปรับไม่เกินค่า 225 ถ้าเกินจากนี้ จะทำให้เสียง Bass ถูกขลิบ ซึ่งจะทำให้เสียง Bass นั้นลดลงไป รูปแบบคำสั่งแสดงดังในตาราง

Start	Device	Mark1	ID Board	Mark2	Command	Mark3	Sub-Command	END
1 Byte	1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	1 Byte	1 Byte
*	0	:	00-31	:	09	=	000-255 or B00-B15	0x0D (enter)
Echo Command จากบอร์ด								
#	0	:	00-31	:	R09	=	VOL or BAS	

- VOL = Volume ; BAS = Bass

Ex.12 การส่งคำสั่ง Volume & BASS ด้วย Command '09' เมื่อต้องการกำหนดให้ Volume = 220 และ BASS = 12 โดยกำหนดให้ ID = 01

```
#include <stdio.h>
#include  <ez8.h>
main()
{  unsigned char enter = 0x0D     ;
   printf("*0:01:09 =220",enter)   ;      // Sent Command 09 Volume=220
   delay(300)                      ;
   printf("*0:01:09 =B12",enter)   ;      // Sent Command 09 BASS = 12
}
```

ข้อควรจำ ในการเริ่มต้นส่ง Command ให้กับบอร์ด ET-MP3 นั้นเพื่อให้การรับ Command และ การส่ง Command สอดคล้องกัน(Sync) ในบอร์ดที่เป็นตัวส่ง Command ควรจะ delay ไว้ประมาณ 100 ms ขึ้นไปแล้วจึงทำการส่ง Command แรกให้กับบอร์ดได้ เพื่อทำให้ บอร์ด ET-REMOTE MP3. อยู่ในสถานะพร้อมรับข้อมูลก่อนเมื่อเริ่มต้น Power up และในกรณีที่ต้องการส่ง Command ติดกัน ควรจะ delay คั่นระหว่าง Command ด้วย ถ้าในระหว่าง Command ไม่ได้มีการตรวจสอบ Echo Command จากบอร์ด MP3 ที่ส่งกลับมาให้ เพื่อที่จะให้บอร์ด MP3 กระทำ Command แรกเรียบร้อยเสียก่อน โดยตัวอย่างโปรแกรมการตรวจสอบ Echo Command (00-09) จะแสดงในตัวอย่างที่ 13

Ex.13 เป็นตัวอย่างการตรวจสอบ Echo Command

```
                    //---------- Check Echo Command ----------
    void Echo_cmm()
     {
      unsigned char buf_cmm[13] ;
      unsigned char n,ch ;
      do{
          do{
              Rx0()           ;
              ch = U0D        ;
            }while(ch != '#')  ;      // Check Byte Start  of Echo ('#'=0x23)
         for(n=0;n<13;n++)            // if ch = #  then  Keep Echo Cmm 12 byte(0:ID:Rxx=xxx)
          {
            Rx0()                   ;
            buf_cmm[n] = U0D        ;
          }
       }while(buf_cmm[5]!='R')   ;      // ('R'=0x52)
     }
```

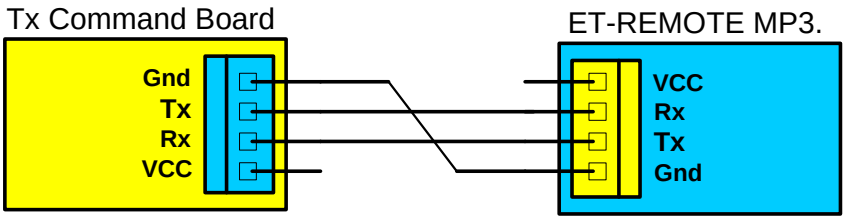
จากตัวอย่างที่13 จะเป็นฟังก์ชันย่อย ในการตรวจสอบ Echo Command แบบ ง่ายๆ คือ จะตรวจสอบเฉพาะตัวอักษร R เท่านั้น การเรียกใช้ก็เพียงเรียกฟังก์ชัน Echo_cmm() ; การทำงานของตัวอย่างนี้ เริ่มแรกก็จะทำการ Check Byte Start ก่อนว่าใช่ '#' หรือไม่ ถ้าใช่แสดงว่าเป็น Echo Command จากนั้นจึงทำการรับข้อมูลที่เหลืออีก 12 Byte ของ Echo Command มาเก็บไว้ในตัวแปร buf_cmm[0-11] แล้วนำค่าที่อยู่ใน buf_cmm[5] มาตรวจสอบว่า ใช่ตัว R หรือไม่ถ้าใช่ก็จะออกจาก Loop while จากนั้นผู้ใช้ก็จะสามารถส่ง Command ต่อไปได้อย่างถูกต้อง จะไม่ทำให้ Command นั้นทับกัน เนื่องจาก ถ้าบอร์ด MP3 ทำการส่ง Echo Command ออกมาให้นั่นแสดงว่าได้กระทำคำสั่งก่อนหน้านี้เรียบร้อยแล้ว

ในตัวอย่างนี้จะเห็นว่า Echo Command จะถูกรับเข้ามาเก็บไว้ยังตัวแปร buf_cmm[0-11] ทั้งหมด ดังนั้นในการตรวจสอบ Echo Command ผู้ใช้สามารถเพิ่มเงื่อนไขการตรวจสอบเข้าไปใน คำสั่ง while เพิ่มได้ อาจจะตรวจสอบ ID ด้วย หรือ ตรวจสอบชนิดของคำสั่งก็ได้ ขึ้นอยู่กับความจำเป็นของผู้ใช้ในการใช้งาน

5.3 การใช้งาน RS232 / RS422 /RS485

5.3.1 การส่ง Command ผ่าน RS232

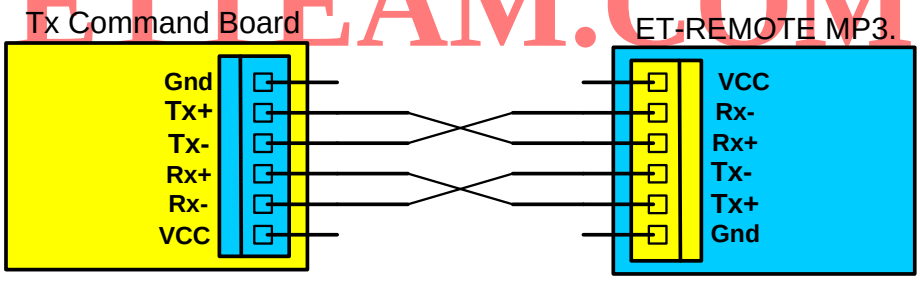
ในการส่ง Command ด้วยการสื่อสารแบบ RS232 นี้เหมาะสำหรับใช้ควบคุมบอร์ด MP3 เพียงบอร์ดเดียว เพราะถ้าพ่วงกันหลายบอร์ดจะทำให้การตรวจสอบ Respond จากบอร์ด MP3. นั้นผิดพลาดได้ ในการใช้งานจะต้องใส่ IC Line Driver Max232 ลงใน Socket ที่กำหนดไว้บนบอร์ด MP3.ก่อน และทำการต่อสายสัญญาณสื่อสารดังรูป



รูป แสดงการต่อสายสื่อสารแบบ RS232 ระหว่างตัวส่งและตัวรับ

5.3.2 การส่ง Command ผ่าน RS422

ในการส่ง Command ด้วยการสื่อสารแบบ RS422 นี้เหมาะสำหรับใช้ควบคุมบอร์ด MP3 เพียงบอร์ดเดียว เพราะถ้าพ่วงกันหลายบอร์ดจะทำให้การตรวจสอบ Respond จากบอร์ด MP3 นั้นผิดพลาดได้ ซึ่งจะเหมือนกับแบบ RS232 แต่มีข้อดีตรงที่สามารถสื่อสารได้ไกลขึ้น ในการใช้งานจะต้องใส่ IC Line Driver 75176 2 ตัว ลงใน Socket ที่กำหนดไว้บนบอร์ด MP3. จากนั้นให้ Set Jumper หมายเลข 12 มาทางด้าน Full และ Jumper หมายเลข 13 มาทางด้าน 422 จากนั้นทำการต่อสายสัญญาณสื่อสารดังรูป

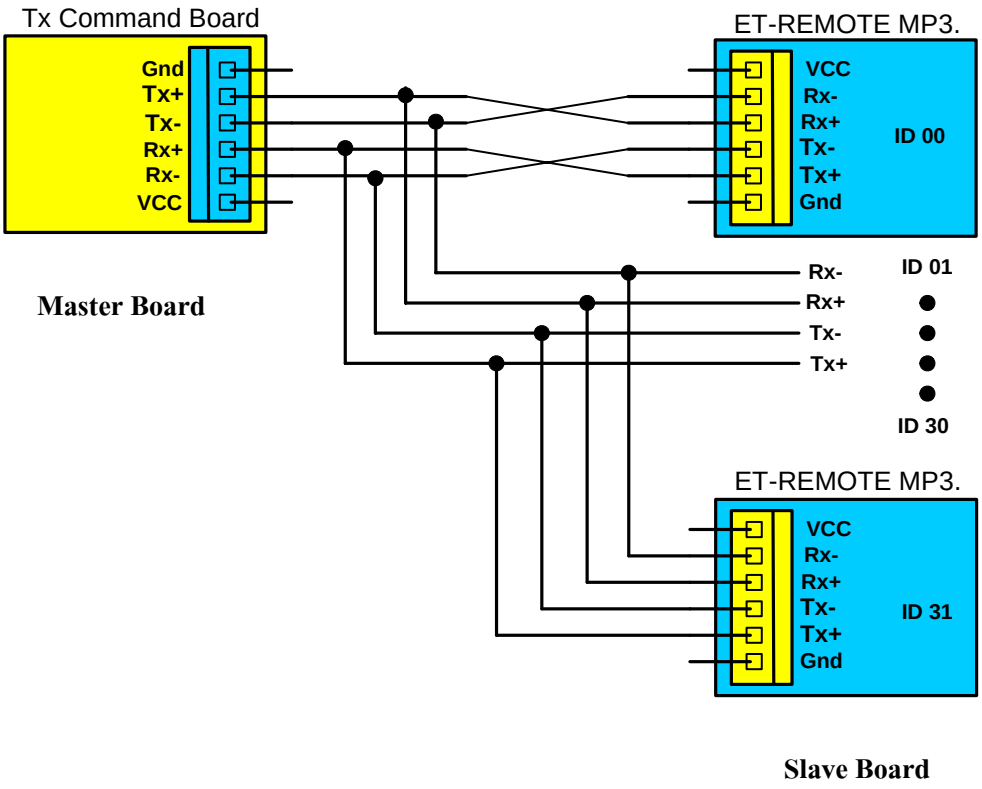


รูป แสดงการต่อสายสื่อสารแบบ RS422 ระหว่างตัวส่งและตัวรับ

5.3.3 การส่ง Command ผ่าน RS485 แบบ 4 Line

ในการส่ง Command ด้วยการสื่อสารแบบ RS485 แบบ 4 Line เราจะใช้การต่อสายสัญญาณแบบ RS422 ซึ่งดูได้จากรูปด้านล่าง การสื่อสารแบบนี้เหมาะสำหรับใช้ควบคุมบอร์ด MP3 หลายๆบอร์ด ซึ่งบอร์ด ET-REMOTE MP3. นี้ สามารถต่อพ่วงกันได้ 32 บอร์ด โดยแต่ละบอร์ดจะต้อง Set ID ให้ต่างกัน ซึ่งสามารถ Set ID ได้ตั้งแต่ค่า 00 -31 ดูได้จากตารางการ Set Jumper ที่กล่าวไว้ในตอนต้นของคู่มือ

ในการใช้งานจะต้องใส่ IC Line Driver 75176 2 ตัว ลงใน Socket ที่กำหนดไว้บนบอร์ด MP3. จากนั้นให้ Set Jumper หมายเลข 12 มาทางด้าน Full และ Jumper หมายเลข 13 มาทางด้าน 485 จากนั้นทำการต่อสายสัญญาณสื่อสารดังรูป



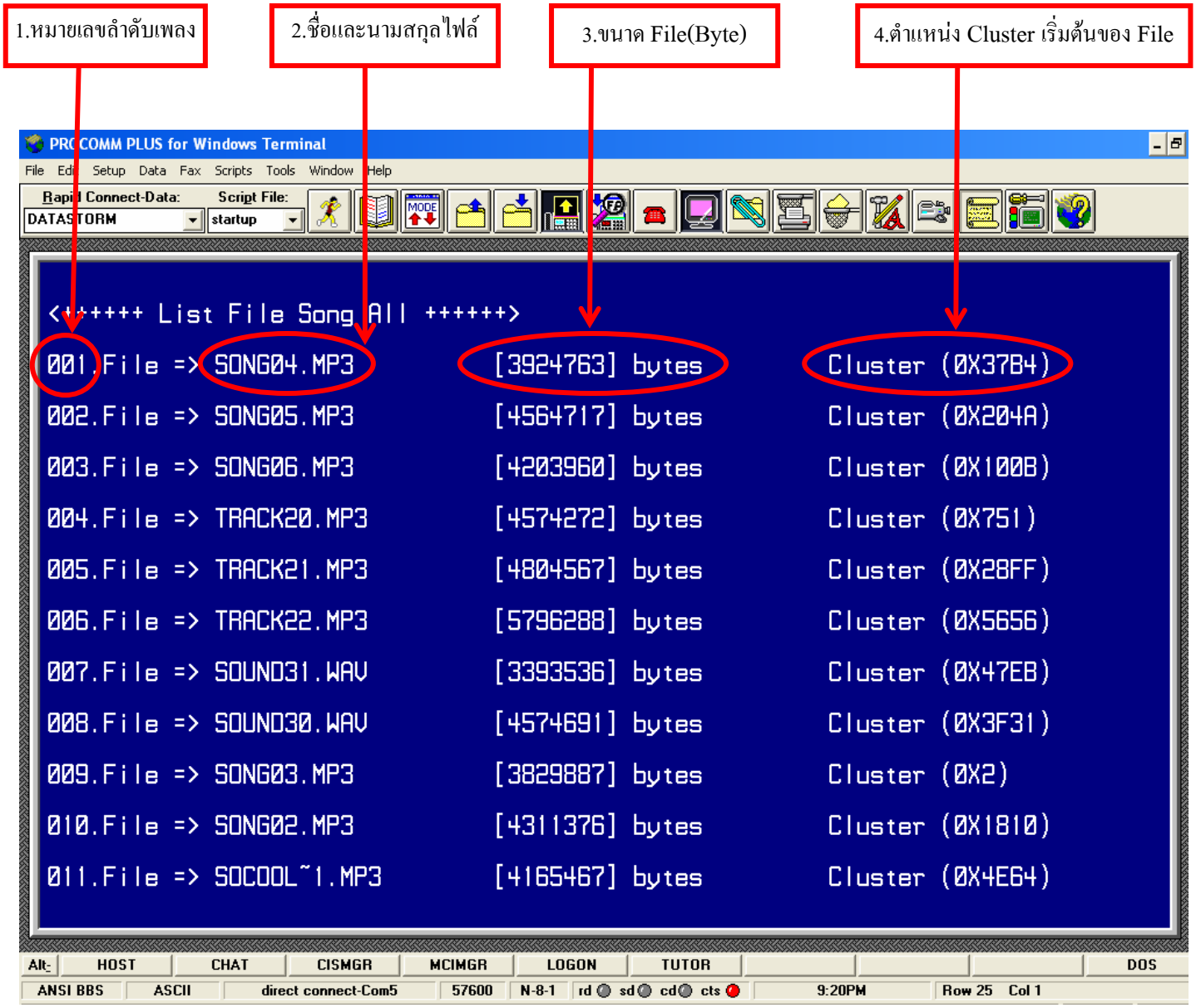
รูป แสดงการต่อสายสื่อสารแบบ RS485 4 Line ระหว่าง ตัวMaster และ ตัว Slave

สำหรับการสื่อสารด้วย RS485 นี้ ตัวบอร์ด MP3 จะรับ Command มากระทำก็ต่อเมื่อ Command นั้นมี ID ตรงกับ ID ของบอร์ดที่ได้ Set ไว้เท่านั้น เมื่อบอร์ดรับ Command เรียบร้อยแล้วก็จะส่ง Echo Command ตอบกลับออกมาให้ ส่วนบอร์ดอื่นๆที่ ID ไม่ตรงก็จะไม่ส่งอะไรออกมา และทุกครั้งเมื่อบอร์ดใดเล่นเพลงจบ บอร์ดนั้นก็จะมีการส่ง Command End of File ออกมาให้ด้วย ถ้า Command End of File นี้ถูกส่งออกมาจาก Slave Board พร้อมกันมากกว่า 2 บอร์ดขึ้นไป จะทำให้ตัว Master Board รับค่า Respond Command จากตัว Slave Board ผิดพลาดได้ซึ่งจะต้องระวังในส่วนนี้ด้วย รวมถึง ถ้ามีการ Set บอร์ดให้สามารถส่งเวลาในการเล่นออกมาด้วยแล้วจะต้องทำการ ON OFF การส่งเวลาให้ถูกจังหวะด้วย มิฉะนั้นจะทำให้ตัว Master Board รับค่า Respond ผิดพลาด ดังนั้นในการควบคุมบอร์ด MP3 หลายๆบอร์ดนั้นถ้าผู้ใช้มีการอ่านค่า Respond Command มาใช้งานจะต้องคำนึงถึงการส่งข้อมูลออกมาจากหลายๆบอร์ดพร้อมกันด้วย เพราะถ้าตัว Slave Board ส่งข้อมูลออกมาพร้อมกันเมื่อไหร่ นั่นคือตัว Master Board จะรับข้อมูลผิดพลาดอย่างแน่นอน จะส่งผลให้ไม่สามารถควบคุมการเล่นได้ตามต้องการได้

ข้อควรจำ *ในการใช้งานโหมด RS232 จะต้องใส่ IC Line Driver MAX232 เข้าไป และต้อง ถอด IC Line Driver 75176 ทั้ง 2 ตัว ออกเสียก่อน(ถ้ามีการใส่ไว้) ในทางกลับกัน ถ้าจะใช้งานในโหมด RS422 หรือ RS485 ก็จะต้องใส่ IC Line Driver 75176 ทั้ง 2 ตัวเข้าไป และจะต้องถอด IC Line Driver Max232 ออกเสียก่อน(ถ้ามีการใส่ไว้) ห้ามใส่พร้อมกันทั้ง 2 เบอร์ มิฉะนั้นจะทำให้การรับส่งข้อมูลนั้นผิดพลาด...*

5.4 การดูรายชื่อ และ หมายเลขลำดับ ของไฟล์ ที่จะนำมาใส่ใน Command

ในการดูรายชื่อ นามสกุลของไฟล์ และหมายลำดับของไฟล์ ที่มีอยู่ใน SD Card ทั้งหมดนั้น มีความจำเป็นอย่างยิ่ง เพื่อให้ผู้ใช้สามารถนำมาแทนค่าลงใน Command Play ได้อย่างถูกต้อง วิธีการดูนั้นผู้ใช้จะต้องใช้การสื่อสาร แบบ RS232 หรือ RS422 เท่านั้น โดยต่อสายสื่อสารจากบอร์ด MP3 ไปยัง Com Port ของ เครื่อง PC ในกรณีที่ใช้สื่อสารแบบ RS422 จะต้องมีชุดแปลงจาก RS422 เป็น RS232 เสียก่อนถึงจะนำมาต่อที่ Com Port ของ PC ได้ แต่ถ้าใช้การสื่อสารแบบ RS232 ก็สามารถนำไปต่อเข้ากับช่อง Com Port ของ PC ได้เลย ซึ่งรูปแบบการต่อก็จะเหมือนกับในรูปหัวข้อ 5.3.1 เมื่อต่อสายสื่อสารจากบอร์ด MP3. เข้ากับ Com Port ของ PC เรียบร้อยแล้ว จากนั้นให้ผู้ใช้ เปิด โปรแกรม Hyperterminal หรือ Procomm ขึ้นมา และ Set ตำแหน่งของ Com Port ให้ตรงกับช่องที่ใช้งาน รวมทั้ง Set Baud Rate ในการรับส่ง ให้ตรงกับ ค่าที่ Set จาก Jumper บนบอร์ด MP3. ด้วย จากนั้นก็ทำการ กด SW. Reset ที่บอร์ด ET-REMOTE MP3 เมื่อบอร์ด Initial Card ผ่าน รายชื่อไฟล์ทั้งหมดก็จะปรากฏขึ้นที่หน้าต่างของ Hyperterminal หรือ Procomm ซึ่งแสดงดังรูปด้านล่าง



รูป แสดงตัวอย่างรายชื่อไฟล์ใน SD Card ที่อ่านได้จาก ET-REMOTE MP3. Board

จากตัวอย่าง List File ด้านบน จะเป็นการอ่านชื่อไฟล์มาจาก SD Card โดยใช้โปรแกรม Procomm เป็นตัวแสดงผล ซึ่งจะเห็นว่า ในตำแหน่งที่1 จะเป็นส่วนของ หมายเลขลำดับ(Number)จะมีอยู่ด้วยกัน 3 หลัก ในการใช้งาน Command Play ด้วย Number ให้ผู้ใช้นำหมายเลขส่วนนี้ไปใส่ไว้ใน Command จากรูปที่วงไว้คือหมายเลข '001' ในตำแหน่งที่ 2 จะเป็นส่วนของชื่อและนามสกุลไฟล์ ซึ่งในภาพที่วงไว้คือ 'SONG04.MP3' เวลาจะใช้งาน Command Play ด้วย File Name ผู้ใช้ก็นำชื่อและนามสกุลของไฟล์ในส่วนนี้ไปแทนลงใน Command ได้เลย ในตำแหน่งที่ 3 และ 4 จะบอกให้ทราบถึง ขนาดของ File และตำแหน่ง Cluster เริ่มต้นที่ใช้เก็บไฟล์นั้นๆ ซึ่งไม่จำเป็นต้องสนใจเนื่องจากไม่ได้นำมาใช้ในส่วนของ Command ต่างๆ

5.5 ตัวอย่างโปรแกรม Application Control ET-REMOTE MP3 V1 .

สำหรับตัวอย่างที่แสดงทั้งหมดนี้ จะใช้ MCU Z8Encore! (Z8F6422) เป็นตัวควบคุมการทำงานของบอร์ด ET-REMOTE MP3. V1. โปรแกรมที่เขียนจะเป็น ภาษา C ซึ่งซอร์ฟแวร์ที่ใช้เขียนคือ ZDS II ดังนั้นเวลาจะRun โปรแกรมตัวอย่างจะต้อง Run ลงใน MCU Z8Encore! จึงจะเห็นผล ถ้าผู้ใช้จะควบคุมบอร์ด MP3. ด้วย MCU เบอร์อื่นๆ ผู้ใช้สามารถนำตัวอย่างที่ให้ไปปรับใช้ได้โดยไม่ยากนัก ซึ่งตัวอย่างที่ให้ไปนี้เป็นเพียงแนวทางการใช้งานเท่านั้น

Application 01. ตัวอย่างนี้จะส่งคำสั่ง ผ่านRS232 หรือ 422 กำหนดให้ ID = 01,Baud Rate การรับส่งข้อมูล = 57600 ตัวอย่างนี้จะอ่านค่า SW. เข้ามาทาง Port B เมื่อ SW.ใดถูกกด ก็จะทำการส่ง Command 00 (Play by File Name) เพื่อเล่นไฟล์ตามที่กำหนดไว้ใน Command นั้นๆ


```
/************************************************************
**         Pro.Application_01 Use RS232/422(CH0)          **
**                                                        **
**           PB0-PB7  =  SW0-SW8 (Input)                  **
**           PC0      =  EOF Pulse(Input)                 **
**           PA4      =  Rx  Uart0                        **
**           PA5      =  Tx  Uart0                        **
**                                                        **
************************************************************/

#include <stdio.h>
#include <ez8.h>

//*************** Delay *****************

 void delay (int count)
  {
    int  i , j ;
    for(i = 0 ; i < = count ; i++)
    for(j = 0 ; j < = count ; j++) ;
  }

//************* Check Buffer UART(0) *************

//------------- Check Rx0 Buffer Empty? ------------

 void Rx0()
  {
    while(!(U0STAT0 & 0x80)){;}
  }

//------------ Check Tx0 Buffer Empty? --------------

 void Tx0()
  {
    while(!(U0STAT0 & 0x04)){;}
  }
```

```
//*************** Check Respond Board ET-REMOTE MP3. ***************

//-------------------- Check Echo Command -----------------------

void Echo_cmm()
 {
   unsigned char buf_cmm[13] ;
   unsigned char n,ch        ;

   do{
      do{
          Rx0()              ;
          ch = U0D           ;
        }while(ch != '#')    ;  //Check Byte Start of Echo Cmm ('#'=0x23)

       for(n=0;n<13;n++)          //if ch =‘#’then Keep Time 5 byte (mm:ss)
       {
         Rx0()                 ;
         buf_cmm[n] = U0D      ;
       }
    }while(buf_cmm[5] != 'R') ;  //('R'=0x52)

 }

//************************** Main Program *****************************

void main(void)
 {
  unsigned char enter,key,sw;

//--------------- initial Uart0 -------------------

   U0BRH   =  0x00      ;
   U0BRL   =  20        ;    // Set Baud Rate = 57600 Kb/s.
   PAAF    =  0x30      ;    // Set Altternet Function PA4-5 for Urat 0
   U0CTL0  =  0xc0      ;    // Control Register Uart0 Function

   PCDD    =  0xFF      ;    // PC = Input(PC0 = EOF)
   PADD    =  0x00      ;
   PBDD    =  0xFF      ;    //PB = Input
   U0D     =  0x00      ;    //Clear data register
   enter   =  0x0D      ;    //Charecter End commmand

//-------------- Start -------------

   delay(500)                       ;    //Delay For Sync Board MP3.
   printf ("*0:01:09=240%c",enter)  ;    //Sent Cmm_09:Set Volum = 240
   Echo_cmm()                       ;    //Check Echo Cmm_09

   while(1)
    {
     delay(300)         ;
     sw    =  PBIN      ;      //Read SW0-SW8(Port B)
     key  = sw          ;

     switch ( key )            //Check sw.
      {
        case 0xFE :                                      //Press SW0 (PB0=0)
                   printf("*0:01:00=n0.wav;%c",enter)   ;//Sent Cmm_00:Play File n0.wav
                   Echo_cmm()                           ; //Check Echo Cmm_00
                   key = 0xFF                           ; //Clear valiable
            break ;

        case 0xFD :                                      //Press SW1. (PB1=0)
                  printf("*0:01:00=N1.WAV;%c",enter)   ; //Sent Cmm_00:Play File n1.wav
                   Echo_cmm()                           ;
                   key = 0xFF                           ;
            break ;
```

```
            case 0xFB :                                          //Press SW2. (PB2=0)
                       printf("*0:01:00=n2.wav;%c",enter)   ; //Sent Cmm_00:Play File n2.wav
                       Echo_cmm()                           ;
                       key = 0xFF                           ;
                  break ;

            case 0xF7 :                                          //Press SW3. (PB3=0)
                       printf("*0:01:00=N3.WAV;%c",enter)   ; //Sent Cmm_00:Play File n3.wav
                       Echo_cmm()                           ;
                       key = 0xFF                           ;
                  break ;

            case 0xEF :                                           //Press SW4 (PB4=0)
                       printf("*0:01:00=n4.wav;%c",enter)   ; //Sent Cmm_00: Play File n4.wav
                       Echo_cmm()                           ;
                       key = 0xFF                           ;
                 break ;

            case 0xDF :                                          //Press SW5. (PB5=0)
                       printf("*0:01:00=N5.WAV;%c",enter)   ; //Sent Cmm_00:Play File n5.wav
                       Echo_cmm()                           ;
                       key = 0xFF                           ;
                 break ;

           case 0xBF  :                                          //Press SW6. (PB6=0)
                       printf("*0:01:00=n6.wav;%c",enter)   ;//Sent Cmm_00: Play File n6.wav
                       Echo_cmm()                           ;
                       key = 0xFF                           ;
             break ;

            case 0x7F :                                           //Press SW7. (PB7=0)

                       printf("*0:01:00=N7.WAV;%c",enter)   ; //Sent Cmm_00:Play File n7.wav
                       Echo_cmm()                           ;
                       key = 0xFF                           ;
             break ;

          }

         while(sw != 0xFF)            //Protect Press SW.Repeat
          {
           PAOUT |=  0x01           ;  //LED Status SW. (PA0 = 1)
            sw  =  PBIN             ;
          }

         PAOUT &= 0xFE              ;  //LED Status SW. (PA0 = 0)
      }
   }
```

Application 02. ตัวอย่างนี้จะส่งคำสั่ง ผ่านRS232 หรือ 422 กำหนดให้ ID = 01,Baud Rate การรับส่งข้อมูล = 57600 ตัวอย่างนี้จะเป็นการเล่นเพลงโดยจะมีการตรวจสอบเวลาในการเล่นเข้ามาเกี่ยวข้อง คือ เมื่อเพลงถูกเล่นไปได้ 5 วินาที เราก็จะทำการแทรกเพลง 1 ครั้ง เมื่อเพลงที่แทรกเล่นจบ ก็จะกลับมาเล่นเพลงเดิมในตอนแรกต่อ จากนั้นก็จะทำการตรวจสอบเวลาการเล่นของเพลงแรกอีก(ครั้งที่2)เมื่อได้ 5 วินาที ก็จะทำการแทรกเพลงอีก 1 ครั้ง เมื่อเพลงที่แทรกถูกเล่นจบลง ก็จะกลับมาเล่นเพลงแรกต่อจากเดิม จากนั้นก็จะรอจนกว่าจะมีการ ส่ง EOF ออกมา นั่นแสดงว่าเพลงหลักนั้นได้จบลงแล้ว จึงกลับไปวนเล่นใหม่

```
/*****************************************************
 **                                                 **
 **     Pro.Application_02 Use RS232/422(CH0)       **
 **                                                 **
 **               PA4 = Rx  Uart0                   **
 **               PA5 = Tx  Uart0                   **
 **                                                 **
 *****************************************************/

  #include <stdio.h>
  #include <ez8.h>

//*************** Delay ******************

void delay (int count)
 {
    int i,j;
      for(i=0;i<=count;i++)
      for(j=0;j<=count;j++);
 }

//************* Check Buffer UART(0) *************

//----------- Check Rx0 Buffer Empty? ----------
  void Rx0()
   {
     while(!(U0STAT0 & 0x80)){;}
   }

//------------ Check Tx0 Buffer Empty? -----------
  void Tx0()
   {
     while(!(U0STAT0 & 0x04)){;}
   }

//*************** Check Respond Board ET-REMOTE MP3.***************

//-------------- Check Echo Command ----------------

void Echo_cmm()
 {
   unsigned char buf_cmm[13] ;
   unsigned char n,ch        ;

   do
    {
     do
       {
          Rx0()               ;
         ch = U0D             ;
       }while(ch != '#')      ;    //Check Byte Start of Echo cmm('#'=0x23)

     for(n=0;n<13;n++)           //if ch = # thun Keep Time 5 byte(mm:ss)
      {
        Rx0()                 ;
       buf_cmm[n] = U0D       ;
      }

    }while(buf_cmm[5]!='R')   ;  //('R'=0x52)
 }
```

```
//----------------- Check Play Time -----------------

void Chk_Time(char T_mH,char T_mL,char T_sH,char T_sL)
 {
  unsigned char buf_time[8] ;
  unsigned char t,ch        ;

   T_mH = T_mH + 0x30 ;  //Convert ASCII Minit mm
   T_mL = T_mL + 0x30 ;

   T_sH = T_sH + 0x30 ;  //Convert ASCII Sec ss
   T_sL = T_sL + 0x30 ;

   do
     {
      do
        {
          Rx0()             ;
          ch = U0D          ;
        }while(ch != '@')   ;  //Check Byte start of Timer('@'=0x40)

      for(t=0;t<8;t++)         //if ch=# then Keep Time 5 byte(mm:ss)
       {
         Rx0()              ;
        buf_time[t] = U0D   ;
       }
     }while((buf_time[3]!= T_mH)||(buf_time[4]!=T_mL)||(buf_time[6]!=T_sH)||
            (buf_time[7]!= T_sL));
 }

//---------------- Check End of File Main -----------------

 void EOF_M()
  {
   unsigned char buf_eof[4] ;
   unsigned char ch,e ;

   do
    {
      do
       {
         Rx0();
         ch = U0D ;
       }while(ch !='$');  //Check Byte Start of EOF main ('$'=0x24)

      for(e=0;e<4;e++)
       {
         Rx0();
         buf_eof[e] = U0D ;
       }
    }while(buf_eof[3] != 'E');  //('E'=0x45)
 }

//---------------- Check End of File Insert -----------------

 void end_ins()
 {
   unsigned char buf_end[4] ;
   unsigned char ch,e       ;

   do
    {
     do
       {
         Rx0()          ;
         ch = U0D       ;
       }while(ch !='$') ; // Check Byte Start of End of file Inseret ('$'=0x24)
```

```
     for(e=0;e<4;e++)            //if ch=$ then Keep eof of file insert 4 byte(ID:e)
      {
        Rx0()                  ;
        buf_end[e] = U0D       ;
      }
    }while(buf_end[3] != 'e') ;  //('e'=0x65)
 }

//************************** Main Program ****************************

 void main(void)
  {
    unsigned char enter,key,sw;

//----------- initial Uart0 ------------
     U0BRH  = 0x00   ;
     U0BRL  = 20     ;   //Set Baud Rate 20=57600 Kb/s.
     PAAF   = 0x30   ;   //Set Altternet Function PA4-5 for Urat 0
     U0CTL0 = 0xc0   ;   //Control Register Uart0 Function

     PBDD   = 0xFF   ;  //PB = Input
     U0D    = 0x00   ;  //Clear data register
     enter  = 0x0D   ;  //Charecter End commmand

 //--------------- Start ----------------

  delay(500);                     //Delay For Sync Board MP3.

  printf("*0:01:09=225%c",enter); //Sent Cmm_09 : Set Volum = 225
  Echo_cmm()                    ; //Check Echo Cmm_09

  printf("*0:01:09=B10%c",enter); //Sent Cmm_09 : Set BASS = B10
  Echo_cmm()                    ; //Check Echo Cmm_09

  printf("*0:01:08=T%c",enter)  ; //Sent Cmm_08 : Sent Play Time on Rs232/422
  Echo_cmm()                    ; //Check Echo Cmm_09

  while(1)
   {
     printf("*0:01:00=TOYOTA.MP3;%c",enter)  ; //Sent Cmm_00:Play File TOYOTA.MP3
     Echo_cmm()                              ; //Check Echo Cmm_00
     Chk_Time(0,0,0,5)                       ; //Check Play Time = 00:05 ?
     printf("*0:01:04=MUSIC_~1.MP3;%c",enter); //Sent Cmm_04:Insert File MUSIC_~1.MP3
     end_ins()                               ; //Check end of file Insert
     Chk_Time(0,0,0,5)                       ; //Check Play Time = 00:05 ?
     printf("*0:01:05=018;%c",enter)         ; //Sent Cmm_05:Insert by Number(018)
     end_ins()                               ; //Check end of file Insert
     EOF_M()                                 ; //Check end of file Main(File TOYOTA.MP3)
   }
 }
```

Application03. ตัวอย่างนี้จะส่งคำสั่ง ผ่านRS232 หรือ 422 กำหนดให้ ID = 01,Baud Rate การรับส่งข้อมูล = 57600 ตัวอย่างนี้จะอ่านค่า SW0.,SW1 เข้ามาทาง PB0,PB1 เมื่อ SW.ใดถูกกด ก็จะทำการส่ง Command 02 (Play by Number) เพื่อเล่นไฟล์ตามหมายเลขลำดับเพลงที่ส่งออกไป โดย PB0= SW_Up คือเพิ่มค่าลำดับเพลงทีละ 1 ส่วน PB1 = SW_Down คือลดค่าลำดับเพลงลงครั้งละ 1 ส่วน PA0 จะใช้ต่อ LED เมื่อเล่นเพลงจบ LED จะติด บอกให้รู้ว่า สามารถกด SW.ได้ใหม่ ถ้ากดขณะเล่นเพลง จะยังไม่มีผลใดๆ เนื่องจากยังติดอยู่ในฟังก์ชัน EOF_Pule() ซึ่งฟังก์ชันนี้ จะอ่านค่า EOF (Logic ‘0’) จากขั้วต่อ EOFของบอร์ด MP3. เพื่อตรวจสอบการสิ้นสุดของไฟล์ที่เล่นอยู่ ในตัวอย่างนี้จะกำหนดลำดับเพลงไว้ที่ 001-035 ดังนั้นผู้ใช้จะต้องมีเพลงอยู่ใน SD Card อย่างน้อย 35 เพลง เพื่อว่าเวลาทดสอบจะได้ไม่ติดค้างอยู่ใน Loop EOF_Pulse ()

```
                    /*******************************************
                     **                                       **
                     **   Pro.Application_03 Use RS232/422(CH0)  **
                     **                                       **
                     **   PB0-PB1 = SW0-SW1 (Input)           **
                     **       PC0 = EOF Pulse(Input)          **
                     **       PA0 = LED Status EOF (Out)      **
                     **       PA4 = Rx  Uart0                 **
                     **       PA5 = Tx  Uart0                 **
                     **                                       **
                     *******************************************/

#include <stdio.h>
#include <ez8.h>

//*************** Delay *****************

void delay (int count)
 {
  int i,j;
  for(i=0;i<=count;i++)
  for(j=0;j<=count;j++);
 }

//************* Check Buffer UART(0) *************

//----------- Check Rx0 Buffer Empty? ----------

 void Rx0()
  {
    while(!(U0STAT0 & 0x80)){;}
  }

//------------ Check Tx0 Buffer Empty? -----------

 void Tx0()
  {
    while(!(U0STAT0 & 0x04)){;}
  }

//*************** Check Respond Board ET-REMOTE MP3. ***************

//-------------- Check Echo Command ----------------

void Echo_cmm()
 {
   unsigned char buf_cmm[13] ;
   unsigned char n,ch ;

   do
    {
     do
      {
        Rx0()            ;
        ch = U0D         ;
      }while(ch != '#')  ;    //Check Byte Start of Echo cmm('#'=0x23)

     for(n=0;n<13;n++)       //if ch = # Keep then Time 5 byte(mm:ss)
      {
       Rx0()               ;
       buf_cmm[n] = U0D    ;
      }
   }while(buf_cmm[5]!='R') ;  //('R'=0x52)
 }
```

```
//---------------- Check EOF for pulse signal -------------

void EOF_Pul()
{
  unsigned char ch ;

  do
   {
     ch  = PCIN       ;  //Check PulseEOF
     ch &= 0x01       ;
   }while(ch != 0x00) ;
}

//************************** Main Program *****************************

void main(void)
 {
  char enter,sw,song=0,numx,numy;

//----------- initial Uart0 ------------

   U0BRH   = 0x00    ;
   U0BRL   = 20      ;    //Set Baud Rate 20=57600 Kb/s.
   PAAF   =  0x30   ;    //Set Alternet Function PA4-5 for Urat 0
   U0CTL0  = 0xc0    ;   //Control Register Uart0 Function

   PADD    = 0x00    ;   //PA0 = Output
   PBDD    = 0x03    ;   //PB0,PB1 = Input(sw0,sw1)
   PCDD    = 0x01    ;   //PC0 = Input(PC0 Receive signal EOF)
   U0D     = 0x00    ;   //Clear data register
   enter   = 0x0D    ;   //Charecter End commmand

 //--------------- Start ----------------

   delay(500);                     // Delay For Sync Board MP3.

   printf("*0:01:09=240%c",enter); //Cmm_09: Set Volum = 240
   Echo_cmm()                    ; // Check Echo

   while(1)
    {
      delay(300)             ;
      sw    =  PBIN          ;   // Read SW0-SW1(Port B)
      sw   &=  0x03          ;

      switch(sw)
       {
         case 0x02 :                            //Press SW0 (PB0=0)
                   PAOUT &= 0xFE              ; // LED Status EOF (PA0=0,LED OFF)
                   song   = song+1            ;
                   numx   = (song/10)+0x30    ; // Convert Number to ASCII
                   numy   = (song%10)+0x30    ;

                   if(song<=35)
                    {
                     printf("*0:01:02=0%c%c;%c",numx,numy,enter); //Sent Cmm_02:Play by
                                                                  //number(001-035)
                     Echo_cmm()     ;            // Check Echo
                     EOF_Pul()      ;            // Check EOF Pulse
                     PAOUT |= 0x01  ;            // LED Status EOF (PA0=1,LED ON)
                    }
                   else
                    song = 0        ;

           break ;
```

```
          case 0x01 :                              //Press SW1 (PB1=0)

                     PAOUT &= 0xFE              ; //LED Status EOF (PA0=0,LED OFF)
                     song   = song-1            ;
                     numx   = (song/10)+0x30    ; //Convert Number to ASCII
                     numy   = (song%10)+0x30    ;

                    if(song >=1)
                     {
                      printf("*0:01:02=0%c%c;%c",numx,numy,enter); //Sent Cmm_02:Play by
                                                                   //number(001-035)
                      Echo_cmm()       ;
                      EOF_Pul()        ;
                      PAOUT |= 0x01    ;         // LED Status EOF (PA0=1,LED ON)
                     }
                    else
                      song = 36 ;
              break ;
        }
    }
}
```

Application 04. ตัวอย่างนี้จะ ส่งคำสั่ง ผ่านRS485 เพื่อทดสอบการควบคุมในลักษณะ Network โดยจะใช้บอร์ด MP3. 3 บอร์ดในการทดสอบ กำหนดให้ ID ของแต่ละ Board = 00,01 และ 02 ตามลำดับ Baud Rate การรับส่งข้อมูล = 57600 และให้ Set Jumper ในการสื่อสารของบอร์ดทั้ง 3 มาทางด้าน Full(Jหมายเลข12) และ ด้าน 485(Jหมายเลข15) ตัวอย่างนี้จะอ่านค่า SW. เข้ามาทาง Port PB0-PB2 เมื่อ SW.ใดถูกกด ก็จะทำการส่ง Command Play เพื่อเล่นไฟล์ตามที่กำหนดไว้ใน Command นั้นๆ


```
                    /*********************************************
                    **                                          **
                    **        Pro.Application_04 RS485(CH0)     **
                    **                                          **
                    **   PB0-PB2 = SW0-SW2 (Input)              **
                    **       PA0 = LED key (OUT)                **
                    **       PA4 = Rx  Uart0                    **
                    **       PA5 = Tx  Uart0                    **
                    **                                          **
                    *********************************************/

#include <stdio.h>
#include <ez8.h>

//*************** Delay *****************

void delay (int count)
{
 int i,j               ;
  for(i=0;i<=count;i++)
 for(j=0;j<=count;j++) ;
}

//************* Check Buffer UART(0) ************

//----------- Check Rx0 Buffer Empty? ----------
void Rx0()
{
 while(!(U0STAT0 & 0x80)){;}
}
```

```
//------------ Check Tx0 Buffer Empty? -----------
 void Tx0()
  {
    while(!(U0STAT0 & 0x04)){;}
  }

//*************** Check Respond Board ET-REMOTE MP3. ***************

//--------------- Check Echo Command ----------------
 void Echo_cmm()
 {
  unsigned char buf_cmm[13] ;
  unsigned char n,ch        ;

  do
   {
    do
     {
      Rx0()            ;
      ch = U0D         ;
     }while(ch != '#') ;    //Check Byte Start of Echo cmm('#'=0x23)

    for(n=0;n<13;n++)       //if ch = # then Keep cmm 13 byte(0:xx:id=xxx)
     {
      Rx0()            ;
      buf_cmm[n] = U0D ;
     }

  }while(buf_cmm[5]!='R');  //('R'=0x52)
 }


//*************************** Main Program ******************************

 void main(void)
  {
   unsigned char enter,key,sw;

  //------------ initial Uart0 ------------

   U0BRH  = 0x00                ;
   U0BRL  = 20                  ;   //Set Baud Rate 20=57600 Kb/s.
   PAAF   = 0x30                ;   //Set Altternet Function PA4-5 for Urat 0
   U0CTL0 = 0xc0                ;   //Control Register Uart0 Function

   PADD   = 0x00                ;
   PBDD   = 0x07                ;   //PB = Input
   U0D    = 0x00                ;   //Clear Buffer data register
   enter  = 0x0D                ;   //Charecter End commmand

 //--------------- Start ----------------

  delay(500)                    ;// Delay For Sync Board MP3.

  printf("*0:00:09=245%c",enter) ; //Cmm Set Volum Board ID 00 = 245
  Echo_cmm()                    ; // Check Echo
  printf("*0:01:09=245%c",enter) ; //Cmm Set Volum Board ID 01 = 245
  Echo_cmm()                    ; // Check Echo
  printf("*0:02:09=245%c",enter) ; //Cmm Set Volum Board ID 02 = 245
  Echo_cmm()                    ; // Check Echo

  while(1)
   {
     delay(300)                 ;
     sw   =  PBIN               ;   // Read SW0-SW2(Port B)
     key  = sw & 0x07           ;
```

```
        switch(key)
         {
          case 0x06 :   //Press SW0 (PB0=0)

               printf("*0:00:00=n1.wav;n2.wav;n3.wav;n4.wav;n5.wav;%c",enter); //Sent Cmm_00
               key = 0xFF  ; // clear register key

          break ;

          case 0x05 :  //Press SW1. (PB1=0)

               printf("*0:01:02=011;012;013;014;015;%c",enter);//Sent Cmm_02:Play by Number
               key = 0xFF ;

          break;


          case 0x03 :  //Press SW2. (PB2=0)

               printf("*0:02:02=025;024;023;022;021;%c",enter); //Sent Cmm_02:Play by Number
               key = 0xFF ;

          break;
         }

        while(sw != 0x07)         //Protect Press SW.Repeat
         {
          PAOUT |=  0x01      ;   //LED Status SW. (PA0 = 1)
          sw     =  PBIN      ;
          sw     =  sw & 0x07 ;
         }
       PAOUT &= 0xFE          ; //LED Status SW. (PA0 = 0)
    }
}
```


สำหรับ Application นี้ จะเห็นว่าในการส่งคำสั่ง Play นั้นเราจะไม่มีการตรวจสอบ Echo เนื่องจากเป็นการส่ง Command เพียง Command เดียว ไม่มีการส่งคำสั่งต่อกันหลายๆคำสั่ง ดังนั้นไม่จำเป็นต้องตรวจสอบก็ได้ หรือจะตรวจสอบก็ได้ โปรแกรมก็ยังทำงานได้ตามปกติ

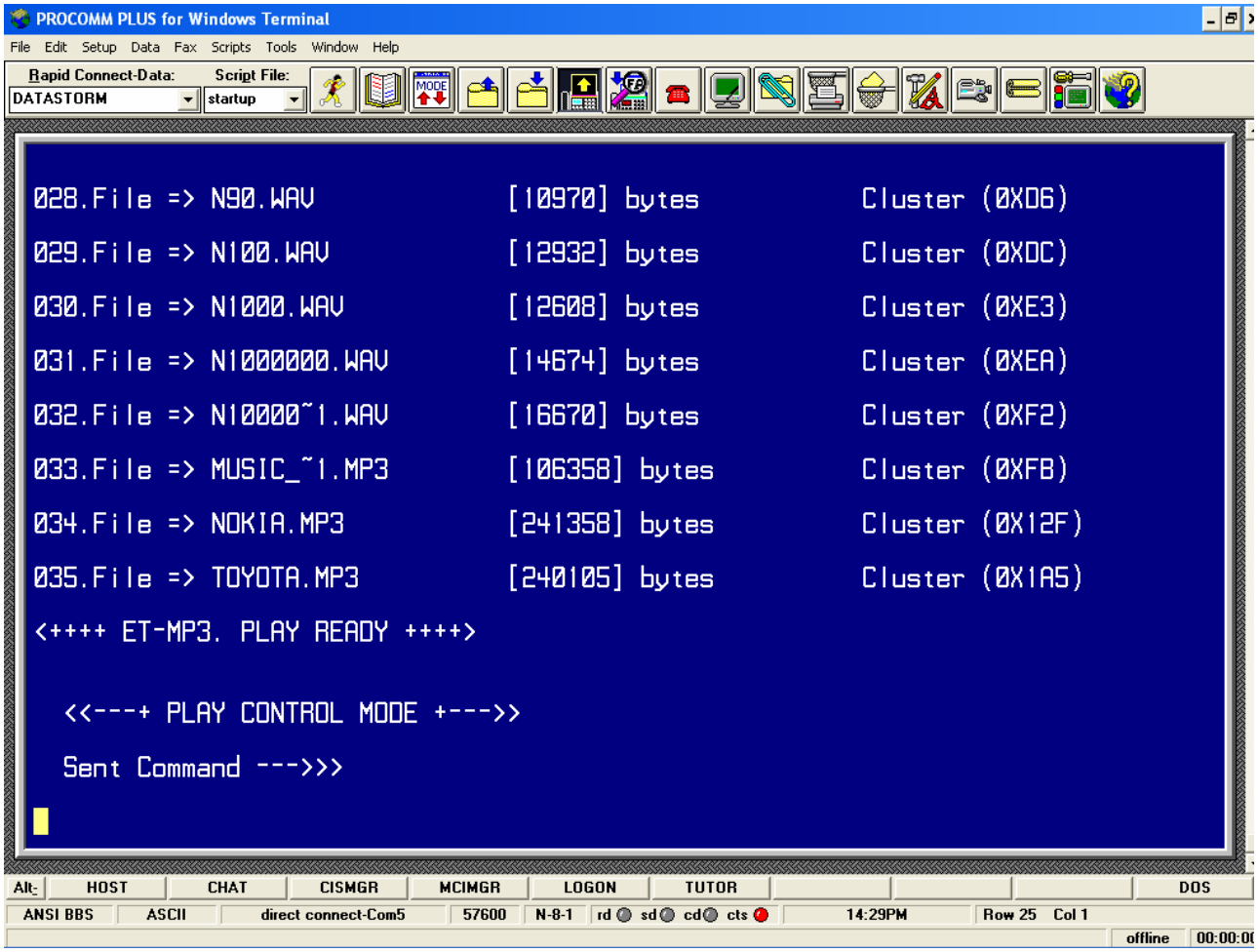
ในการทดสอบ Application 01-04 นี้ ให้ผู้ใช้ Copy ตัวอย่างไฟล์ MP3 และ WAV ที่ให้มากับ CD ลงใน SD Card ที่จะใช้ทดสอบทั้งหมดเพื่อให้โปรแกรมทำงานได้ถูกต้อง

5.6 การทดสอบส่ง Command ผ่าน RS232/422/485 โดยใช้ โปรแกรม PROCOMM

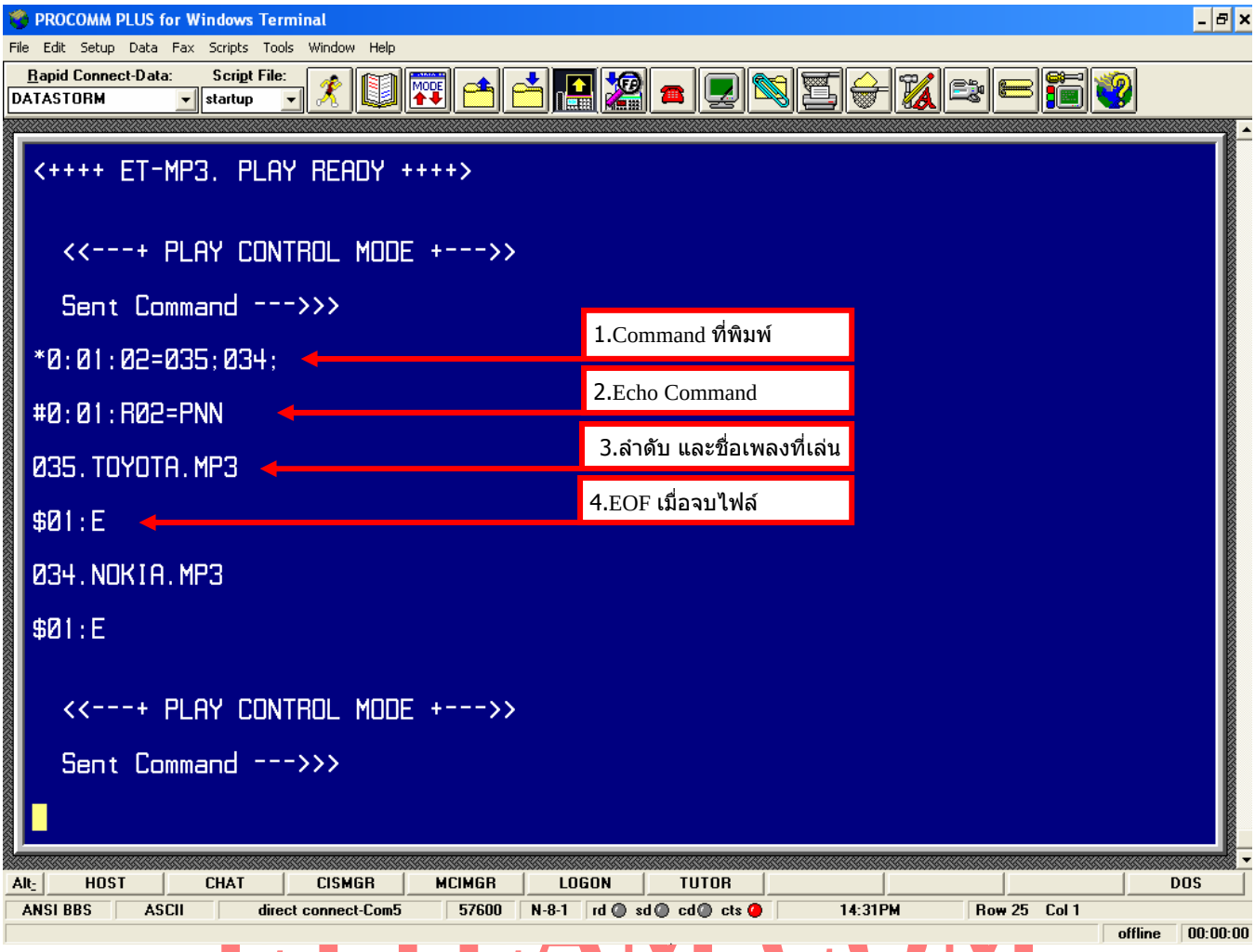
ในการทดสอบนี้จะเป็นการ Test Board ET-REMOTE MP3 แบบง่ายๆ โดยจะใช้โปรแกรม PROCOMM เป็นตัวรับ Command ที่พิมพ์จาก Key Board และส่งออกผ่านทาง RS232 ของ PC ไปยัง RS232/422 หรือ 485 ของบอร์ด MP3 ในกรณีที่ใช้ RS485 นั้น เวลาพิมพ์ Command จะไม่มีการแสดง Command ที่พิมพ์ออกที่หน้าต่างของ Procomm แต่จะแสดง Echo Command ที่บอร์ด MP3 ส่งกลับมาให้เห็น ถ้า Command ที่พิมพ์นั้นถูกต้อง ถ้าสื่อสารด้วย RS232/422 นั้นขณะที่พิมพ์ Command จะแสดงตัวอักษรขณะพิมพ์ให้เห็นที่หน้าจอด้วย ในกรณีที่พิมพ์ผิดให้กด Enter เพื่อเริ่มพิมพ์ Command ใหม่อีกครั้ง หลังพิมพ์ Command เสร็จจะต้องกด Enter เสมอ เพื่อสิ้นสุดการส่ง Command นั้นๆ ซึ่งจะเป็นไปตามรูปแบบ Command ที่กล่าวไปแล้วข้างต้น แสดงตัวอย่างการใช้งานดังรูปด้านล่าง

ในการทดสอบนั้นผู้ใช้จะต้องกำหนด Comport ใน Procomm ให้ตรงกับที่ต่อใช้งานจริง และกำหนด Baud Rate ใน Procomm ให้ตรงกับ Baud Rate ที่ Set ไว้ที่บอร์ด MP3. เวลาส่ง Command ก็พิมพ์คำสั่งตามรูปแบบที่กล่าวไปข้างต้น ในส่วน

ของ ID จะต้องพิมพ์ให้ตรงกับที่ Set ไว้บนบอร์ดด้วย ซึ่งในการทดสอบนี้ ให้ Set Jumper หมายเลข 17 ของบอร์ด MP3.มาด้าน Control ซึ่งจะ Set เหมือนกับใช้ MCU ควบคุมการทำงานทุกประการ ส่วนการ Set Port สื่อสาร RS 232/422/485 ของบอร์ด MP3 ก็ให้ Set เหมือนที่กล่าวไปข้างต้น (ในกรณีที่จะส่ง Command จาก PC โดยการสื่อสารแบบ RS 422/485 นั้น ผู้ใช้จะต้องมีชุดแปลง Port จาก RS232 ไปเป็น RS422 หรือ RS485 ด้วย)



หลังจากที่ Reset Board MP3 ที่หน้าจอ Procomm จะแสดงดังรูปด้านบน คือ พร้อมรับ Command (ถ้าสื่อสารแบบ RS485 จะไม่แสดงข้อความใดๆให้เห็นให้รอประมา 5 วินาทีแล้วจึงเริ่มส่ง Command ได้)

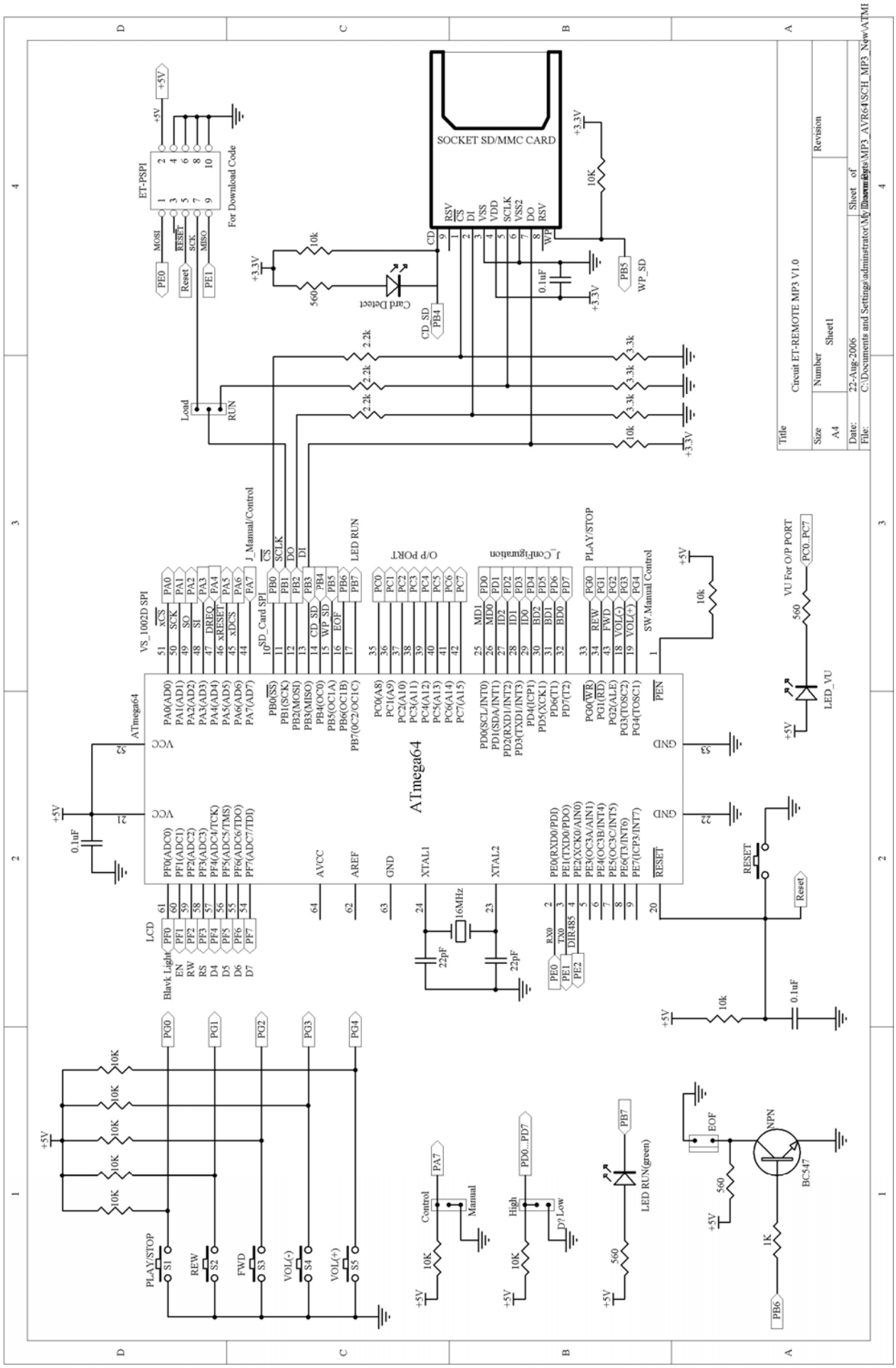


รูปด้านบนนี้ จะแสดงหลังจากที่ส่ง Command ไปแล้ว และ Command นั้นถูกต้อง จากรูปตัวอย่าง Command ที่ส่งคือ ' *0:01:02=035;034;Enter ' ID ที่ใช้คือ 01 โดย Command นี้ จะสั่งให้เล่นเพลงตามหมายเลขลำดับที่กำหนด คือ 035 และ 034 หลังจากกด Enter ก็จะมี Echo Command ส่งออกมาคือ ' #0:01:R02=PNN ' จากนั้นเพลงก็จะถูกเล่น และส่งหมายเลขลำดับ รวมทั้งชื่อและนามสกุลของเพลงที่เล่นอยู่ออกมาให้เห็น ในที่นี้คือ ' 035.TOYOTA.MP3 ' เมื่อเพลงที่เล่นจบลง ก็จะส่ง Command EOF ออกมา คือ ' $01:E ' ถ้ามีการ Enable ให้แสดงเวลาการเล่น เวลาก็จะถูกส่งออกมาให้เห็นด้วยเช่นกัน จากนั้นเพลงที่2 ก็จะถูกเล่นในลำดับต่อมาคือ ' 034.NOKIA.MP3 ' เมื่อเพลงที่ 2 จบ ก็จะมีการส่ง EOF ออกมาเช่นเดิม ถ้าไม่มีการกำหนดให้เล่นซ้ำไว้ หน้าจอก็จะแสดงสภาวะพร้อมรับ Command ให้เห็นดังรูปด้านบน อีกครั้ง

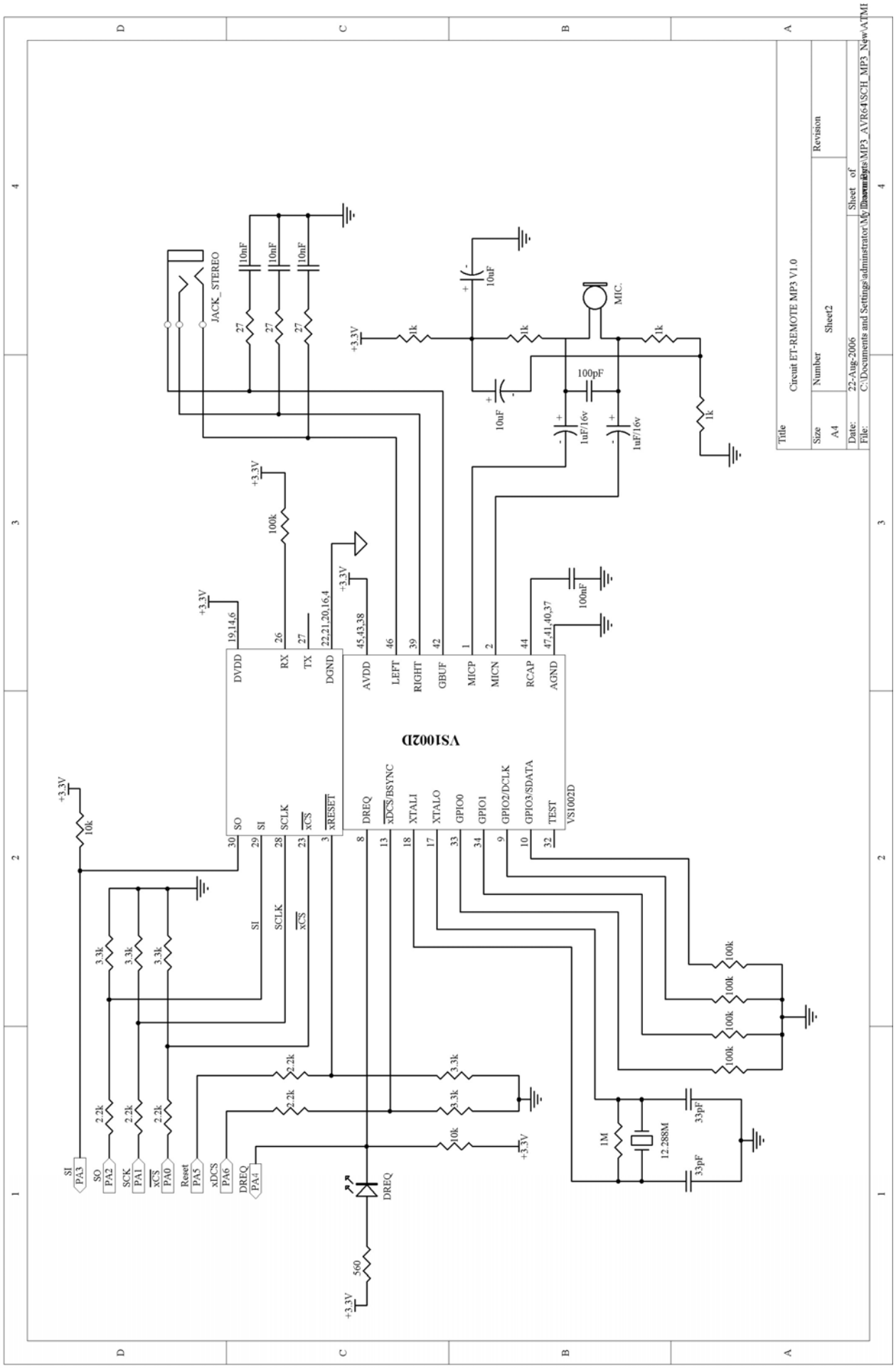
********** MANUAL ET-REMOTE MP3 V1.0 **********

By Mr.Sittiphol Yooyod.

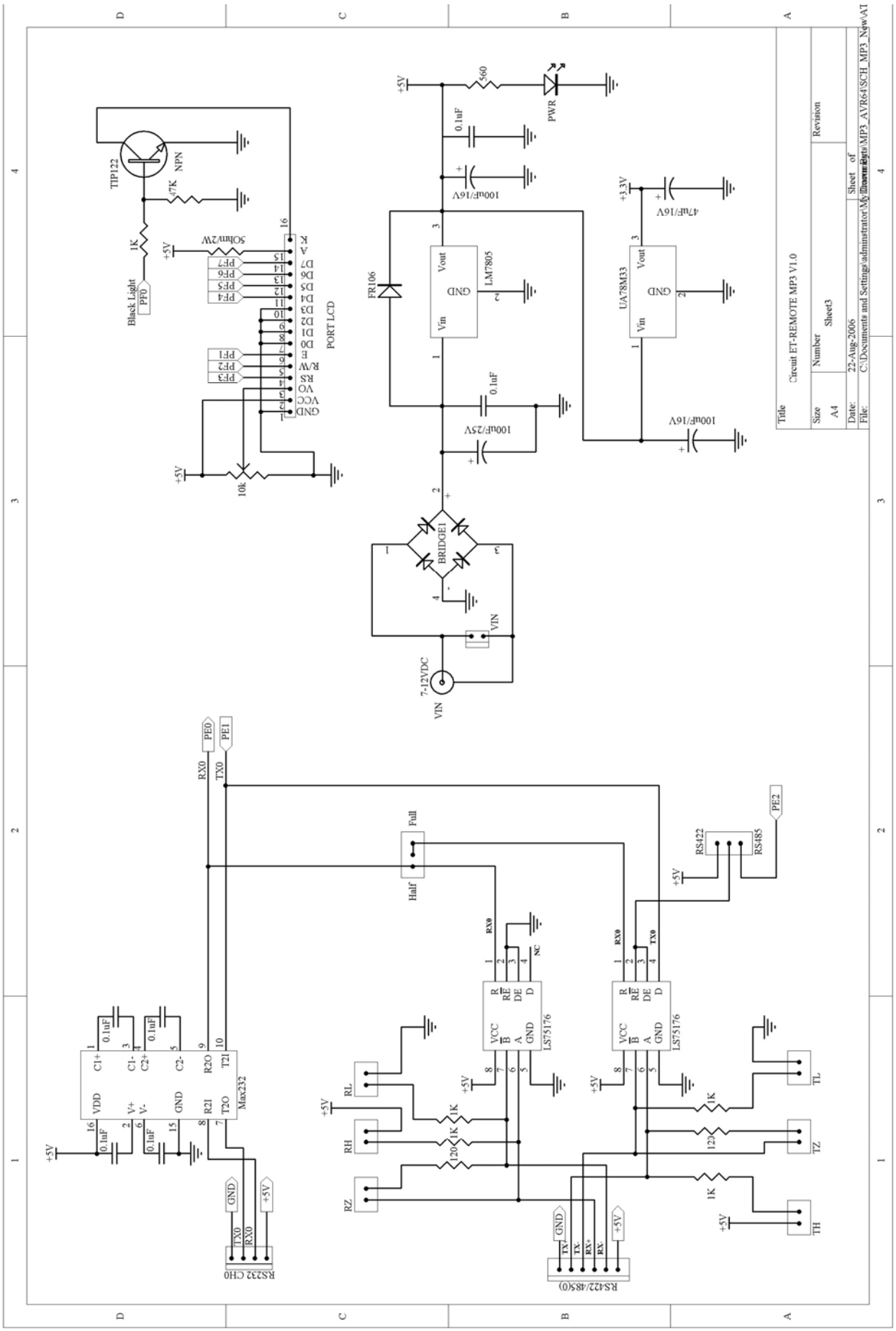
www.ett.co.th



วงจรบอร์ด ET-REMOTE MP3 V1.0 (Sheet1)



วงจรบอร์ด ET-REMOTE MP3 V1.0 (Sheet2)



วงจรบอร์ด ET-REMOTE MP3 V1.0 (Sheet3)