

TEST LCD ETT 0123456789

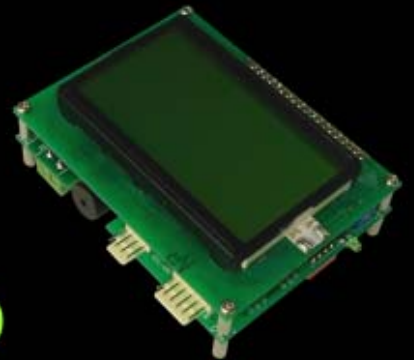
GRAPHIC LCD MODULE

คู่มือการใช้งาน

User's Manual

ET-REMOTE GLCDI2864 VI.0

ET-REMOTE GLCDI2864 VI.0 BACKLIGHT



บริษัท อีทีที จำกัด ETT CO., LTD.

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>

www.etteam.com

Tel : 02-7121120 Fax : 02-3917216

email : sale@etteam.com

สารบัญ

เรื่อง	หน้า
1. คุณสมบัติของบอร์ด ET-REMOTE GLCD12864	3
2. โครงสร้างของบอร์ด ET-REMOTE GLCD12864	4
3. การทำงานของบอร์ด ET-REMOTE GLCD12864	8
3.1 <i>Self-Test Mode</i>	8
3.2 <i>RUN-Mode</i>	9
4. รูปแบบและการใช้งานคำสั่งต่างๆที่บอร์ด ET-REMOTE GLCD12864 รองรับ	10
4.1 กลุ่มคำสั่งที่เกี่ยวข้องกับการ <i>Control Display</i> และอุปกรณ์บนบอร์ด	10
- COMMAND '00' (<i>Clear Screen</i>)	10
- COMMAND '01' (<i>Invert Screen</i>)	10
- COMMAND '02' (<i>Display Screen</i>)	11
- COMMAND '03' (<i>On/Off Back Light</i>)	11
- COMMAND '04' (<i>Sound</i>)	12
4.2 กลุ่มคำสั่งที่เกี่ยวข้องกับ <i>Graphic Mode</i>	13
- COMMAND '10' (<i>Plot Circle</i>)	13
- COMMAND '11' (<i>Plot Ellipse</i>)	14
- COMMAND '12' (<i>Plot Dot</i>)	14
- COMMAND '13' (<i>Plot Triangle</i>)	15
- COMMAND '14' (<i>Plot Rectangle</i>)	16
- COMMAND '15' (<i>Plot Line</i>)	17
- COMMAND '16' (<i>Plot Line Thick</i>)	17
- COMMAND '17' (<i>Plot Polygon</i>)	18
4.3 กลุ่มคำสั่งที่เกี่ยวข้องกับ <i>Picture Mode</i>	19
- วิธีการแปลงรูปภาพให้เป็น <i>Hex Code</i>	19
- COMMAND '20' (<i>Write Picture</i>)	23
- COMMAND '21' (<i>Demo Test GLCD</i>)	24
- COMMAND '22' (<i>Display Wall paper</i>)	24
4.4 กลุ่มคำสั่งที่เกี่ยวข้องกับ <i>Text Mode</i>	25
- COMMAND '30' (<i>Set Cursor Position</i>)	25
- COMMAND '31' (<i>Write Message1</i>)	26
- COMMAND '32' (<i>Write Message2</i>)	27
- COMMAND '33' (<i>Del Text ขนาด 7x5 dot</i>)	28
- COMMAND '34' (<i>ON/OFF Cursor</i>)	29

สารบัญ (ต่อ)

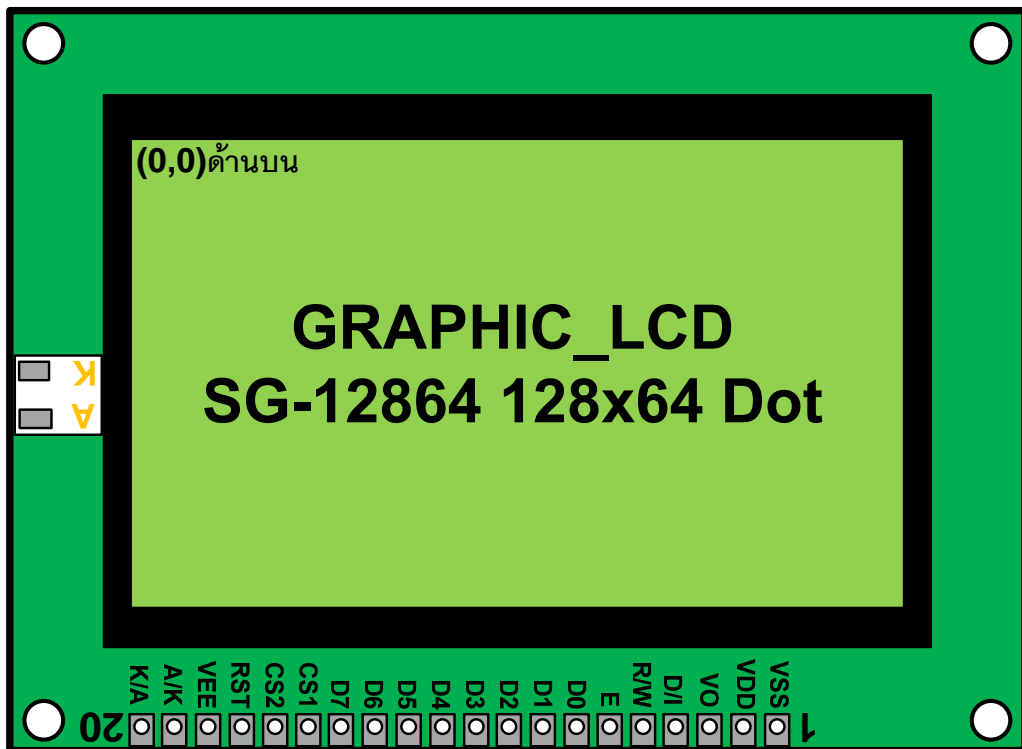
เรื่อง	หน้า
5. การต่อใช้งาน RS232/422 และ 485 แบบ 4 เส้น	30
5.1 การส่ง Command ผ่าน RS232	30
5.2 การส่ง Command ผ่าน RS422	30
5.3 การส่ง Command ผ่าน RS485 แบบ 4 Line	30
6. การ Set Up บอร์ด ET-REMOTE GLCD12864	31
7. ตัวอย่างโปรแกรม	32
7.1 ตัวอย่างการทดสอบส่งคำสั่งผ่านทาง <i>Hyper Terminal</i>	32
7.2 ตัวอย่างโปรแกรม ส่งคำสั่งด้วย <i>MCU</i>	34
Appendix A : สรุปตารางคำสั่ง	38
Appendix B : ตาราง ASCII Code สมอ.	42
Appendix C : DISPLAY GRAPHIC LCD 128x64	43
Appendix D : Circuit Board	44

ET- REMOTE GLCD12864 V1.0

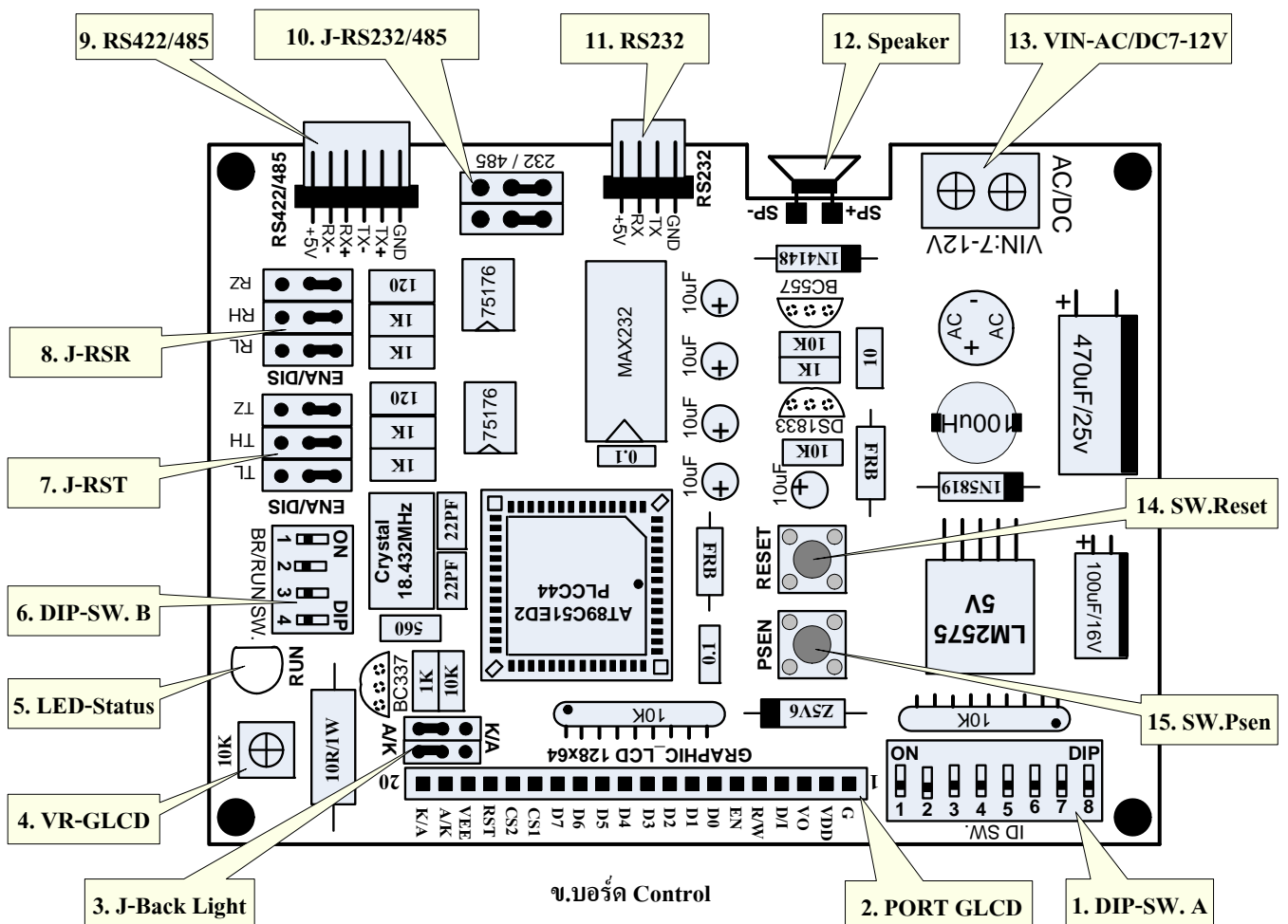
1. คุณสมบัติของบอร์ด ET-REMOTE GLCD12864 V1.0

- 1.1 เป็นชุด DISPLAY ขนาด 128x64 Dot พร้อมบอร์ดควบคุมการแสดงผล โดยใช้ส่งคำสั่งแบบ ASCII Command
- 1.2 สามารถส่งคำสั่งควบคุมการทำงานได้ทาง Port RS232,RS422 และ RS485 แบบ 4 เส้น
- 1.3 Baud Rate ในการสื่อสารเลือกได้โดย DIP SW. คือ 1200,2400,4800,9600,19200,38400,57600 และ 115200
- 1.4 สามารถกำหนด ID ให้กับบอร์ดด้วย DIP SW. ได้ตั้งแต่ 00-FF เพื่อใช้สื่อสารแบบ Network ด้วย RS485
- 1.5 สามารถสร้างเสียงตัวโน้ต และ ควบคุมการเปิดปิดของ Back Light ได้
- 1.6 มี VR ปรับความเข้มของ GLCD รวมทั้ง LED แสดงสถานะ การทำงานของคำสั่งนั้นๆ
- 1.7 มีระบบ Self-Test และ Demo อยู่ภายในตัว
- 1.8 ใช้แหล่งจ่ายไฟเลี้ยงบอร์ด AC/DC 7-12 V
- 1.9 สามารถ Invert DISPLAY ให้แสดงผลเป็นตรงข้ามจากที่แสดงอยู่ได้
- 1.10 มีการทำงาน 3 Mode คือ Text Mode , Graphic Mode และ Picture Mode ซึ่งสามารถทำงานร่วมกันได้
- 1.11 ใน Text Mode
 - สามารถแสดงผลได้ทั้งตัวอักษร ภาษาไทย-อังกฤษ ไม่ต้องเสียเวลาในการสร้าง Font เอง
 - ขนาดของตัวอักษรสำหรับภาษาอังกฤษ 5x7 Dot(กว้างxสูง) และสำหรับภาษาไทยขนาดสูง 7 Dot ส่วนความกว้างขึ้นอยู่กับ ตัวอักษรนั้นๆ ซึ่งจะทำได้ตัวอักษรที่สวยงาม โดย Font ที่แสดง จะเป็น Font เดียวกับ MS Sans Serif ของ Windows
 - สามารถกำหนดให้แสดงตัวอักษรในตำแหน่งใดๆบนหน้าจอได้ และโดยปกติหนึ่งหน้าจอจะแสดงข้อความได้ที่ 4 บรรทัดหรือมากกว่า ส่วนในหนึ่งบรรทัดจะแสดงตัวอักษรได้กี่ตัวนั้น จะขึ้นอยู่กับขนาดความกว้างของตัวอักษรที่ส่งมาในแต่ละตัวซึ่งจะมีความกว้างไม่เท่ากัน
 - สามารถลบตัวอักษร ในตำแหน่งใดๆบนหน้าจอได้
 - สามารถส่ง Text ข้อความได้ครั้งละ 200 ตัวอักษร (ตัวอักษร+สระ+วรรณยุกต์)
 - สามารถเลือกจัดข้อความได้ว่าจะให้ข้อความที่ส่งมา ชิดซ้าย,ชิดขวา หรืออยู่กึ่งกลางของบรรทัดนั้นๆ (ข้อความส่วนที่เกินบรรทัดจะถูกตัดทิ้ง)
 - สามารถเลือกได้ว่าเมื่อข้อความจบบรรทัดแล้วให้ขึ้นบรรทัดใหม่ให้โดยอัตโนมัติ
 - เมื่อข้อความเต็มหน้าจอข้อความส่วนที่เกินจะถูกตัดทิ้งไป และสามารถสั่งเคลียร์หน้าจอใหม่ได้
- 1.12 ใน Graphic Mode และ Picture Mode
 - สามารถสร้างรูป สามเหลี่ยม,สี่เหลี่ยม,วงกลม,วงรี,และเส้นตรงใดๆ ตามขนาดและ ตำแหน่งใดๆ บนหน้าจอตามที่ต้องการได้
 - สามารถ Fill รูปให้เป็นสีค่าทั้งรูปได้ หรือจะให้เป็นลายเส้นก็ได้
 - สามารถนำภาพจากภายนอกที่เป็น File แบบ BIT MAP ขนาด128x64 Dot มา Convert ให้เป็น Code เสียก่อนแล้วทำการส่ง Code นั้นให้กับบอร์ด ภาพนั้นก็就会被นำมาแสดงให้เห็นที่หน้าจอได้

2. โครงสร้างของบอร์ด ET-REMOTE GLCD12864 V1.0



ก. Graphic LCD 128x64 Dot



ข.บอร์ด Control

รูปที่ 2.1 แสดงโครงสร้างของบอร์ด ET-REMOTE GLCD12864 V1.0

สำหรับบอร์ด ET-REMOTE GLCD12864 V1.0 จะแบ่งโครงสร้างออกเป็น 2 ส่วน ส่วนแรกจะเป็นส่วนของจอแสดงผล Graphic LCD ขนาด 128x64 Dot โดยจะมีขาต่อใช้งาน 20 ขา มี Back Light แสดงในรูป ก. และส่วนที่ 2 จะเป็นส่วนของบอร์ดควบคุมสำหรับรับคำสั่งจากผู้ใช้งานทาง RS232/RS422 และ RS485 แบบ 4 เส้น แสดงในรูป ข. โดยหน้าที่ของส่วนต่างๆในบอร์ดควบคุมสามารถสรุปได้ดังนี้

หมายเลข 1 DIP-SW.A :เป็น DIP SW. สำหรับกำหนด ID ให้กับตัวบอร์ดสามารถกำหนดได้ตั้งแต่ ID '00'-'FF'

โดย ID นี้ จะมีประโยชน์เมื่อต่อใช้งานบอร์ดหลายๆบอร์ดในลักษณะของ Network ซึ่งจะใช้กับการสื่อสารแบบ RS485 โดยการ Set ID นั้นจะเป็นไปตามตารางด้านล่าง

ตารางที่ 2.1 DIP-SW.A สำหรับ SET ID. BOARD ('00' -'FF')

หมายเลข DIP-SW.A								ID CODE (ASCII- 2 byte)
S8	S7	S6	S5	S4	S3	S2	S1	
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	' 00 '
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	' 01 '
OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	' 02 '
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON	' 03 '
OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	' 04 '
OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	' 05 '
OFF	OFF	OFF	OFF	OFF	ON	ON	OFF	' 06 '
OFF	OFF	OFF	OFF	OFF	ON	ON	ON	' 07 '
OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	' 08 '
OFF	OFF	OFF	OFF	ON	OFF	OFF	ON	' 09 '
OFF	OFF	OFF	OFF	ON	OFF	ON	OFF	' 0A '
OFF	OFF	OFF	OFF	ON	OFF	ON	ON	' 0B '
OFF	OFF	OFF	OFF	ON	ON	OFF	OFF	' 0C '
OFF	OFF	OFF	OFF	ON	ON	OFF	ON	' 0D '
OFF	OFF	OFF	OFF	ON	ON	ON	OFF	' 0E '
OFF	OFF	OFF	OFF	ON	ON	ON	ON	' 0F '
OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	' 10 '
↓	↓	↓	↓	↓	↓	↓	↓	
ON	ON	ON	ON	ON	ON	OFF	OFF	' FC '
ON	ON	ON	ON	ON	ON	ON	OFF	' FE '
ON	ON	ON	ON	ON	ON	ON	ON	' FF '

สำหรับการอ่านค่า ID จาก DIP SW. นี้ เราจะอ่านในลักษณะของ Code BCD8421 โดยให้ S8-S5 เป็น 4 บิต High และ S4-S1 เป็น 4 บิต Low เมื่ออ่านค่าได้ ก็ให้นำค่า Code ที่ได้มาต่อกันก็จะได้ค่า ID ออกมา เวลาจะนำไปใช้งานก็ให้มอง ID นี้อยู่ในรูปของ ASCII CODE คือแยกออกเป็น 2 Byte เช่น ID '00' เวลาใช้ใน Command ก็จะสามารถแทนด้วย ASCII '00' หรือแทนด้วยเลขฐาน16 คือ 0x30,0x30 ได้ หรือ ID 'FC' = 0x46,0x43 เป็นต้น ซึ่งสามารถเทียบค่า ASCII กับเลขฐาน16 ได้จากตารางท้ายคู่มือ ถ้าผู้ใช้อ่านค่า ID จากการ DIP SW.A ไม่ถูก ก็สามารถดูค่า ID ได้จาก หน้าจอ GLCD โดยการเลื่อน DIP-SW.B หมายเลข4ไปที่ตำแหน่ง ON ที่หน้าจอ GLCD ก็จะแสดง ID ของบอร์ดให้เห็น

หมายเหตุ ทุกครั้งที่มีการเปลี่ยน ID บอร์ด ค่า ID ที่เปลี่ยนจะมีผลก็ต่อเมื่อมีการ Reset board หรือ เมื่อเลื่อน DIP-SW.B หมายเลข4 ไป ที่ตำแหน่ง ON แล้วรอนจนกว่าจะเห็นค่า ID ที่หน้าจอ GLCD มีการ Update ค่าใหม่

หมายเลข 2 PORT GLCD : ขั้วต่อนี้จะมีขนาด 20 Pin ใช้สำหรับต่อ Graphic LCD โดยมีการจัดเรียงขาตามรูปด้านบน

หมายเลข 3 J-Back Light : จะเป็น Jumper คู่ 2 ตัวใช้สำหรับสลับขั้ว Back Light ของ GLCD ในขาที่ 19 และ 20 เนื่องจาก GLCD บางรุ่นจะเรียงขา ไว้สลับกับที่เรียงไว้บนบอร์ด โดยให้ทำการ Set ดังนี้ ถ้าขา 19 เป็นขั้ว A และ ขา 20 เป็นขั้ว K ให้ Set Jumper ทั้งสองมาทางด้าน A/K แต่ถ้าขา 19 เป็นขั้ว K และขา 20 เป็นขั้ว A ให้ Set Jumper ทั้งสองนี้มาทางด้าน K/A แทน

หมายเลข 4 VR-GLCD : เป็น VR ปรับค่า ใช้สำหรับปรับความเข้มของจอ GLCD ที่นำมาต่อใช้งาน

หมายเลข 5 LED Status : LED นี้จะมี 2 หน้าทีคือ 1.จะติดเมื่อกำลังทำงานใน Self-Test Mode และดับเมื่อทำงานเสร็จ และจะติดอีกครั้ง เมื่อกลับมาเริ่มต้นในโหมดนี้อีก 2. ใน Run Mode เมื่อผู้ใช้ทำการส่งคำสั่งมายังบอร์ดถูกต้องและบอร์ดเริ่มทำคำสั่งที่ส่งมา LED ก็จะติด และจะดับลงเมื่อกระทำคำสั่งนั้นเรียบร้อยแล้ว ถ้าเกิด LED ไม่ติด หรือติดค้างไม่ยอมดับนั้นแสดงว่า คำสั่งที่ส่งมาผิดรูปแบบที่กำหนด หรือ กระทำคำสั่งที่ส่งมาไม่สมบูรณ์ ก่อนจะเริ่มส่งคำสั่งใหม่ควรส่ง 0x0D หรือ Enter เพื่อ Clear Status LED ก่อน

หมายเลข 6 DIP-SW.B : เป็น DIP SW. สำหรับกำหนด Baud Rate ในการรับส่งข้อมูล ให้กับตัวบอร์ด และใช้เลือก โหมดการทำงาน ระหว่าง Self-Test Mode และ Run Mode โดย SW. หมายเลข 1-หมายเลข3 จะใช้สำหรับ Set ค่า Baud Rate ซึ่งสามารถเลือกได้ 8 ค่า ตามตารางที่2.3 ส่วน SW. หมายเลข4 จะใช้ สำหรับเลือก โหมดการทำงานระหว่าง Self-Test Mode และ Run Mode แสดงดังตารางที่2.2

ตารางที่2.2 DIP-SW.B สำหรับ เลือกโหมดการทำงาน

หมายเลข DIP-SW.B	หน้าที่
S4	
OFF	Start การทำงานใน Run Mode เพื่อรอรับคำสั่งจากผู้ใช้
ON	Start การทำงานใน Self-Test Mode เพื่อใช้ทดสอบการทำงานของบอร์ด และจอ GLCD รวมทั้ง แสดงค่า ID และค่า Baud Rate ที่ Set ให้กับบอร์ด

ตารางที่ 2.3 DIP-SW.B สำหรับ SET Baud Rate (1200-115200 bit/s)

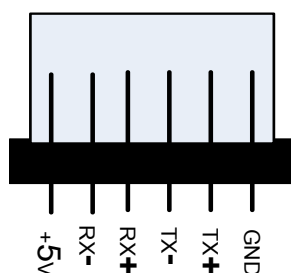
หมายเลข DIP-SW.B			Baud Rate (bit/s)
S3	S2	S1	
OFF	OFF	OFF	1200
OFF	OFF	ON	2400
OFF	ON	OFF	4800
OFF	ON	ON	9600
ON	OFF	OFF	19200
ON	OFF	ON	38400
ON	ON	OFF	57600
ON	ON	ON	115200

การตั้งค่า Baud Rate นอกจากเทียบตำแหน่ง DIP-SW. จากตารางแล้ว ผู้ใช้ยังสามารถตั้งค่า Baud Rate ที่ Set ให้กับตัวบอร์ด ได้อีกทางหนึ่งผ่านทางจอ GLCD โดยการ Set DIP SW.B หมายเลข 4 ไปที่ตำแหน่ง ON เพื่อเข้าสู่โหมด Self-Test ค่า Baud Rate ก็จะแสดงให้เห็น

หมายเหตุ ทุกครั้งที่มีการเปลี่ยนค่า Baud Rate ค่าที่เปลี่ยนจะมีผลก็ต่อเมื่อมีการ Reset board หรือ เมื่อเลื่อน DIP-SW.B หมายเลข 4 ไปที่ตำแหน่ง ON แล้วรอนจนกว่าจะเห็นค่า Baud Rate ที่หน้าจอ LCD มีการ Update ค่าใหม่

หมายเลข 7 , 8 J-RST, J-RSR : เป็น Jumper สำหรับเพิ่มระยะการสื่อสารแบบ RS422/485 ให้ได้ไกลขึ้นกว่าปกติ โดยให้ Set Jumper RL,RH,RZ และ TL,TH,TZ มาทางด้าน Enable

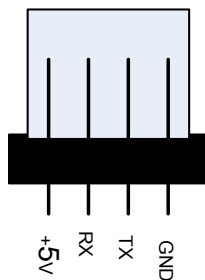
หมายเลข 9 RS422/485 : เป็น Connector 6 PIN สำหรับต่อสายสัญญาณ RS422/485 เมื่อต้องการจะสื่อสารหรือส่ง Command จากภายนอกเข้ามาควบคุมบอร์ด โดยมีการจัดเรียงขั้วต่อ ดังรูปที่ 2.2



รูปที่ 2.2 แสดงการจัดเรียงขั้วต่อ RS422/485

หมายเลข 10 J-RS232/485 : เป็น Jumper คู่ 2 ตัว ใช้สำหรับสลับรูปแบบการสื่อสาร ระหว่าง RS232 หรือ RS422/485 โดย เมื่อใช้การสื่อสารแบบ RS232 ก็ให้ Set Jumper ทั้ง 2 มาทางด้าน RS232 แต่ถ้าใช้การสื่อสารแบบ RS422/485 ก็ให้ Set Jumper ทั้ง 2 มาทางด้าน RS422/485

หมายเลข 11 RS232 : เป็น Connector 4 PIN สำหรับต่อสายสัญญาณ RS232 เมื่อต้องการจะสื่อสารหรือ ส่ง Command จากภายนอกเข้ามาควบคุมบอร์ด โดยมีการจัดเรียงขั้วต่อดังรูปที่ 2.3



รูปที่ 2.3 แสดงการจัดเรียงขาหัวต่อ RS232

หมายเลข 12 Speaker : ถ้าโพงนี้จะมีเสียงดังขึ้นเมื่อเริ่มจ่ายไฟให้กับบอร์ด และเมื่อเข้าสู่โหมด Self-Test นอกจากนี้ ผู้ใช้ยังสามารถส่งคำสั่งมาควบคุมให้เกิดเสียงดังขึ้นที่ลำโพงตามจังหวะตัวโน้ต ได้อีกด้วย

หมายเลข 13 VIN-AC/DC 7-12V : เป็นหัวต่อไฟเลี้ยงบอร์ด โดยรับแรงดันได้ทั้ง AC และ DC ที่ค่า 7-12 V (ไม่มีขั้ว)

หมายเลข 14 SW.Reset : เป็นสวิตช์ สำหรับ Reset MCU ให้เริ่มทำงานใหม่ เมื่อเกิด กรณีบอร์ด แสงค์ และยังไม่ใช้ ร่วมกับ SW.PSEN เพื่อ Download โปรแกรม เมื่อจะ UP Date Firm ware

หมายเลข 15 SW.Psen : สวิตช์นี้จะใช้งานร่วมกับ SW.Reset เพื่อใช้ Download โปรแกรม ในการ Up Date Firm ware ใหม่ โดยกด SW.PSEN ค้างไว้ แล้วตามด้วยการกด SW.Reset จากนั้นก็ปล่อย SW. Reset และตามด้วยการปล่อย SW.PSEN เป็นลำดับสุดท้าย แล้วถึงจะเริ่ม Download Firm ware ใหม่ได้ ซึ่งโดยปกติ แล้วกระบวนการนี้ทางบริษัท ETT จะเป็นผู้จัดการในส่วนของการ โปรแกรม Firm ware ให้เอง

3. การทำงานของบอร์ด ET-REMOTE GLCD12864

เมื่อเริ่มจ่ายไฟให้กับบอร์ดจะมีเสียงดนตรีในการเปิดเครื่องดังขึ้น จากนั้นตัวบอร์ดก็จะทำการ Set ค่า ID และค่า Baud Rate ตามที่ผู้ใช้ได้ Set ไว้จาก DIP-SW.A และ DIP-SW.B ตามลำดับ หลังจากเสียงดนตรีจบลง ตัวบอร์ดก็จะส่ง Respond Command Sync Byte ออกมาทาง RS232/422/485 ตามรูปแบบด้านล่าง เพื่อบอกให้ผู้ใช้ทราบว่า ตัวบอร์ดพร้อมเริ่มรับคำสั่งแล้ว ซึ่งถ้าผู้ใช้ไม่ตรวจสอบ Sync Byte ก่อนตอนเริ่มเปิดเครื่อง อาจจะใช้การ delay หน่วงเวลาของบอร์ดที่จะทำการส่งคำสั่งไว้ก่อนก็ได้ มิฉะนั้นแล้ว คำสั่งที่ส่งมาอาจจะไม่มีผลใดๆถ้าตัวบอร์ด GLCD ยังไม่พร้อมรับคำสั่ง การตรวจสอบ Sync Byte นี้จะใช้ได้กรณีที่ใช้บอร์ด GLCD เพียงบอร์ดเดียวเท่านั้น ถ้าต่อร่วมกันหลายบอร์ดจะทำให้ Respond Sync Byte ถูกส่งออกมาพร้อมกัน จะทำให้บอร์ดตัวรับอ่านข้อมูลผิดพลาดได้

* xx=OK < ----- Respond Sync Byte Command 6 Byte

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

หลังจากที่ตัวบอร์ดส่ง Command Sync byte ออกมาแล้ว ก็จะเข้าสู่โหมดการทำงานตามที่ผู้ใช้กำหนดจาก DIP-SW.B หมายเลข 4 โดยมีอยู่ด้วยกัน 2 โหมดคือ

3.1 Self-Test Mode : เมื่อ DIP-SW.B หมายเลข 4 อยู่ในตำแหน่ง ON โปรแกรมก็จะเข้าสู่การทำงานในโหมดนี้ โดยวัตถุประสงค์ของโหมดนี้ก็เพื่อใช้ทดสอบการทำงานของบอร์ด Control และตัว GLCD รวมทั้งแสดงค่า ID และ Baud Rate ที่ผู้ใช้กำหนด ให้ผู้ใช้ได้เห็นซึ่งจะช่วยให้ไม่ต้องเสียเวลาในการเทียบตำแหน่งของ DIP SW. จากตารางข้างต้น

เมื่อโปรแกรมเริ่มต้นเข้ามาทำงานที่โหมคนีจะมีเสียงดนตรีดังขึ้น และ LED ก็จะติดจากนั้นที่หน้าจอ GLCD ก็ จะแสดงค่า ID และ Baud Rate ให้เห็นที่ค่าเวลาหนึ่งจากนั้นเมื่อ LED ดับลง แสดงว่าจบการทำงานของโหมคนี โปรแกรมจะ ถูกกำหนดให้วนกลับมาเริ่มต้นทำงานใหม่อีกครั้ง ทรานซิสเตอร์ DIP-SW.B หมายเลข 4 ยังคงอยู่ในตำแหน่ง ON ซึ่งในขณะที่ อยู่ในโหมคนีผู้ใช้สามารถ Set ค่า ID และ Baud Rate ได้ เมื่อเห็นค่าที่ Set บนหน้าจอ GLCD มีการเปลี่ยนแปลงแล้วนั้น แสดงว่าค่า ID และ Baud Rate มีการ Update ค่าเรียบร้อยแล้ว และสามารถกลับไปยัง RUN Mode เพื่อรับคำสั่งจากผู้ใช้งาน โดยใช้ค่า ID และ Baud Rate ตัวใหม่นี้ได้เลย

3.2 RUN-Mode : เมื่อ DIP-SW.B หมายเลข 4 อยู่ในตำแหน่ง OFF โปรแกรมก็จะเข้าสู่การทำงานในโหมคนี ซึ่งใน โหมคนีจะทำหน้าที่รับคำสั่งจากผู้ใช้งานทาง RS232 , RS422 และ RS485 แบบ 4 เส้น เมื่อเข้าสู่ในโหมคนีหน้าจอ GLCD ก็ จะถูก Clear ให้ว่าง ซึ่งผู้ใช้สามารถส่งคำสั่งมาควบคุมการทำงานของ GLCD ได้เลย โดยรูปแบบของคำสั่งต่างๆและตัวอย่าง การใช้งาน สามารถดูได้จากหัวข้อที่ 4 เมื่อผู้ใช้งานทำการส่งคำสั่งให้กับบอร์ด ถ้าคำสั่งนั้นถูกต้องตามรูปแบบที่กำหนดและตัว บอร์ดเริ่มกระทำคำสั่งที่ส่งมาจะสังเกตเห็น LED ติด เมื่อคำสั่งนั้นถูกกระทำสมบูรณ์แล้ว LED ก็จะดับ ถ้าเกิดส่งคำสั่ง มาแล้ว LED ไม่ติด แสดงว่าคำสั่งที่ส่งมาผิดรูปแบบที่กำหนด หรือ LED ติดค้างแสดงว่าการทำงานของคำสั่งนั้นผิดพลาดให้ ทำการส่งค่า 0x0D หรือ กดคีย์ Enter เพื่อทำการ Reset คำสั่ง หรือ Reset ตัวบอร์ดใหม่ถ้า LED ยังติดค้างอยู่

ทุกครั้งที่โปรแกรมกระทำคำสั่งเสร็จสิ้นแล้วจะส่ง Respond End Command ออกมาให้กับผู้ใช้งานทาง RS232/422 หรือ 485 ด้วย โดยมีรูปแบบของ Respond End Command ที่ส่งออกมาดังนี้

* xx#yy=OK

< ----- Respond End Command 9 Byte

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

yy = หมายเลขคำสั่งที่ส่ง แสดงเป็น ASCII 2 หลัก ('00'-'34')

ซึ่ง Command ที่ส่งออกมานี้จะมีประโยชน์กับผู้ใช้งาน โดยจะเป็นตัวบอกให้ผู้ใช้งานสามารถส่งคำสั่งต่อไปได้ ซึ่งเมื่อผู้ใช้งานได้ส่ง คำสั่งแรกไปแล้วผู้ใช้งานจะต้องทำการตรวจสอบ End Command ก่อนที่จะทำการส่งคำสั่งต่อไปให้กับบอร์ดเสมอ มิฉะนั้น คำสั่งที่ส่งไปครั้งที่ 2 นี้จะไม่ถูกกระทำ และอาจจะทำให้โปรแกรมทำคำสั่งแรกผิดพลาดได้ด้วย เนื่องจากเวลาการทำงาน ของแต่ละคำสั่งนั้นไม่เท่ากัน จึงไม่แนะนำให้ใช้การ delay ก่อนส่งคำสั่งต่อไป ให้ใช้การตรวจสอบ End Command แทน ซึ่งดูตัวอย่างโปรแกรมได้ในหัวข้อของ ตัวอย่างโปรแกรม

หมายเหตุ ในขณะที่ทำงานอยู่ใน RUN Mode หรือ Self-Test Mode ผู้ใช้งานสามารถเลื่อน DIP-SW.B หมายเลข 4 ทำการเปลี่ยนโหมคกลับไป กลับมาได้ ใน RUN Mode ถ้ามีการเปลี่ยนโหมคขณะที่โปรแกรมยังทำคำสั่งไม่สมบูรณ์ โปรแกรมก็จะทำคำสั่งจนสมบูรณ์ก่อนถึง จะทำการเปลี่ยนโหมคให้

4. รูปแบบและการใช้งานคำสั่งต่างๆที่บอร์ด ET-REMOTE GLCD12864 รองรับ

รูปแบบของ Command และ Data ที่ใช้นั้นจะอยู่ในรูปของ ASCII Code ทั้งหมด ซึ่งสามารถแทนด้วยสัญลักษณ์ ASCII เช่น '*' หรือแทนด้วย ASCII Code ซึ่งก็จะเท่ากับ 0x2A , '0'=0x30 , 'A' = 0x41 เป็นต้น

ในส่วนของ Byte Start ให้แทนด้วยค่า 0x1B (Key Esc) และ Byte จบ ให้แทนด้วย 0x0D (Key Enter) เนื่องจาก 2 Byte นี้ไม่มีสัญลักษณ์ ASCII ให้ใช้แทน จึงแทนด้วย ASCII CODE ได้เท่านั้น ในกรณีของค่า ID ที่เป็นตัวอักษร 'A-F' ให้ใช้ตัวพิมพ์ใหญ่เท่านั้น

สำหรับคำสั่งที่ใช้ติดต่อกับบอร์ด ET-REMOTE GLCD12864 นี้จะมีอยู่ด้วยกันทั้งหมด 21 คำสั่ง เพื่อให้ง่ายต่อการทำความเข้าใจเราจะขอแบ่งคำสั่งออกเป็น 4 กลุ่มตามหน้าที่การใช้งาน ดังนี้

4.1 กลุ่มคำสั่งที่เกี่ยวข้องกับการ Control Display และอุปกรณ์บนบอร์ด

COMMAND '00' (Clear Screen)

เป็นคำสั่งสำหรับ Clear หน้าจอ GLCD ให้ว่าง โดยในสภาวะปกติเมื่อใช้คำสั่งนี้หน้าจอจะถูกเคลียร์ให้เป็นสีขาว (ว่าง) แต่ถ้ามีการ Set ค่า Invert ให้เป็น '1' ด้วยคำสั่ง '01' ไว้ เมื่อใช้คำสั่งนี้หน้าจอจะถูก Clear ให้เป็นสีดำแทน ซึ่งมีรูปแบบคำสั่งดังในตาราง

Start	ID	Mark1	Command	Mark2	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	00	=	Enter (0x0D)
Respond Command จากบอร์ด					
*	00-FF	#	00	=	OK

Ex1. ให้ ID = '9A'

```
main()
{
  char esc = 0x1B , enter = 0x0D ;

  printf("%c9A#00=%c",esc,enter) ; // Clear Screen
}
```

COMMAND '01' (Invert Screen)

เป็นคำสั่งสำหรับ Invert ข้อมูลก่อนที่จะนำไปแสดงออกบนหน้าจอให้มีสถานะเป็นตรงกันข้ามกับที่เป็นอยู่ โดยกำหนดที่ช่อง Invert ของคำสั่ง ถ้ากำหนดเป็น '1' จะเป็นการ Invert (พื้นหลังสีดำ) ถ้ากำหนดเป็น '0' จะเป็นสภาวะปกติ (พื้นหลังสีขาว(default)) เมื่อใช้คำสั่งนี้โดยให้ช่อง Invert เป็น '1' แล้ว ข้อมูลที่หน้าจอจะยังไม่เปลี่ยนจนกว่าจะมีการส่งข้อมูลออกมา Plot ที่หน้าจอถึงจะเห็นผลการ Invert หรือ ถ้าที่หน้าจอมีข้อมูลแสดงอยู่แล้ว และต้องการ Invert หน้าจอที่แสดงอยู่ ก็ให้ส่งคำสั่งนี้ออกไปก่อน แล้วตามด้วยคำสั่ง Display Screen (คำสั่งหมายเลข '02') ก็จะเห็นผลการ Invert โดยไม่ต้องส่งข้อมูลเดิมออกไปซ้ำใหม่ ถ้าต้องการยกเลิกการ Invert ก็ให้ส่งคำสั่งนี้ออกไปอีกครั้งหนึ่งโดยให้กำหนดค่าในช่อง Invert เป็น '0' รูปแบบของคำสั่งจะแสดงดังในตาราง

Start	ID	Mark1	Command	Mark2	Invert	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	01	=	0-1	Enter (0x0D)
Respond Command จากบอร์ด						
*	00-FF	#	01	=		OK

ช่อง Invert = '0' : Normal (default) , '1' : Invert

Ex2. ให้ ID = '5F'

```
main()
{
    char esc = 0x1B , enter = 0x0D ;

    printf("%c5F#01=1%c",esc,enter) ; // Invert Display

    printf("%c5F#01=0%c",esc,enter) ; // ยกเลิก Invert Display
}
```

COMMAND '02' (Display Screen)

เป็นคำสั่งสำหรับ แสดงข้อมูลออกหน้าจอ GLCD ซึ่งคำสั่งนี้จะมีจุดมุ่งหมายเพื่อใช้คู่กับคำสั่ง Invert คือเมื่อส่งคำสั่ง Invert ออกไปแล้วจะเห็นว่าข้อมูลเดิมบนหน้าจอจะยังไม่ถูก Update ถ้าจะให้ Update ก็จะต้องส่งข้อมูลเดิมเข้าไปอีกรอบหนึ่ง ซึ่งคำสั่งนี้จะช่วยแก้ไขในส่วนนี้ โดยหลังจากส่งคำสั่ง Invert ออกไปแล้วก็ให้ตามด้วยคำสั่งนี้ ก็จะทำให้หน้าจอนั้นถูก Update ทันทีโดยไม่ต้องมาเสียเวลาส่งข้อมูลเดิมออกไปอีก รูปแบบคำสั่งแสดงดังตาราง

Start	ID	Mark1	Command	Mark2	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	02	=	Enter (0x0D)
Respond Command จากบอร์ด					
*	00-FF	#	02	=	OK

Ex3. ให้ ID = 'AA'

```
main()
{
    char esc = 0x1B , enter = 0x0D ;

    printf("%cAA#02=%c",esc,enter) ; //Display Screen
}
```

COMMAND '03' (On/Off Back Light)

เป็นคำสั่งสำหรับ ON หรือ OFF Back Light ของ GLCD โดยการกำหนดที่ช่อง Back Light ถ้าเป็น '1' = ON(default) ถ้าเป็น '0' = OFF ซึ่งก่อนใช้คำสั่งนี้ให้ตรวจสอบ Jumper หมายเลข 3 ด้วยว่ามีกร Set ตำแหน่งขั้ว Back Light ของ GLCD ไว้ถูกต้องหรือไม่ รูปแบบของคำสั่งแสดงดังตาราง

Start	ID	Mark1	Command	Mark2	Back Light	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	03	=	0-1	Enter (0x0D)
Respond Command จากบอร์ด						
*	00-FF	#	03	=	OK	

ช่อง Back Light = '0' : OFF Back Light , '1' : ON Back Light (default)

Ex4. ให้ ID = '55'

```
main()
{
    char esc = 0x1B , enter = 0x0D ;

    printf("%c55#03=1%c" , esc, enter) ; //Back Light ON
    printf("%c55#03=0%c" , esc, enter) ; //Back Light OFF
}
```

COMMAND '04' (Sound)

เป็นคำสั่งสำหรับ สร้างเสียงออกลำโพงเป็นเสียงตัวโน้ต โด เร มี ฟา ซอล ลา ที่ สามารถเลือกเสียงตัวโน้ตและกำหนดความยาวเสียงได้ โดยกำหนดเสียงตัวโน้ตที่ช่อง Note ได้ตั้งแต่ค่า '0-7' และกำหนดความยาวของเสียงในช่อง Delay ได้ตั้งแต่ค่า '0-9' รูปแบบคำสั่งแสดงดังตาราง

Start	ID	Mark1	Command	Mark2	Note	Mark3	Delay	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	04	=	0-7	,	0-9	Enter (0x0D)
Respond Command จากบอร์ด								
*	00-FF	#	04	=	OK			

ช่อง Note = '0' : No Sound , '1' : เสียง โด (524 Hz) , '2' : เสียง เร (587 Hz) , '3' : เสียง มี (659 Hz)
 '4' : เสียง ฟา (698Hz) , '5' : เสียง ซอล (784 Hz) , '6' : เสียง ลา (880 Hz) , '7' : เสียง ที (988 Hz)

ช่อง Delay = '0' : ความยาวเสียงสั้นสุด และความยาวเสียงเพิ่มขึ้นเรื่อยๆตามหมายเลข โดย '9' : ความยาวเสียงยาวสุด

Ex5. ให้ ID = '00'

```
main()
{
    char esc = 0x1B , enter = 0x0D ;

    printf("%c00#04=1,4%c" , esc, enter) ; //สร้างเสียง โด ความยาว 4
}
```

4.2 กลุ่มคำสั่งที่เกี่ยวข้องกับ Graphic Mode

คำสั่งในกลุ่มนี้จะเป็นคำสั่งในการวาดรูป สามเหลี่ยม , สี่เหลี่ยม , รูปหลายเหลี่ยม (ไม่เกิน 9 เหลี่ยม) , วงกลม , วงรี , จุด และเส้นตรง ในแนวนอน แนวตั้ง และเส้นทแยง ซึ่งรูปเหล่านี้สามารถวาดให้ปรากฏที่จุดใดๆบนหน้าจอก็ได้ด้วยการกำหนดพิกัดจุด x,y ของรูปให้กับคำสั่งที่ใช้ โดยค่าที่ใช้กำหนดให้กับพิกัด x จะต้องเป็นค่า ASCII ขนาด 3 Byte ที่เป็นตัวเลข 0-9 ('000-127') ส่วนพิกัด y จะต้องเป็น ASCII ขนาด 2 Byte ที่เป็นตัวเลข 0-9 ('00-31') เช่นกัน ซึ่งผู้ใช้สามารถดูค่าตำแหน่งพิกัดของจุดต่างๆบนหน้าจอได้จากตารางจอภาพในภาคผนวก ซึ่งจะช่วยให้สะดวกขึ้นในการกำหนดตำแหน่งพิกัดจุด โดยให้ผู้ใช้มองที่มุมบนด้านซ้ายของหน้าจอ GLCD เป็นพิกัดเริ่มต้น $x,y = (0,0)$

ลักษณะของรูปที่วาดจะเป็นในลักษณะของ ลายเส้น และสามารถ Fill รูปให้เป็นสีค่าทั้งรูปได้ ยกเว้นรูปหลายเหลี่ยมจะ Fill เป็นสีค่าไม่ได้แต่จะสามารถกำหนดความหนาของเส้นที่วาดได้ สำหรับการ Fill รูปให้เป็นสีค่านั้นจะใช้เวลาในการ Fill พอสมควรโดยเฉพาะรูปสามเหลี่ยมซึ่งอาจจะทำให้เสียเวลาในการที่จะส่งคำสั่งต่อไปได้

การ Plot รูปออกหน้านั้นจะเป็นในลักษณะการ Plot ไปทับข้อมูลเก่าที่แสดงอยู่บนหน้าจอ จุดใดที่อยู่บนหน้าจอที่ไม่ถูก Plot รูปใหม่ไปทับ จะไม่ถูกลบทิ้ง จะยังคงแสดงให้เห็นอยู่ ซึ่งจะป็นข้อดีที่ผู้ใช้จะสามารถเขียนข้อความลงไปบนรูปที่วาดได้เพียงแต่จะต้องกำหนดพิกัดของรูปหรือข้อความอย่าให้ไปทับพิกัดเดิมในตำแหน่งที่เราต้องการจะใช้แสดงผล

COMMAND '10' (Plot Circle)

เป็นคำสั่งสำหรับวาดรูปวงกลมแบบเป็นลายเส้น ค่า พารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

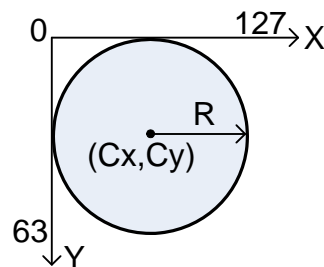
1. ค่าพิกัดจุด Center ในช่อง Cx (3 Byte: '000-127') และช่อง Cy (2 Byte: '00-63')
2. ค่ารัศมีของวงกลม ในช่อง Radius (2 Byte: '00-31')
3. ค่า Fill ในช่อง Fill เมื่อ '1' = Fill (แสดงเป็นสีค่าทั้งรูป) , '0' = ไม่ Fill (แสดงเป็นลายเส้นสีดำ) ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง'01') แล้วตามด้วยคำสั่งนี้ ลักษณะของรูปก็จะแสดงตรงกันข้ามกับที่กล่าวไว้ข้างต้น

Start	ID	Mk1	Command	Mk2	Cx	Mk3	Cy	Mk4	Radius	Mk5	Fill	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	10	=	000-127	,	00-63	,	00-31	,	0-1	Enter (0x0D)
Respond Command จากบอร์ด												
*	00-FF	#	10	=	OK							

Ex5. ให้ ID = '01' Plot วงกลมที่มีจุดศูนย์กลางอยู่ที่พิกัด Cx= dot ที่ 60 ,

Cy= dot ที่ 30 และ มีรัศมียาว 20 dot ไม่ Fill รูป

```
main()
{
    char esc = 0x1B , enter = 0x0D ;
    printf("%c01#10=060,30,20,0%c" , esc, enter) ; // Plot วงกลม
}
```



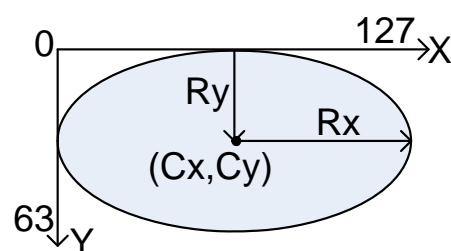
COMMAND '11' (Plot Ellipse)

เป็นคำสั่งสำหรับวาดรูปวงรีแบบเป็นลายเส้น ถ้าต้องการรูปวงรีในแนวนอนให้กำหนดค่า $R_x > R_y$ แต่ถ้าต้องการรูปวงรีในแนวตั้งให้กำหนดค่า $R_y > R_x$ โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

1. ค่าพิกัดจุด Center ในช่อง Cx (3 Byte: '000-127') และช่อง Cy (2 Byte: '00-63')
2. ค่ารัศมีของวงรี แนวแกน X ในช่อง Rx (2 Byte: '00-64')
3. ค่ารัศมีของวงรี แนวแกน Y ในช่อง Ry (2 Byte: '00-32')
4. ค่า Fill ในช่อง Fill เมื่อ '1' = Fill (แสดงเป็นสีดำทั้งรูป) , '0' = ไม่ Fill (แสดงเป็นลายเส้นสีดำ) ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง'01') แล้วตามด้วยคำสั่งนี้ ลักษณะของรูปก็จะแสดงตรงกันข้ามกับที่กล่าวไว้ข้างต้น

Start	ID	Mk1	Command	Mk2	Cx	Mk3	Cy	Mk4	Rx	Mk5	Ry	Mk6	Fill	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	11	=	000-127	,	00-63	,	00-64	,	00-32	,	0-1	Enter (0x0D)
Respond Command จากบอร์ด														
*	00-FF	#	11	=	OK									

Ex6. ให้ ID = '02' Plot วงรีที่มีจุดศูนย์กลางอยู่ที่พิกัด Cx= dot ที่ 63 , Cy = dot ที่ 31 และ มีรัศมีแนวแกน x ยาว Rx=64 dot และรัศมีแนวแกน y ยาว Ry = 32 dot ไม่ Fill รูป



```
main()
{
    char esc = 0x1B , enter = 0x0D ;
    printf( "%c02#11=063,31,64,32,0%c" , esc, enter ) ; // Plot วงรี
}
```

COMMAND '12' (Plot Dot)

เป็นคำสั่งสำหรับ Plot จุด 1 จุด บนหน้าจอในตำแหน่งที่กำหนด โดยส่งคำสั่งไป 1 ครั้งจะเท่ากับ Plot 1 จุด โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

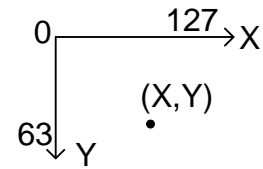
1. ค่าพิกัดแกน X ในช่อง X (3 Byte: '000-127')
2. ค่าพิกัดแกน Y ในช่อง Y (2 Byte: '00-63')

ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง'01') แล้วตามด้วยคำสั่งนี้ ลักษณะของจุดก็จะแสดงเป็นจุดสีขาว

Start	ID	Mark1	Command	Mark2	X	Mark3	Y	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte
Esc (0x1B)	00-FF	#	12	=	000-127	,	00-63	Enter (0x0D)
Respond Command จากบอร์ด								
*	00-FF	#	12	=	OK			

Ex7. ให้ ID = '03' Plot จุดที่มีพิกัด อยู่ที่ X= dot ที่ 30 , Y= dot ที่ 15

```
main()
{
    char esc = 0x1B , enter = 0x0D ;
    printf( "%c03#12=030,15%c" ,esc,enter) ; //Plot dot 1 dot
}
```



COMMAND '13' (Plot Triangle)

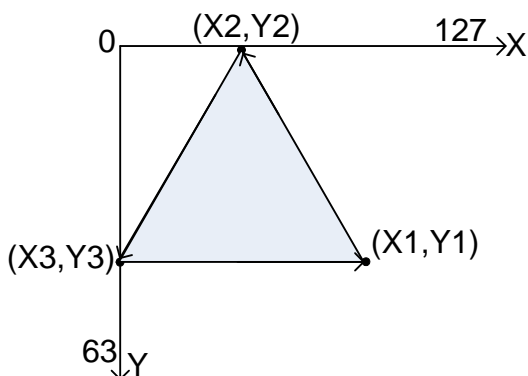
เป็นคำสั่งสำหรับวาดรูปสามเหลี่ยมแบบเป็นลายเส้น ซึ่งคำสั่งนี้ผู้ใช้จะต้องกำหนดพิกัดจุด (x,y) ของรูปสามเหลี่ยมทั้งหมด 3 จุด โดยลักษณะการทำงานของโปรแกรม จะลากเส้นจากพิกัดจุดที่ 1 ไปหาพิกัดจุดที่ 2 และลากเส้นจากพิกัดจุดที่ 2 ไปหาพิกัดจุดที่ 3 สุดท้ายจะลากเส้นจากพิกัดจุดที่ 3 ไปหาพิกัดจุดที่ 1 ก็จะได้รูป 3 เหลี่ยมออกมา ดังนั้น ถ้าผู้ใช้ต้องการได้รูปสามเหลี่ยมออกมาในลักษณะใดก็ให้กำหนดพิกัดจุดตามลักษณะการวาดของโปรแกรม ที่กล่าวไปข้างต้น ค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

- ค่าพิกัดจุดที่ 1 ในช่อง X1 (3 Byte: '000-127') และช่อง Y1 (2 Byte: '00-63')
- ค่าพิกัดจุดที่ 2 ในช่อง X2 (3 Byte: '000-127') และช่อง Y2 (2 Byte: '00-63')
- ค่าพิกัดจุดที่ 3 ในช่อง X3 (3 Byte: '000-127') และช่อง Y3 (2 Byte: '00-63')
- ค่า Fill ในช่อง Fill เมื่อ '1' = Fill (แสดงเป็นสีดำทั้งรูป) , '0' = ไม่ Fill (แสดงเป็นลายเส้นสีดำ) ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง '01') แล้วตามด้วยคำสั่งนี้ ลักษณะของรูปก็จะแสดงตรงกันข้ามกับที่กล่าวไว้ข้างต้น สำหรับการ Fill ของรูปสามเหลี่ยมนี้จะใช้เวลา ในการ Fill ซ้ำกว่ารูปอื่นๆและอาจจะ Fill ได้ไม่เต็มรูป มีเป็นจุดสีขาวบ้าง

Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	Mk6
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte
Esc (0x1B)	00-FF	#	13	=	000-127	,	00-63	,	000-127	,	00-63	,
Respond Command จากบอร์ด												
*	00-FF	#	13	=	OK							

(ต่อ)

X3	Mk7	Y3	Mk8	Fill	END
3 byte	1 byte	2byte	1 byte	1 byte	1 byte
000-127	,	00-63	,	0-1	Enter (0x0D)



Ex8. ให้ ID = '04' Plot รูปสามเหลี่ยม ที่มีพิกัด (X1= dot ที่ 100 ,Y1 = dot ที่ 60) , (X2 = dot ที่ 50,Y2 = dot ที่ 0), (X3 = dot ที่ 0 , Y3 = dot ที่ 60) ไม่ Fill รูป

```
main()
{
    char esc = 0x1B , enter = 0x0D ;
    printf( "%c04#13=100,60,050,00,000,60,0%c" ,esc,enter) ; //Plot สามเหลี่ยม
}
```

COMMAND '14' (Plot Rectangle)

เป็นคำสั่งสำหรับวาดรูป สี่เหลี่ยมผืนผ้า หรือ สี่เหลี่ยมจัตุรัส แบบเป็นลายเส้น ซึ่งคำสั่งนี้ผู้ใช้จะต้องกำหนดพิกัดจุด (x,y) ของรูปสี่เหลี่ยมทั้งหมด 2 จุด คือ พิกัดจุดที่ 1 เป็นด้านมุมบนซ้ายของรูป และ พิกัดจุดที่ 2 เป็นด้านมุมล่างขวาของรูป

นอกจากนี้คำสั่งนี้ยังสามารถใช้สำหรับ ลบข้อความ หรือรูป ที่อยู่ ณ ตำแหน่งใดๆบนหน้าจอได้ เพียงแต่ผู้ใช้สร้างรูปสี่เหลี่ยมนี้ไปครอบทับในพื้นที่ ที่ต้องการจะลบ โดยต้องกำหนดช่อง Fill เป็น '1' ถ้าพื้นหลังของหน้าจอเป็นสีดำ หรือ กำหนดเป็น '2' ถ้าพื้นหลังหน้าจอเป็นสีขาว ค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

1. ค่าพิกัดจุดที่ 1 ในช่อง X1 (3 Byte: '000-127') และช่อง Y1 (2 Byte: '00-63')
2. ค่าพิกัดจุดที่ 2 ในช่อง X2 (3 Byte: '000-127') และช่อง Y2 (2 Byte: '00-63')
3. ค่า Fill ในช่อง Fill เมื่อ '0' = ไม่ Fill (แสดงเป็นลายเส้นสีดำ) , '1' = Fill ดำ (เป็นสีดำทั้งรูป) , '2' = Fill ขาว (เป็นสีขาวทั้งรูปไม่มีลายเส้น) ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง'01') แล้วตามด้วยคำสั่งนี้ ลักษณะของรูปจะเปลี่ยนเป็นตรงข้ามก็ต่อเมื่อให้ช่อง Fill เป็น '0' เท่านั้น แต่ถ้าให้ช่อง Fill เป็น '1' หรือ '2' คำสั่ง Invert จะไม่มีผลทำให้รูปที่วาดเปลี่ยนแปลง

Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	Mk6	Fill	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	14	=	000-127	,	00-63	,	000-127	,	00-63	,	0-2	Enter (0x0D)
Respond Command จากบอร์ด														
*	00-FF	#	14	=	OK									

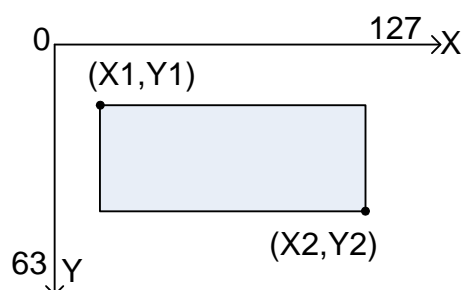
ช่อง Fill '0' = รูปเป็นลายเส้น , '1' = รูปเป็นสีดำทั้งรูป , '2' = รูปเป็นสีขาวทั้งรูป ไม่มีลายเส้น

Ex9. ให้ ID = '05' Plot รูปสี่เหลี่ยม ที่มีพิกัด

(X1= dot ที่ 10 ,Y1 = dot ที่ 10) และ

(X2 = dot ที่ 100 , Y2 = dot ที่ 30), ไม่ Fill รูป

```
main()
{
    char esc = 0x1B , enter = 0x0D ;
    printf( "%c05#14=010,10,100,30,0%c" ,esc,enter) ; //Plot สี่เหลี่ยม
}
```



COMMAND '15' (Plot Line)

เป็นคำสั่งสำหรับวาดเส้นตรงใน แนวตั้ง แนวนอน หรือ แนวเฉียง ก็ได้ซึ่งจะขึ้นอยู่กับตำแหน่งพิกัดจุด ที่กำหนดในคำสั่ง โดยคำสั่งนี้จะต้องกำหนดพิกัดจุด (x,y) จำนวน 2 จุด คือจุดหัว และ จุดท้าย ของเส้นที่จะวาด โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

1. ค่าพิกัดจุดที่ 1(หัว) ใน ช่อง X1 (3 Byte: '000-127') และ ช่อง Y1 (2 Byte: '00-63')
2. ค่าพิกัดจุดที่ 2 (ท้าย) ใน ช่อง X2 (3 Byte: '000-127') และ ช่อง Y2 (2 Byte: '00-63')

ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง'01') แล้วตามด้วยคำสั่งนี้ ลักษณะของเส้นจะเปลี่ยนเป็นสีขาว

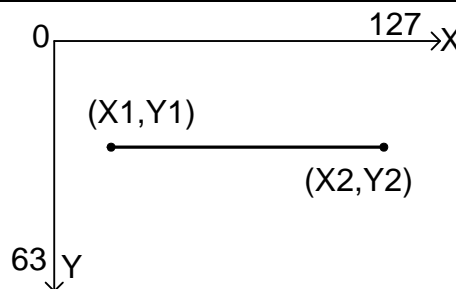
Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte
Esc (0x1B)	00-FF	#	15	=	000-127	,	00-63	,	000-127	,	00-63	Enter (0x0D)
Respond Command จากบอร์ด												
*	00-FF	#	15	=	OK							

Ex10. ให้ ID = '06' Plot เส้นตรงแนวนอน ที่มีพิกัด

(X1= dot ที่ 5 ,Y1 = dot ที่ 30) และ

(X2 = dot ที่ 120 , Y2 = dot ที่ 30)

```
main()
{
  char  esc = 0x1B , enter = 0x0D ;
  printf( "%c06#15=005,30,120,30%c" ,esc,enter) ; // Plot เส้นตรง
}
```

**COMMAND '16' (Plot Line Thick)**

เป็นคำสั่งสำหรับวาดเส้นตรงใน แนวตั้ง แนวนอน หรือ แนวเฉียง ก็ได้ซึ่งจะขึ้นอยู่กับตำแหน่งพิกัดจุด ที่กำหนดในคำสั่ง และสามารถกำหนดความหนาของเส้นได้ 5 ระดับ โดยคำสั่งนี้จะต้องกำหนดพิกัดจุด (x,y) จำนวน 2 จุด คือจุดหัว และ จุดท้าย ของเส้นที่จะวาด โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

1. ค่าพิกัดจุดที่ 1(หัว) ในช่อง X1 (3 Byte: '000-127') และช่อง Y1 (2 Byte: '00-63')
2. ค่าพิกัดจุดที่ 2 (ท้าย) ในช่อง X2 (3 Byte: '000-127') และช่อง Y2 (2 Byte: '00-63')
3. ค่าความหนาของเส้นในช่อง Thick ขนาด 1 Byte ซึ่งกำหนดค่าความหนาได้ตั้งแต่ระดับ '0' - '4'

ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง'01') แล้วตามด้วยคำสั่งนี้ ลักษณะของเส้นจะเปลี่ยนเป็นสีขาว

Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	Mk6	Thick	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	16	=	000-127	,	00-63	,	000-127	,	00-63	,	0-4	Enter (0x0D)
Respond Command จากบอร์ด														
*	00-FF	#	16	=	OK									

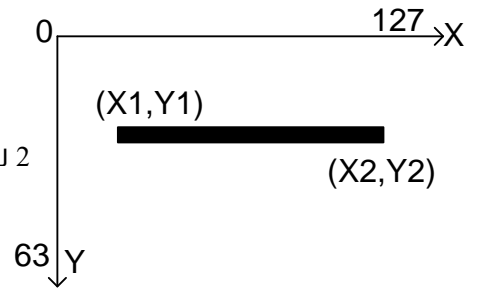
กรณีที่วาดเส้นตรงในแนวตั้งฉาก ไม่ว่าจะกำหนดความหนาที่ระดับใด จะ ได้ความกว้างของเส้นแค่ 1 dot เสมอ

Ex11. ให้ ID = '06' Plot เส้นตรงแนวนอน ที่มีพิกัด

(X1 = dot ที่ 5 , Y1 = dot ที่ 30) และ

(X2 = dot ที่ 120 , Y2 = dot ที่ 30), กำหนดความหนาเส้นที่ระดับ 2

```
main()
{
  char  esc = 0x1B , enter = 0x0D ;
  printf( "%c06#16=005,30,120,30,%2c" ,esc,enter) ; // Plot เส้นตรงหนา
}
```



COMMAND '17' (Plot Polygon)

เป็นคำสั่งสำหรับวาดรูปหลายเหลี่ยมโดยสามารถวาดรูปได้ตั้งแต่ 3 เหลี่ยม จนถึง 9 เหลี่ยม และสามารถกำหนดความหนาของเส้นที่จะวาดได้ 5 ระดับตั้งแต่ '0'-'4' ที่ช่อง Thick การสร้างรูปด้วยคำสั่งนี้จะไม่สามารถ Fill รูปได้ ในการกำหนดตำแหน่งพิกัด (x,y) ของรูปที่จะวาด ให้กำหนดจำนวนจุดเท่ากับจำนวนด้านของรูปที่จะวาด เช่น ต้องการวาดรูป 6 เหลี่ยมก็ให้กำหนดพิกัดจุด จำนวนหกจุด เป็นต้น การทำงานของโปรแกรมก็จะเริ่ม Plot โดยลากเส้นจากจุดที่1 ไปหาจุดที่2 จากจุดที่2 ไปหาจุดที่3 เรียงกันไปในลักษณะนี้จนครบจำนวนจุดหรือจำนวนด้านตามที่ผู้ใช้กำหนด โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

1. ค่าจำนวนจุดหรือด้านของรูปที่จะวาด ในช่อง NumPoint (1 Byte : '3-9') จะต้องสอดคล้องกับจำนวนพิกัดจุดที่จะใส่ในช่อง Position(x,y)ด้วย เช่น ถ้ารูป 3เหลี่ยม ก็ให้ใส่ '3' และในช่อง Position(x,y) ก็จะต้องเป็น (x1,y1,x2y2,x3y3)เป็นต้น
2. ค่าตำแหน่งพิกัดจุดในช่อง Position (x,y) ให้ใส่พิกัดจุดตามจำนวนด้านของรูปที่จะวาดซึ่งรับได้สูงสุดไม่เกิน 9 ด้าน
3. ค่าความหนาของเส้นในช่อง Thick (1 Byte) ซึ่งกำหนดค่าความหนาได้ตั้งแต่ระดับ '0' - '4'

ในกรณีที่ Set คำสั่ง Invert ไว้ (คำสั่ง'01') แล้วตามด้วยคำสั่งนี้ ลักษณะลายเส้นของรูปที่วาดจะเปลี่ยนเป็นสีขาว

Start	ID	Mk1	Command	Mk2	NumPoint	Mk3	Position (x,y)	Mk6	Thick	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	X1,Y1, ... ,X9,Y9	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	17	=	3-9	,	000-127,00-63, ... ,000-127,00-63	,	0-4	Enter (0x0D)
Respond Command จากบอร์ด										
*	00-FF	#	17	=	OK					

ช่อง NumPoint = จำนวนจุดหรือด้านของรูปที่จะวาดกำหนดได้ตั้งแต่ '3'-'9'

ช่อง Position(x,y) = ค่าตำแหน่งพิกัดจุด x: 3 byte('000-127') , y : 2 byte('00-63')

ช่อง Thick = ความหนาของเส้นที่จะวาด กำหนดได้ตั้งแต่ '0'-'4'

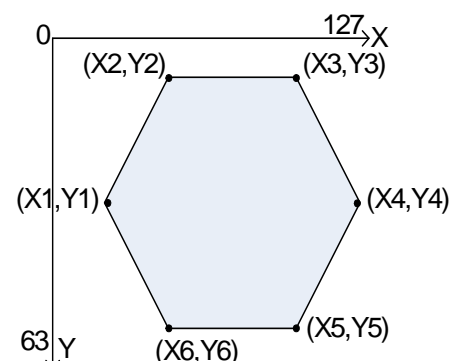
Ex12. ให้ ID = '07' Plot รูป 6 เหลี่ยม ที่มีพิกัด

(X1= dot ที่ 20 , Y1 = dot ที่ 30) ,(X2 = dot ที่ 40 , Y2 = dot ที่ 10),

(X3= dot ที่ 87 , Y3 = dot ที่ 10) ,(X4 = dot ที่ 107 , Y4 = dot ที่ 30),

(X5= dot ที่ 87 , Y5 = dot ที่ 50) ,(X6 = dot ที่ 40 , Y6 = dot ที่ 50),

กำหนดความหนาของเส้นที่ระดับ 3



```

main()
{
    char  esc = 0x1B , enter = 0x0D      ;

    printf( "%c07#17=6,020,30,040,10,087,10,107,30,087,50,040,50,3%c" ,esc,enter)  ;
}


```

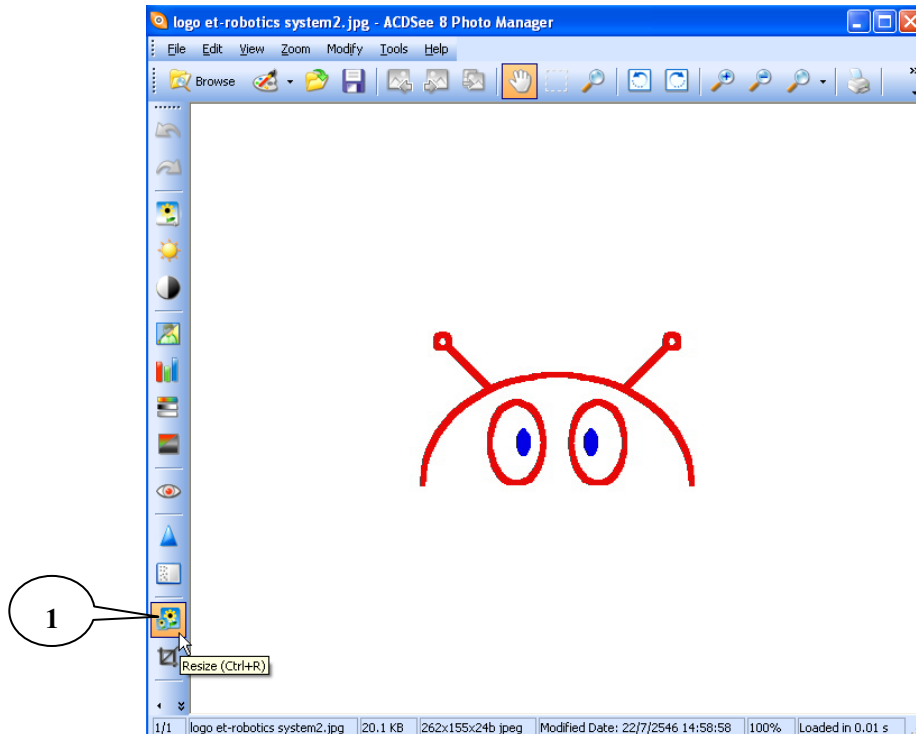
หมายเหตุ ในการกำหนดพิกัดของรูปหรือ เส้น จะต้องดูด้วยว่า พิกัดที่กำหนดนั้นอยู่ในขอบเขตของหน้าจอตลอดช่วงของการ Plot รูป มิเช่นนั้นจะทำให้รูปที่วาด ผิดพลาด หรือ โปรแกรมค้างได้ อาทิเช่น วาดรูปวงกลม รัศมี 10 Dot พิกัด Center (x,y)ของวงกลมก็จะต้องกำหนดให้อยู่ในช่วงที่ลากร์ศมีออกไปแล้วยังคงอยู่ในขอบเขตของหน้าจอในทุกๆด้าน เป็นต้น

4.3 กลุ่มคำสั่งที่เกี่ยวข้องกับ Picture Mode

คำสั่งในกลุ่มนี้จะเป็นคำสั่งที่เกี่ยวข้องกับการนำรูปภาพจากภายนอกส่งมาแสดงยังบอร์ด GLCD รวมทั้งการแสดงรูป Back Ground ที่มีมาให้พร้อมกับตัวบอร์ด สำหรับในการส่งรูปภาพมาแสดงบนหน้าจอ GLCD นั้น ผู้ใช้จะต้องทำการ Convert รูปให้ได้ค่า Code ออกมาก่อนแล้วเก็บ Code ไว้ที่หน่วยความจำของบอร์ดที่ผู้ใช้ ใช้ส่ง Command แล้วจึงทำการส่ง Command และ Code ของรูปภาพ มายังบอร์ด GLCD รูปก็จะถูกนำมาแสดงบนหน้าจอ GLCD โดยปกติแล้ว code ที่ Convert ได้จากรูปจะมีขนาด 1024 Byte ซึ่งจะเต็มหน้าจอ GLCD พอดี

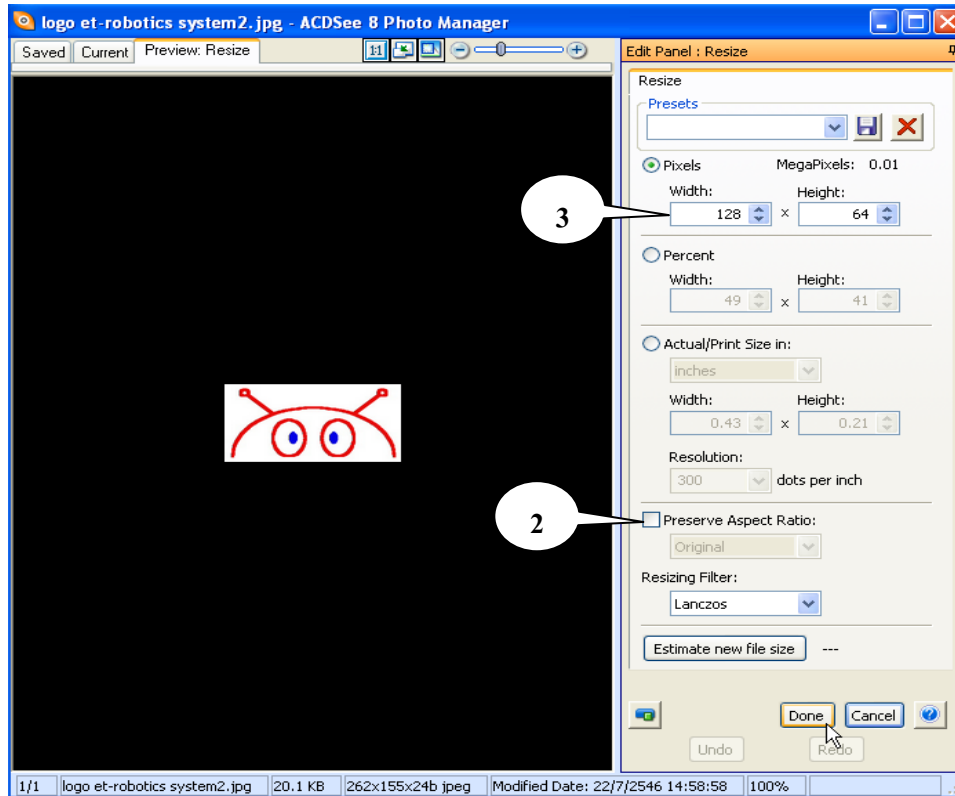
วิธีการแปลงรูปภาพให้เป็น Hex Code

1.) ทำการลดขนาดของรูปให้มีขนาด(กว้างxสูง) 128x64 Pixel โดย เปิดรูปขึ้นมาด้วยโปรแกรม ACDSee แล้วเลือกที่ไอคอน Resize ( หมายเลข1) ดังแสดงในรูปที่ 4.3.1



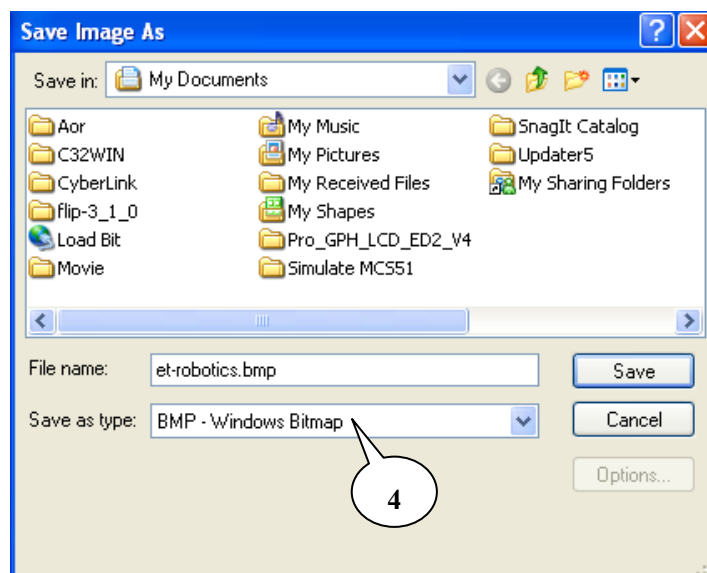
รูปที่ 4.3.1

2.) จะได้นหน้าต่างออกมาดังรูปที่ 4.3.2 ให้เอาเครื่องหมายถูกในช่อง Preserve Aspect Ratio: (หมายเลข2)ออก จากนั้นให้เลือกที่ Pixels และกำหนดขนาดในช่อง width:128 x height: 64 (หมายเลข3) เสร็จแล้วคลิก Done




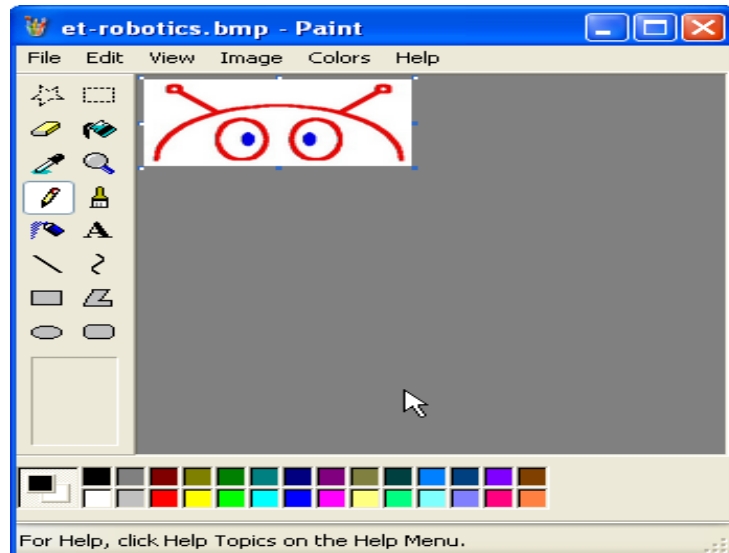
รูปที่ 4.3.2

3.) หลังจาก คลิก Done แล้วก็จะกลับมายังหน้าต่างเหมือนในรูปที่ 4.3.1 จากนั้นให้ไปที่เมนู File แล้วเลือก Save as.. ให้ทำการตั้งชื่อ File แล้ว Save เป็นนามสกุลจุด BMP เท่านั้น ดังรูปที่ 4.3.3 หลังจากตั้งชื่อเรียบร้อยแล้วก็กด Save จากนั้นก็ให้ปิดโปรแกรม ACDSee ได้



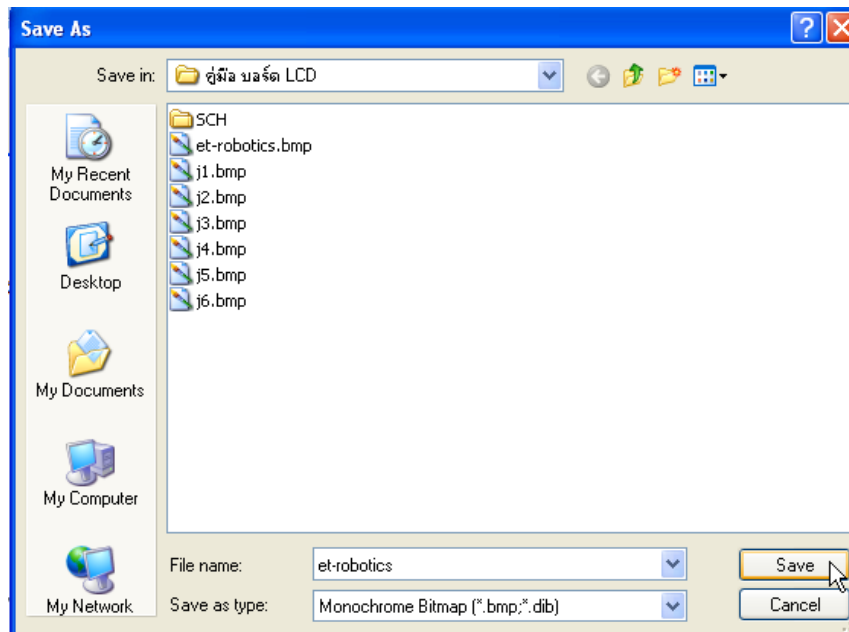
รูปที่ 4.3.3

4.) ทำการเปิดโปรแกรม Paint () ของ Window ขึ้นมาแล้วไปที่ เมนู File เลือก Open... เพื่อทำการเปิด File รูปที่ได้ Save ไว้ในข้อ 3 ขึ้นมา จะได้หน้าต่างแสดงดังรูปที่ 4.3.4



รูปที่ 4.3.4

5.) ไปที่เมนู File แล้วเลือก Save as.. ในช่อง File name ให้ทำการตั้งชื่อ File ในช่อง Save as type: ให้เลือก Monochrome Bitmap (*.bmp;*.dib) ดังแสดงในรูปที่ 4.3.5 จากนั้นกด Save ก็จะปรากฏหน้าต่างเหมือนกับรูปที่ 4.3.4 อีกครั้ง แต่รูปที่แสดงจะเปลี่ยนเป็นขาวดำแล้ว จากนั้นก็ให้ปิดโปรแกรม Paint ได้



รูปที่ 4.3.5

6.) ตอนนี้เราจะได้ File รูปที่สามารถใช้ Convert Hex Code ได้แล้ว ต่อไปก็ให้ผู้ใช้เปิดโปรแกรม glcd_editor.exe ที่ให้มากับแผ่น CD ขึ้นมา (Copy มาไว้ในเครื่องก่อนเปิด) จะได้หน้าต่างดังรูปที่ 4.3.6 ให้เลือกที่แท็บ KS0108 จากนั้นคลิกที่ปุ่ม Load BMP Picture เพื่อโหลด File รูปที่ผู้ใช้ได้ Save ไว้ด้วยโปรแกรม Paint ในหัวข้อที่ 5 เข้ามายังโปรแกรมนี้

8) จากรูปที่ 4.3.7 หลังจากได้ Code แล้ว ก็ให้กดที่ปุ่ม Copy CODE to Clipboard โดยสามารถเลือกรูปแบบ Code ได้ว่าต้องการ Copy code ในรูปแบบใด ซึ่งขึ้นอยู่กับบอร์ดที่ผู้ใช้ ใช้เขียนโปรแกรมส่ง Command ว่าเขียนด้วยภาษาอะไร หลังจาก Copy แล้ว ก็นำไปวางไว้ที่หน้าต่างของโปรแกรมที่ผู้ใช้ ใช้เขียนโปรแกรม ซึ่ง Code ที่ Copy ไปวางนี้อาจจะต้องแก้ไขบ้างในส่วนของการประกาศตัวแปร เพื่อปรับให้สอดคล้องกับรูปแบบของโปรแกรมที่ใช้เขียน จากนั้นก็ให้ส่ง Code นี้ไปยังบอร์ด GLC ตามรูปแบบของคำสั่งในการส่งรูปภาพ

COMMAND '20' (Write Picture)

เป็นคำสั่งสำหรับนำ File ภาพแสดงออกหน้าจอ LCD โดย File ภาพนี้จะต้องทำการแปลงให้อยู่ในรูปของ Code เสียก่อน ตามขั้นตอนที่กล่าวไว้ข้างต้น โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

1.) Data ที่เป็น Code ของไฟล์ภาพ ที่มีขนาด 1024 Byte โดยกำหนดในช่อง Data File BMP

Start	ID	Mark1	Command	Mark2	Data File BMP	END
1 byte	2 byte	1 byte	2 byte	1 byte	1024 byte	1 byte
Esc (0x1B)	00-FF	#	20	=	Code	Enter (0x0D)
Respond Command จากบอร์ด						
*	00-FF	#	20	=	OK	

ช่อง Code = ข้อมูลของไฟล์ภาพ ที่ได้จากการ Convert ขนาด 1024 byte

Ex13. ให้ ID = '08' แสดงรูปภาพออกหน้าจอ GLCD

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

// -----
// GLCD Picture name: et-robotics.bmp ตัวอย่าง Code File
// GLCD Model: KS0108 128x64
// -----

rom char robotics_bmp[1024]= { //เก็บข้อมูลภาพ 1024 byte ในพื้นที่ Flash
    0,0,0,0,0,0,0,0,0,0,0,0,240,240,48,48,48,
    240,240,192,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        : : :
        . . .
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,15,15,0,0,0,0

};

//----- Transmit data -----
void Tx(unsigned char data) //Function Sent data for Z8Encore
{
    while(!(U0STAT0 & 0x04)){;}
    U0D = data;
}

main()
{
    int k;
    char esc = 0x1B , enter = 0x0D ;
```



```

printf( "%c08#20=", esc)           ; //ส่ง command '20'

for(k=0;k<1024;k++)
    Tx(robotics_bmp[k])           ; //ส่ง data 1024 byte

printf( "%c", enter)             ; //End byte of Command
}

```

COMMAND '21' (Demo Test GLCD)

เป็นคำสั่งสำหรับทดสอบการทำงานของจอ GLCD โดยมีรูปแบบคำสั่งดังนี้

Start	ID	Mark1	Command	Mark2	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	21	=	Enter (0x0D)
Respond Command จากบอร์ด					
*	00-FF	#	21	=	OK

Ex14. ให้ ID = '09' Demo Test

```

main()
{
    char esc = 0x1B , enter = 0x0D ;
    printf( "%c09#21=%c", esc, enter) ; // Demo Test
}

```

COMMAND '22' (Display Wall paper)

เป็นคำสั่งสำหรับแสดงภาพ Wallpaper ซึ่งคำสั่งนี้ผู้ใช้สามารถเรียกรูปที่มีมาให้ในตัวบอร์ด GLCD ออกมาแสดงที่หน้าจอ GLCD ได้ อาจจะใช้เป็นรูปพิกหน้าจอก็ได้ซึ่งจะมีรูปภาพให้เลือกทั้งหมด 7 รูป โดยจะต้องกำหนดค่าพารามิเตอร์ในช่อง Picture เพื่อเลือกรูปที่ต้องการให้แสดง โดยกำหนดค่าได้ตั้งแต่ '1-7'

Start	ID	Mark1	Command	Mark2	Picture	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	22	=	'1-7'	Enter (0x0D)
Respond Command จากบอร์ด						
*	00-FF	#	22	=		OK

ช่อง Picture = เลือกรูปที่จะให้แสดงออกหน้าจอ GLCD (เลือกค่า '1-7')

Ex15. ให้ ID = '0A' แสดงรูป Wall Paper หมายเลข4

```

main()
{
    char esc = 0x1B , enter = 0x0D ;
    printf( "%c0A#22=4%c", esc, enter) ; // Display Wallpaper
}

```

4.4 กลุ่มคำสั่งที่เกี่ยวข้องกับ Text Mode

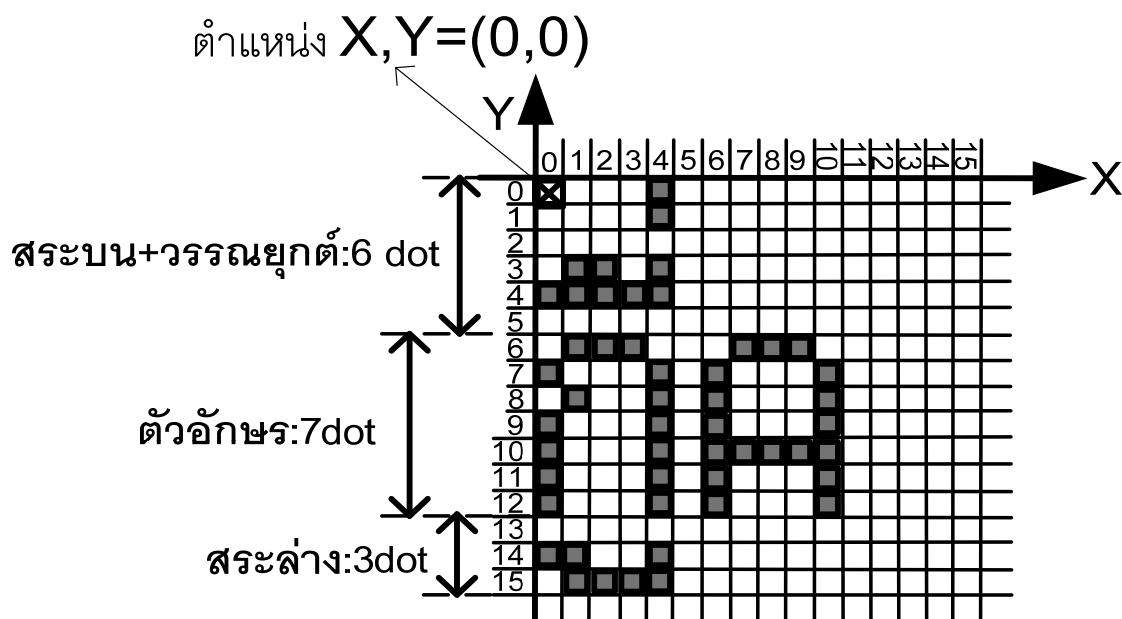
คำสั่งในกลุ่มนี้จะเป็นคำสั่งที่เกี่ยวข้องกับการ Plot ข้อความออกแสดงผลที่หน้าจอ GLCD โดยรูปแบบคำสั่งมีดังนี้

COMMAND '30' (Set Cursor Position)

คำสั่งนี้จะใช้สำหรับกำหนดตำแหน่งของ Cursor หรือตำแหน่งเริ่มต้นในเขียนข้อความหรือตัวอักษร ซึ่งโดยปกติถ้าเปิดเครื่องขึ้นมาค่าตำแหน่งเริ่มต้นนี้จะถูกกำหนดไว้ที่ตำแหน่ง (0,0) และหลังจากเขียนข้อความหรือตัวอักษรออกไปแล้วค่าตำแหน่ง Cursor นี้ ก็จะถูกเลื่อนให้โดยอัตโนมัติ

ความเข้าใจเกี่ยวกับการ Set ตำแหน่งที่จะเขียนข้อความหรือตัวอักษรบนหน้าจอ GLCD

ก่อนอื่นผู้ใช้งานจะต้องเข้าใจก่อนว่า ขนาดของตัวอักษรที่ใช้ในบอร์ดนี้ จะมีความสูง 7 dot แต่ความกว้างของตัวอักษรแต่ละตัวนั้นจะไม่เท่ากัน ตัวอักษร 1 ตัว ไม่ว่าจะเป็นภาษาไทยหรือภาษาอังกฤษ จะใช้พื้นที่ความสูงที่ 16 dot โดยใน 16 dot นี้ก็จะแบ่งเป็นพื้นที่ของ สระบน-สระล่าง และตัวอักษร รูปที่ 4.4.1 จะเห็นว่าถ้าผู้ใช้ Set ตำแหน่งเริ่มต้นในการเขียนข้อความที่ตำแหน่ง $x,y = (0,0)$ เราก็จะนับตำแหน่งนี้ลงไป 16 dot ซึ่งจะเป็นเนื้อที่ความสูงของ สระบน+ตัวอักษร+สระล่าง พื้นที่ความสูงของตัวอักษรจริงก็จะอยู่ที่ตำแหน่ง dot ที่ 6 ถึง dot ที่ 12 ในกรณีที่เป็นข้อความหรือตัวอักษรภาษาอังกฤษที่ไม่มีสระบน-ล่าง พื้นที่ของสระนี้ก็จะถูกเว้นว่างไว้ ซึ่งจะทำให้ผู้ใช้สามารถกำหนดตัวอักษรในบรรทัดที่ 2 ให้เข้ามาใกล้กับตัวอักษรที่อยู่ในบรรทัดแรกได้ ซึ่งก็จะทำให้ในหนึ่งหน้าจออาจจะเขียนข้อความได้มากกว่า 4 บรรทัด โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้คือ ค่าพิกัดจุดเริ่มต้นของข้อความใน ช่อง X (3Byte: '000-127') และ ช่อง Y (2 Byte: '00-63')



รูปที่ 4.4.1 แสดงการแบ่งพื้นที่ตัวอักษรบนหน้าจอ GLCD

Start	ID	Mark1	Command	Mark2	X	Mark3	Y	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2 byte	1 byte
Esc (0x1B)	00-FF	#	30	=	000-127	,	00-63	Enter (0x0D)
Respond Command จากบอร์ด								
*	00-FF	#	30	=	OK			

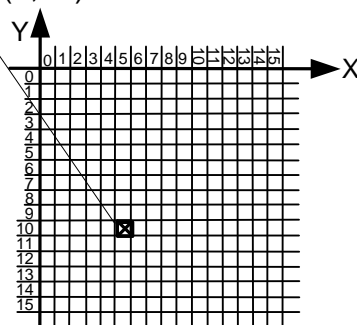
Ex16. ให้ ID = '0B' กำหนดตำแหน่งเริ่มต้นเขียนข้อความ

ที่พิกัดตำแหน่ง X= dot ที่ 5 ,Y = dot ที่ 10

```
main()
{
    char  esc = 0x1B , enter = 0x0D      ;

    printf( "%c0B#30=005,10%c" ,esc,enter) ; //Set x,y
}
```

Position X,Y=(5,10)



COMMAND '31' (Write Message1)

สำหรับคำสั่งนี้จะใช้สำหรับเขียนข้อความหรือตัวอักษรให้แสดงผลบนหน้าจอ GLCD ซึ่งลักษณะการทำงานของคำสั่งนี้จะ Up date เฉพาะข้อความหรือตัวอักษรที่ส่งมา ออกหน้าจอเท่านั้น โดยคำสั่งนี้จะสามารถเลือกโหมดสีของตัวอักษรได้ 2 แบบว่าจะให้แสดงตัวอักษรเป็นสีขาว หรือสีดำก็ได้ ซึ่งจะไม่สนใจคำสั่ง Invert (คำสั่ง'01') ที่จะ Set ไว้หรือไม่ก็ตาม ดังนั้นไม่ว่าพื้นหลังของหน้าจอ GLCD จะเป็นสีขาวหรือดำ ผู้ใช้ก็สามารถกำหนดตัวอักษรได้ทั้งสีขาวและดำ เพียงแต่ถ้ากำหนดสีตัวอักษรสีเดียวกับพื้นหลังก็จะทำให้มองไม่เห็นตัวอักษรที่เขียนเท่านั้นเอง ในคำสั่งนี้เมื่อเขียนข้อความหมดบรรทัดแล้วข้อความส่วนที่เกินก็จะถูกนำมาเขียนในบรรทัดใหม่ให้อัดโน้มนัด แต่ถ้าข้อความเต็มหน้าจอแล้ว ข้อความส่วนที่เกินในบรรทัดสุดท้ายของหน้าจอจะถูกตัดทิ้งไป โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

- 1.) กำหนดสีของตัวอักษรในช่อง Mode (1 byte) โดย '0' = แสดงตัวอักษรสีขาว และ '1' = แสดงตัวอักษรสีดำ
- 2.) กำหนดข้อความที่จะเขียนออกหน้าจอ ในช่อง Data Text ,จำนวนตัวอักษรไม่เกิน 200 byte (สระ+วรรณยุกต์+ตัวอักษร)

Start	ID	Mark1	Command	Mark2	Mode	Mark3	Data Text	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	1-200 byte	1 byte
Esc (0x1B)	00-FF	#	31	=	0-1	:	ข้อความ	Enter (0x0D)
Respond Command จากบอร์ด								
*	00-FF	#	31	=			OK	

ช่อง Mode = '0' : แสดงอักษรสีขาว , '1':แสดงอักษรสีดำ

ช่อง Data Text = Data ASCII Thai-Eng :1-200 byte

Ex17. ให้ ID = '0C' เริ่มข้อความที่ตำแหน่ง X,Y=(10,20) เขียนข้อความ “สวัสดีครับ” ออกหน้าจอ ใช้ตัวอักษรสีดำ

```
main()
{
    char  esc = 0x1B , enter = 0x0D      ;

    printf( "%c0C#30=010,20%c" ,esc,enter) ; //Set x,y(10,20)

    printf( "%c0C#31=1:สวัสดีครับ" ,esc,enter) ; //Write Message สวัสดีครับ
}
```

ในตัวอย่างนี้ไม่ได้ตรวจสอบ Respond Command ซึ่งในความเป็นจริงควรจะตรวจสอบด้วยเพื่อความไม่ผิดพลาดในการทำคำสั่งที่ 2 ถ้าโปรแกรมที่ใช้เขียนไม่สนับสนุนภาษาไทย เวลาพิมพ์ข้อความภาษาไทยลงไปก็อาจจะเห็นเป็นตัวอักษรที่อ่านไม่รู้เรื่องแต่ก็ไม่มีอะไรสามารถทำได้

COMMAND '32' (Write Message2)

สำหรับคำสั่งนี้จะใช้สำหรับเขียนข้อความหรือตัวอักษรให้แสดงผลบนหน้าจอ GLCD เหมือนกับคำสั่ง '31' แต่ลักษณะการทำงานของคำสั่งนี้จะเป็นการ Update ข้อมูลทั้งหน้าจอต่อการส่งคำสั่งในแต่ละครั้ง โดยคำสั่งนี้จะสามารถเลือกรูปแบบการจัดเรียงข้อความได้ 4 แบบ ซึ่งสามารถกำหนดรูปแบบได้ในช่อง Mode สำหรับสีของตัวอักษรในคำสั่งนี้จะขึ้นอยู่กับคำสั่ง Invert (คำสั่ง '01') คือถ้า Invert = '1' ตัวอักษรก็จะถูกเขียนด้วยสีขาว(พื้นหลังจะสีดำ) ถ้า Invert = 0 ตัวอักษรก็จะถูกเขียนด้วยสีดำ(พื้นหลังสีขาว) โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่

1.) เลือกรูปแบบการจัดเรียงข้อความ ในช่อง Mode (1 Byte) โดยเลือกรูปแบบได้ตั้งแต่ค่า '0' - '3' เมื่อ

'0' = แสดงข้อความได้จนเต็มหน้าจอต่อการส่งคำสั่ง 1 ครั้ง เมื่อจบบรรทัดแล้วข้อความส่วนที่เกินจะถูกนำมาเขียน

ในบรรทัดใหม่ให้โดยอัตโนมัติ ถ้าเป็นบรรทัดสุดท้ายของหน้าจอ ข้อความที่เกินจะถูกตัดทิ้งไป

'1' = แสดงข้อความครั้งละ 1 บรรทัดต่อการส่งคำสั่ง 1 ครั้ง และจะจัดข้อความของบรรทัดนั้นให้ชิดซ้าย ข้อความส่วนที่เกินบรรทัดจะถูกตัดทิ้ง

'2' = แสดงข้อความครั้งละ 1 บรรทัดต่อการส่งคำสั่ง 1 ครั้ง และจะจัดข้อความของบรรทัดนั้นให้อยู่กึ่งกลางของหน้าจอ ข้อความส่วนที่เกินบรรทัดจะถูกตัดทิ้ง

'3' = แสดงข้อความครั้งละ 1 บรรทัดต่อการส่งคำสั่ง 1 ครั้ง และจะจัดข้อความของบรรทัดนั้นให้ชิดขวา ข้อความส่วนที่เกินบรรทัดจะถูกตัดทิ้ง

2.) กำหนดข้อความที่จะเขียนออกหน้าจอ ในช่อง Data Text ,จำนวนตัวอักษรไม่เกิน 200 byte (สระ+วรรณยุกต์+ตัวอักษร)

Start	ID	Mark1	Command	Mark2	Mode	Mark3	Data Text	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	1-200 byte	1 byte
Esc (0x1B)	00-FF	#	32	=	0-3	:	ข้อความ	Enter (0x0D)
Respond Command จากบอร์ด								
*	00-FF	#	32	=				OK

Ex18. ให้ ID = '0D' เริ่มข้อความที่ตำแหน่ง X,Y=(0,0) เขียนข้อความ "บริษัทอิตีที" ออกหน้าจอ โดยให้ข้อความอยู่กึ่งกลางของหน้าจอ

```
main()
{
  char esc = 0x1B , enter = 0x0D ;

  printf("%c0D#30=000,00%c", esc, enter) ; //Set x,y(0,0)

  printf("%c0D#32=2:บริษัทอิตีที", esc, enter) ; //Write Message บริษัทอิตีที
}
```

ในตัวอย่างนี้ไม่ได้ตรวจสอบ Respond Command ซึ่งในความเป็นจริงควรจะตรวจสอบด้วยเพื่อความไม่ผิดพลาดในการทำคำสั่งที่ 2 ถ้าโปรแกรมที่ใช้เขียนไม่สนับสนุนภาษาไทย เวลาพิมพ์ข้อความภาษาไทยลงไปก็อาจจะเห็นเป็นตัวอักษรที่อ่านไม่รู้เรื่องแต่ก็ไม่มีอะไรสามารถทำได้

ขอบเขตเกี่ยวกับการเขียนข้อความด้วยคำสั่ง '31' และ '32'

- ก่อนที่จะใช้คำสั่งนี้ก็ควรกำหนดตำแหน่งเริ่มต้นในการเขียนด้วยคำสั่ง '30' เสียก่อน
- ในการส่งข้อความออกไปแต่ละครั้งใน 1 คำสั่ง จะสามารถส่งข้อความได้ไม่เกิน 200 byte (สระ+ตัวอักษร+วรรณยุกต์)
- เมื่อข้อความที่ส่งมาถูกแสดงเกิน 1 หน้าจอข้อความส่วนที่เกินจะถูกตัดทิ้งไป
- ตัวอักษรสามารถพิมพ์จาก Key Board ได้เลย หรือถ้าส่งเป็น Code ก็ให้ดูตาราง ASCII ในภาคผนวกได้
- หลักการส่งตัวอักษรภาษาไทยมาให้กับบอร์ด GLCD จะใช้หลักการเหมือนกับการพิมพ์ภาษาไทยทั่วไป เช่น คำว่า "น้ำ" = น - ุ - ํ หรือ "สี่" = ส - ุ - ี่ เป็นต้น
- ในการเว้นวรรคตัวอักษร ให้ใช้ Space Bar หรือแทนด้วยค่า 0x20 โดย 1 Space bar จะเท่ากับช่องว่าง 5 dot ในกรณีที่ต้องการช่องว่างน้อยกว่า 5 dot ผู้ใช้สามารถส่งค่า 0x01 ออกมาแทนตัวอักษรได้ โดยส่ง 1 ครั้งจะแทนช่องว่าง 1 dot
- การเขียนข้อมูลออกมาแต่ละครั้งที่ตำแหน่งเดียวกันถ้าไม่ลบข้อมูลเก่าที่แสดงอยู่ออกก่อนข้อมูลใหม่จะมาทับซ้อนกับข้อมูลเดิมซึ่งจะทำให้อ่านไม่รู้เรื่อง
- ในการส่งตัวอักษรออกไปแต่ละตัว โปรแกรมจะเว้นช่องว่างระหว่างตัวอักษรไว้ให้ 1 dot โดยอัตโนมัติ

COMMAND '33' (Del Text ขนาด 7x5 dot)

สำหรับคำสั่งนี้จะใช้สำหรับลบตัวอักษรที่มีขนาดความสูง 7 dot กว้าง 5 dot เท่านั้น เช่นตัวเลข 0-9 โดยจะลบตัวอักษรในตำแหน่งที่ Cursor ปรากฏอยู่ หรือจะลบตัวอักษรที่อยู่หน้า Cursor ก็ได้ (สามารถกำหนดตำแหน่ง Cursor ได้จากคำสั่ง '30') ถ้าต้องการลบตัวอักษรหรือข้อความที่มีขนาดต่างจากนี้ เช่น ตัวอักษรหรือสระ ภาษาไทย ให้ใช้คำสั่ง '14' แทน โดยกำหนด ช่อง Fill = '1' หรือ '2' ซึ่งจะทำให้สามารถลบตัวอักษรที่อยู่ในตำแหน่งใดๆบนหน้าจอก็ได้ ค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่ ค่าในช่อง Mode ถ้าเป็น

'0' = ลบตัวอักษรสีดำในตำแหน่งที่ Cursor ปรากฏ , '1' = ลบตัวอักษรสีขาวในตำแหน่งที่ Cursor

'2' = ลบตัวอักษรสีดำที่อยู่ในตำแหน่งหน้า Cursor , '3' = ลบตัวอักษรสีขาวที่อยู่ในตำแหน่งหน้า Cursor

Start	ID	Mark1	Command	Mark2	Mode	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	33	=	0-3	Enter (0x0D)
Respond Command จากบอร์ด						
*	00-FF	#	33	=		OK

ช่อง Mode = '0' : ลบตัวอักษรสีดำในตำแหน่งที่ Cursor ปรากฏ , '1' : ลบตัวอักษรสีขาวในตำแหน่งที่ Cursor ปรากฏ

'2' : ลบตัวอักษรสีดำที่อยู่ในตำแหน่งหน้า Cursor , '3' : ลบตัวอักษรสีขาวที่อยู่ในตำแหน่งหน้า Cursor

Ex19. ให้ ID = '0E' ลบตัวอักษรสีดำที่ตำแหน่ง X,Y=(0,0)

```
main()
{
    char  esc = 0x1B , enter = 0x0D      ;

    printf("%c0E#30=000,00%c",esc,enter) ; // Set ตำแหน่งที่จะลบตัวอักษร x,y(0,0)
    delay(200)                             ; //delay คำสั่ง
    printf("%c0E#33=0",esc,enter)         ; // ลบตัวอักษร
}
```

ในการกำหนดตำแหน่งที่จะลบตัวอักษรนั้นก็ให้กำหนดตำแหน่งด้วยคำสั่ง '30' ตามรูปแบบที่กล่าวไว้ได้เลย หลังจากทำคำสั่ง '33' แล้ว คำสั่งก็จะทำการลบตัวอักษรเฉพาะในพื้นที่ ที่เป็นส่วนของตัวอักษรเท่านั้น ส่วนในพื้นที่สระเบน-ล่าง จะไม่ถูกลบ

COMMAND '34' (ON/OFF Cursor)

สำหรับคำสั่งนี้จะใช้สำหรับ ON/OFF Cursor เมื่อสั่ง ON Cursor ผู้ใช้ก็จะเห็น Cursor กระทบในตำแหน่งต่อจากตัวอักษรตัวสุดท้ายที่เขียนไว้ล่าสุด เมื่อเขียนตัวอักษรเสร็จ ตัว Cursor ก็จะขยับไปด้านขวาให้อัตโนมัติ นอกจากนี้ผู้ใช้สามารถกำหนดตำแหน่ง Cursor เพื่อใช้แสดงตำแหน่งของตัวอักษรที่จะเขียนต่อไปได้โดยใช้คำสั่ง '30' เป็นตัวกำหนดตำแหน่งเช่นเดิม ซึ่งจะทำให้ผู้ใช้เห็นได้ง่ายขึ้น ว่าอักษรที่จะถูกเขียนต่อไปอยู่ที่ตำแหน่งใดบนหน้าจอ โดยค่าพารามิเตอร์ที่จะต้องกำหนดให้กับคำสั่งนี้ได้แก่ค่าในช่อง OFF/ON ถ้าเป็น '0' = Cursor OFF , '1' = Cursor ON

Start	ID	Mark1	Command	Mark2	OFF/ON	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	34	=	0-1	Enter (0x0D)
Respond Command จากบอร์ด						
*	00-FF	#	34	=		OK

ช่อง OFF/ON = '0' : Cursor OFF

'1' : Cursor ON

Ex20. ให้ ID = '0F' กำหนดให้ Currсор ON ที่ตำแหน่ง X,Y = (30,30)

```
main()
{
    char esc = 0x1B , enter = 0x0D ;

    printf("%c0F#30=030,30%c", esc, enter) ; // Set ตำแหน่งที่จะแสดง Cursor : x,y (30,30)

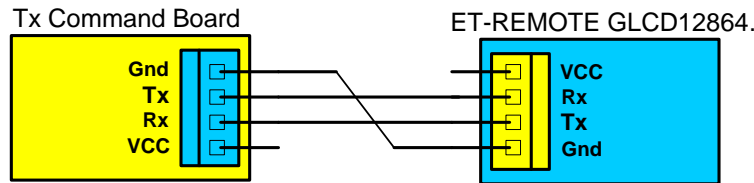
    printf("%c0F#34=1", esc, enter) ; // Set Cursor ON
}
```

ข้อควรจำ ในการนับตำแหน่งพิกัดบนหน้าจอ GLCD เพื่อใช้แทนลงในคำสั่งที่ต้องกำหนดค่าพารามิเตอร์เป็นพิกัดตำแหน่งนั้น ในแนวแกน X จะนับตั้งแต่ dot ที่ 0 จนถึง dot ที่ 127 ส่วนในแนวแกน Y จะนับตั้งแต่ dot ที่ 0 จนถึง dot ที่ 63 ซึ่งอ้างอิงตามตารางของหน้าจอ GLCD ในภาคผนวก และในการส่งคำสั่งให้กับตัวบอร์ด GLCD นั้น หลังจากส่งคำสั่งแล้วก็ควรจะเช็ค Respond ของคำสั่งด้วยเสมอ ก่อนที่จะส่งคำสั่งต่อไป อาจจะเช็คเฉพาะคำว่า "OK" 2 Byte สุดท้ายก็ได้ เพราะมันจะนั่นถ้าผู้ใช้ส่งคำสั่งออกไปขณะที่บอร์ด GLCD ยังไม่ส่ง "OK" ออกมาก็จะทำให้คำสั่งที่ส่งไปไม่มีผลใดๆ

5. การต่อใช้งาน RS232/422 และ 485 แบบ 4 เส้น

5.1 การส่ง Command ผ่าน RS232

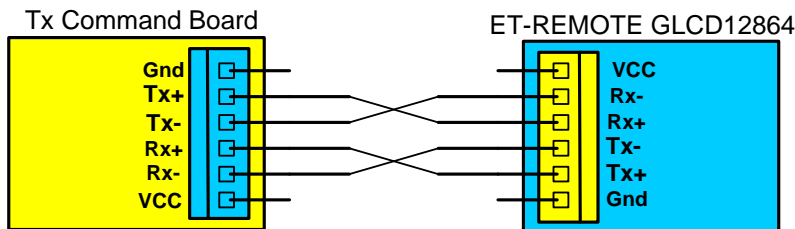
ในการส่ง Command ด้วยการสื่อสารแบบ RS232 นี้เหมาะสำหรับใช้ควบคุมบอร์ด GLCD เพียงบอร์ดเดียว ในการใช้งานจะต้องใส่ IC Line Driver Max232 ลงใน Socket ที่กำหนดไว้บนบอร์ด GLCD จากนั้นให้ Set Jumper หมายเลข 10 ทั้ง 2 ตัวมาทางด้าน 232 และทำการต่อสายสัญญาณสื่อสารดังรูป



รูปที่ 5.1 แสดงการต่อสายสื่อสารแบบ RS232 ระหว่างตัวส่งและตัวรับ

5.2 การส่ง Command ผ่าน RS422

ในการส่ง Command ด้วยการสื่อสารแบบ RS422 นี้เหมาะสำหรับใช้ควบคุมบอร์ด GLCD เพียงบอร์ดเดียว ซึ่งจะเหมือนกับแบบ RS232 แต่มีข้อดีตรงที่สามารถสื่อสารได้ไกลขึ้น ในการใช้งานจะต้องใส่ IC Line Driver 75176 2 ตัว ลงใน Socket ที่กำหนดไว้บนบอร์ด GLCD จากนั้นให้ Set Jumper หมายเลข 10 ทั้ง 2 ตัวมาทางด้าน 485 แล้วทำการต่อสายสัญญาณสื่อสารดังรูป

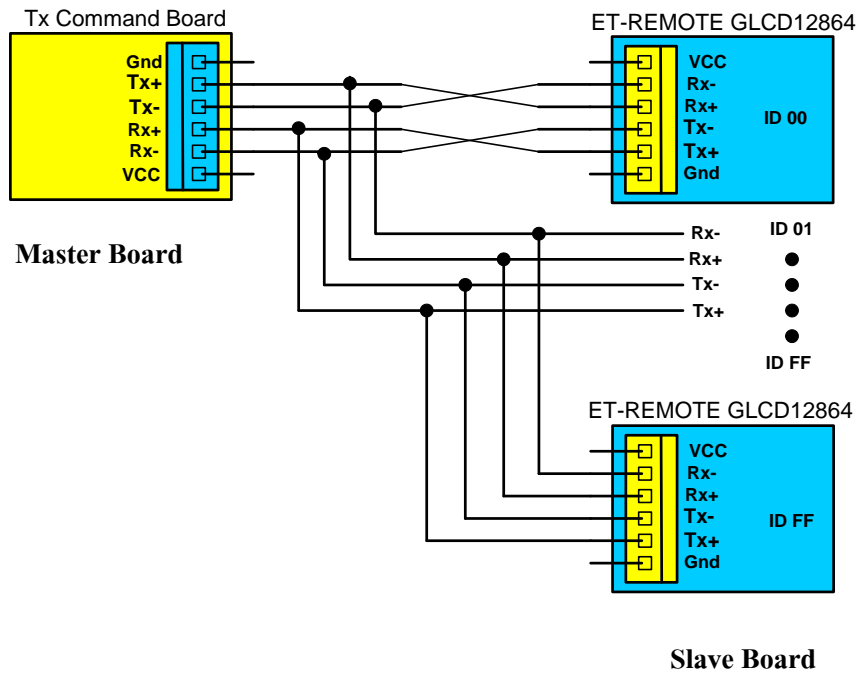


รูปที่ 5.2 แสดงการต่อสายสื่อสารแบบ RS422 ระหว่างตัวส่งและตัวรับ

5.3 การส่ง Command ผ่าน RS485 แบบ 4 Line

ในการส่ง Command ด้วยการสื่อสารแบบ RS485 แบบ 4 Line เราจะใช้การต่อสายสัญญาณแบบ RS422 ซึ่งดูได้จากรูปด้านล่าง การสื่อสารแบบนี้เหมาะสำหรับใช้ควบคุมบอร์ด GLCD หลายๆบอร์ด ซึ่งบอร์ด GLCD นี้ในแต่ละบอร์ดสามารถ Set ID ได้ตั้งแต่ค่า '00' – 'FF' ดูได้จากตารางการ Set Jumper ที่กล่าวไว้ในตอนต้นของคู่มือ

ในการใช้งานจะต้องใส่ IC Line Driver 75176 2 ตัว ลงใน Socket ที่กำหนดไว้บนบอร์ด GLCD จากนั้นให้ Set Jumper หมายเลข 10 มาทางด้าน 485 แล้วทำการต่อสายสัญญาณสื่อสารดังรูป



รูปที่5.3 แสดงการต่อสายสื่อสารแบบ RS485 4 Line ระหว่าง ตัวMaster และ ตัว Slave

สำหรับการสื่อสารด้วย RS485 นี้ ตัวบอร์ด GLCD จะรับ Command มากระทำก็ต่อเมื่อ Command นั้นมี ID ตรงกับ ID ของบอร์ดที่ได้ Set ไว้เท่านั้น เมื่อบอร์ดกระทำ Command เรียบร้อยแล้วก็จะส่ง Respond Command ตอบกลับออกมาให้ ส่วนบอร์ดอื่นๆที่ ID ไม่ตรงก็จะไม่ส่งอะไรออกมา ถ้า Respond Command นี้ถูกส่งออกมาจาก Slave Board พร้อมกันมากกว่า 2 บอร์ดขึ้นไป จะทำให้ตัว Master Board รับค่า Respond Command จากตัว Slave Board ผิดพลาดได้ซึ่งจะต้องระวังในส่วนนี้ด้วย

6. การ Set Up บอร์ด ET-REMOTE GLCD12864

- 1) Set Jumper หมายเลข 10 มาทางด้าน 232 หรือ 485 ขึ้นอยู่กับว่าผู้ใช้จะเลือกรูปแบบการสื่อสารแบบใด
- 2) Set Jumper หมายเลข 3 เพื่อกำหนดขั้ว Back Light ของ GLCD ที่นำมาใช้ให้ถูกต้อง
- 3) กำหนด Baud Rate ในการสื่อสารให้กับตัวบอร์ด GLCD โดย Set ที่ DIP-SW.B หมายเลข 1,2,3 ดูการเลือกได้จากตารางที่2.3
- 4) กำหนด ID ให้กับตัวบอร์ด GLCD โดย Set ที่ DIP-SW.A หมายเลข 1-8 ดูการเลือกได้จากตารางที่ 2.1
- 5) ต่อ DISPLAY GLCD เข้ากับตัวบอร์ด Control ให้ระวางหันด้านให้ถูกต้องด้วย
- 6) ปรับ VR บนบอร์ด เพื่อปรับความเข้มของจอ GLCD ให้พอดี
- 7) ต่อสายสื่อสารที่ขั้วต่อ RS232 หรือ RS485 จากตัวบอร์ด GLCD ไปยังขั้วต่อของบอร์ดที่ทำหน้าที่ส่ง Command
- 8) จ่ายไฟ AC/DC 7-12 V ให้กับตัวบอร์ด GLCD จะต้องได้ยินเสียงดนตรีดังขึ้น
- 9) ตรวจสอบ DIP-SW.B หมายเลข4 ถ้าอยู่ในตำแหน่ง ON แสดงว่าอยู่ใน Self-Test Mode ซึ่งตัวบอร์ดก็จะแสดงค่า Baud Rate และค่า ID ที่ผู้ใช้ Set ไว้ให้เห็นว่าถูกต้องหรือไม่ ถ้าต้องการเปลี่ยนแปลงค่าทั้ง 2 นี้ใหม่ก็ให้ Set DIP-SW.A และ B ใหม่ได้โดยแล้วรอนกว่าจะเห็นค่าที่ Set ใหม่เปลี่ยนแปลง ก็แสดงว่า Set ค่าใหม่เรียบร้อยแล้ว ถ้า DIP-SW.B หมายเลข4 อยู่ในตำแหน่ง OFF แสดงว่าอยู่ในโหมด RUN ที่หน้าจอ GLCD ก็จะว่างป่าว ตัวบอร์ด GLCD ก็พร้อมที่จะรับคำสั่งจากผู้ใช้

7. ตัวอย่างโปรแกรม

7.1 ตัวอย่างการทดสอบส่งคำสั่งผ่านทาง Hyper Terminal

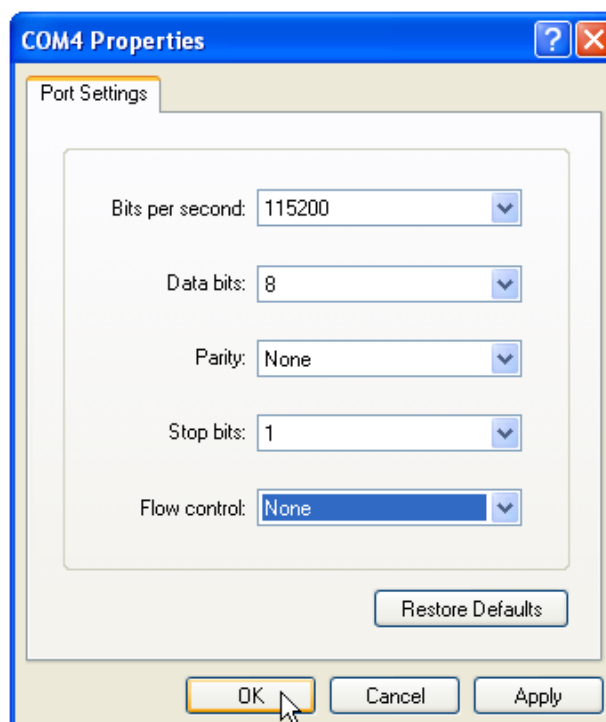
- 1) Set Jumper หมายเลข 10 มาทางด้าน 232
- 2) กำหนด Baud Rate ในการสื่อสาร 115200 bit/s โดย Set DIP-SW.B หมายเลข 1-3 ที่ตำแหน่ง ON
- 3) กำหนด ID บอร์ดเท่ากับ '00' โดย Set DIP-SW.A หมายเลข 1-8 ไปที่ตำแหน่ง OFF
- 4) ต่อสายสื่อสารที่ขั้วต่อ RS232 ของบอร์ด ไปยัง COM Port ของ PC ที่จะใช้งาน
- 5) เลื่อน DIP-SW.B หมายเลข4 มาที่ตำแหน่ง OFF เพื่อเข้าสู่โหมด RUN พร้อมรับคำสั่ง
- 6) เปิดโปรแกรม Hyper Terminal ขึ้นมา แล้วทำการ Set ค่าตามรูปด้านล่าง




(1) ตั้งชื่อ

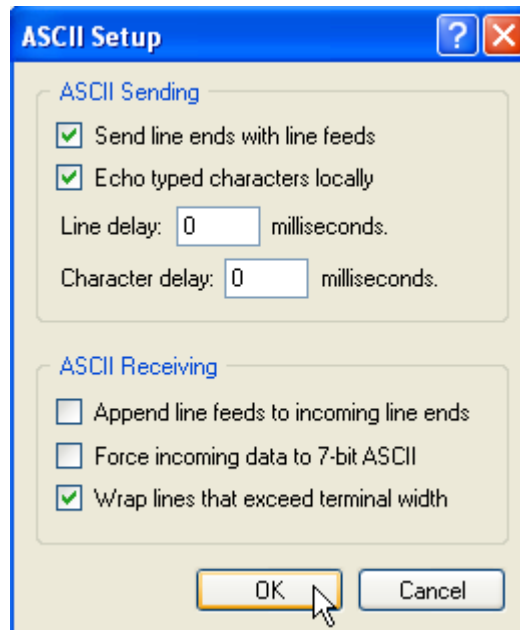


(2) เลือก COM PORT ที่ใช้สื่อสาร



(3) Set คุณสมบัติ Com Port ตามรูปด้านบน

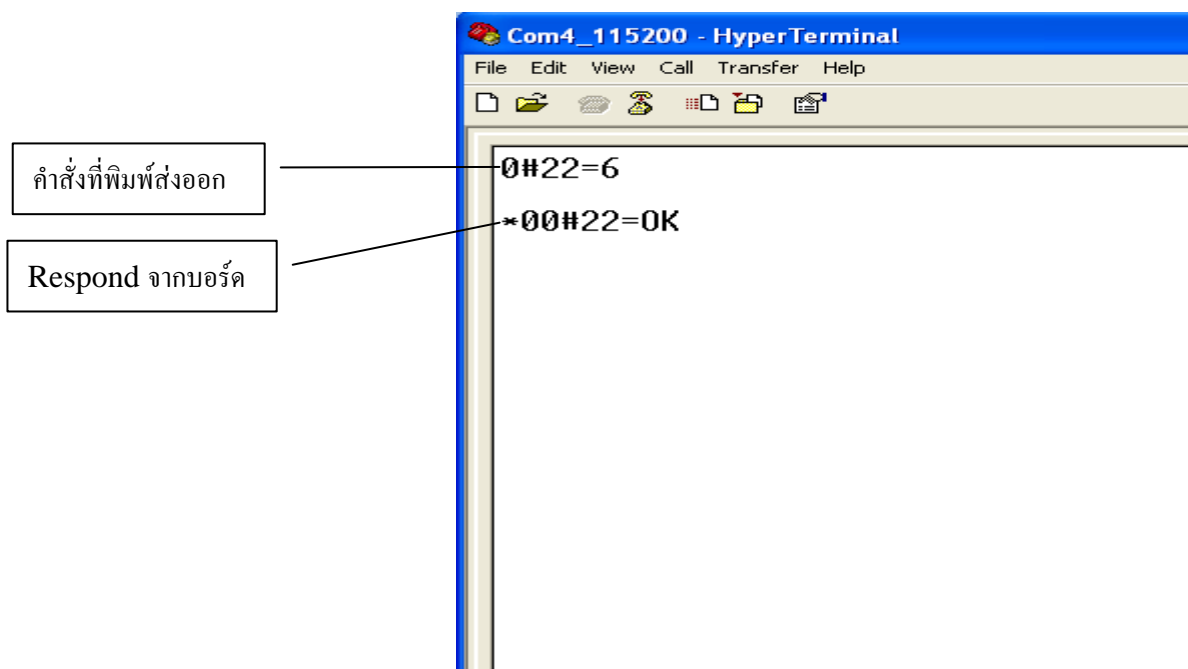
- 7) Click ที่ปุ่ม Properties () จะมีหน้าต่างขึ้นมา แล้วเลือกที่แท็บ Settings จากนั้นเลือกที่ปุ่ม ASCII Setup... จะได้หน้าต่างขึ้นมาดังรูปด้านล่าง แล้วทำการเลือกตามรูปด้านล่าง เสร็จแล้วกด OK



- 8) ลองพิมพ์คำสั่งส่งให้กับบอร์ด ในที่นี้จะทดสอบด้วยคำสั่ง '22' ซึ่งจะแสดงภาพ Logo ETT ออกที่หน้าจอ LCD โดยพิมพ์คำสั่งดังนี้

Esc 00 # 22 = 6 Enter

จากคำสั่งให้กด key Esc ตามด้วย 00 ตามด้วย # ตามด้วย 22 ตามด้วย = ตามด้วย 6 และสุดท้ายกด Enter ไม่ต้องมีเว้นวรรค ซึ่งที่หน้าจอ Hyper Terminal จะแสดงดังรูปด้านล่าง



จะเห็นว่าคำสั่งที่พิมพ์ส่งออกไปจะไม่แสดงเลข 0 ตัวหน้าให้เห็นที่หน้าจอของ Hyper Terminal ซึ่งเกิดจากการกด Key Esc ก่อน แต่ไม่เป็นไร Code ได้ถูกส่งออกไปแล้ว เมื่อพิมพ์เสร็จกด Key Enter คำสั่งก็จะถูกกระทำ หลังทำคำสั่งเสร็จตัวบอร์ด GLCD ก็จะส่ง Respond ออกมาให้เห็นตามรูปแบบที่ได้กล่าวไว้ข้างต้น

7.2 ตัวอย่างโปรแกรม ส่งคำสั่งด้วย MCU

ตัวอย่างที่ 1 ตัวอย่างนี้จะส่งคำสั่งด้วย MCU Z8 Encore(Z8F6422) เขียนด้วยภาษา C บน Compiler ZDS II กำหนดให้ส่งคำสั่งผ่านทาง Uart0(RS232:CH0) , Baud Rate =115200 bit/s และกำหนดให้บอร์ด GLCD Set ID = '00' โดยเขียนโปรแกรมส่งคำสั่ง เพื่อทำการวาดตารางดังรูปด้านล่างออกบนหน้าจอ GLCD

บริษัท อีทีที จำกัด (2007)

ET-ROBOTICS	ราคา(บาท)
R-MOTOR	210.-
R-TRACKER3	350.-

```

/*****
**                                     **
**   Ex1.Buile Table On Display GLCD   **
**                                     **
*****/

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

//----- Receive Data -----

unsigned char Rx()
{
    while(!(U0STAT0 & 0x80)){;}
    return U0D
}

//----- Check Respond Command -----

void acho()
{
    unsigned char P,K;

    do{
        P = Rx()          ; //Read Respond 'O'
    }while(P != 'O') ; //ถ้าไม่ใช่ 'O' กลับไปอ่านต่อ

    do{
        K = Rx()          ; //Read Respond 'K'
    }while(K != 'K') ; //ถ้าไม่ใช่ 'K' กลับไปอ่านต่อ

}

//----- Main -----

void main(void)
{
    unsigned char esc=0x1B,enter=0x0D,sp=0x01;

    U0BRH  = 0x00 ;
    U0BRL  = 10   ; //Set Baud Rate 11520
    PAAF   = 0x30 ; //Set Alternate Pin PA4-5 for Urat 0
    U0CTL0 = 0xC0 ; //Control Register Uart0 Function

```

```

//----- Buile Table On Display GLCD -----

printf("%c00#00=%c",esc,enter)      ; //Cmm Clear Screen
acho();

printf("%c00#30=000,00%c",esc,enter); //Cmm Set Cursor Start Text(0,0)
acho();

printf("%c00#32=1:บริษัท อีทีที จำกัด (2007)%c",esc,enter); //Cmm write Text close Left
acho();

printf("%c00#14=000,16,127,63,0%c",esc,enter) ; //Cmm Plot Rectang
acho();

printf("%c00#30=003,13%c",esc,enter)           ; //Cmm Set Cursor Start Text(3,13)
acho();

printf("%c00#31=1:ET-ROBOTICS ราคา(บาท)%c",esc,enter); //Cmm write Text
acho();

printf("%c00#16=000,28,126,28,1%c",esc,enter) ; //Cmm Plot Line Thick
acho();

printf("%c00#15=070,16,070,63%c",esc,enter)    ; //Cmm Plot Line แนวตั้ง
acho();

printf("%c00#30=003,28%c",esc,enter)           ; //Cmm Set Cursor Start Text(3,28)
acho();

printf("%c00#31=1:R-MOTOR          210.-%c",esc,enter); //Cmm write Text
acho();

printf("%c00#15=000,45,126,45%c",esc,enter)    ; //Cmm Plot Line แนวนอน
acho();

printf("%c00#30=003,45%c",esc,enter)           ; //Cmm Set Cursor Start Text(3,45)
acho();

printf("%c00#31=1:R-TRACKER3      %c%c350.-%c",esc,sp,sp,enter); //Cmm write Text
acho();

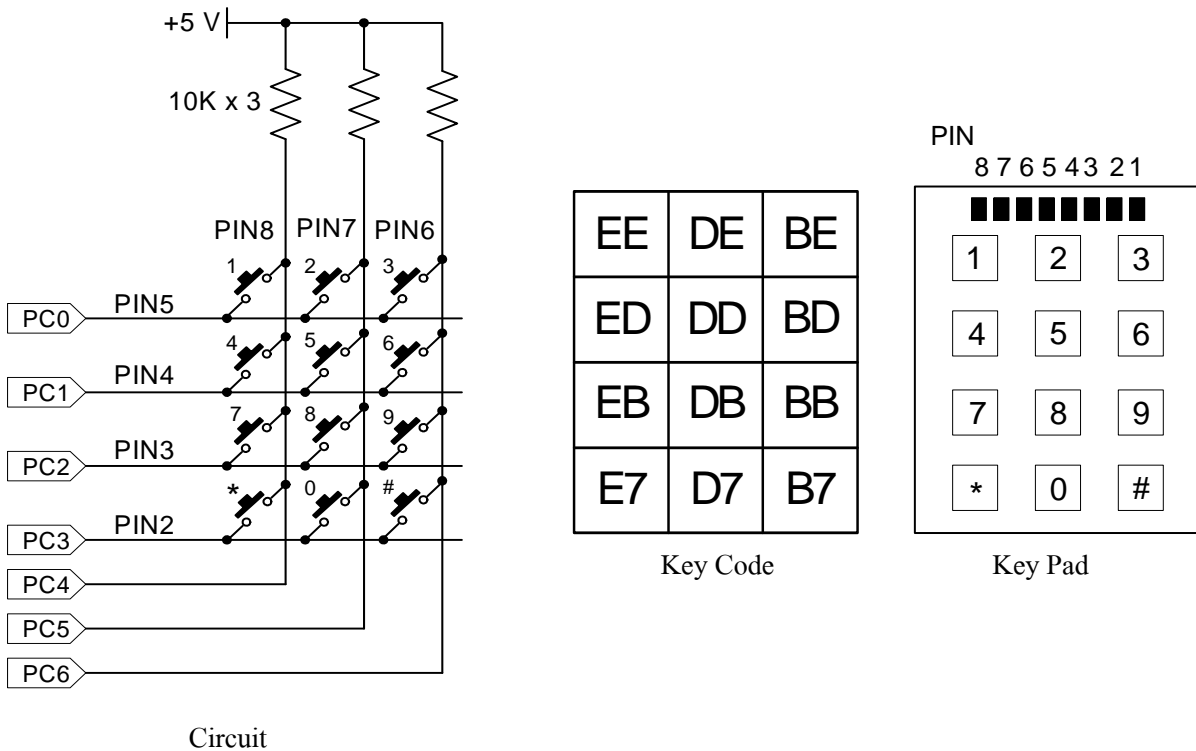
}

```

จากโปรแกรมนี้จะเห็นว่าเราจะตรวจสอบ Respond เฉพาะ 2 ตัวสุดท้ายคือตัว O และ K เท่านั้นก็ได้ ซึ่งโปรแกรมนี้ก็จะทำการวาดตารางและเขียนข้อความลงไปด้วย ดังที่แสดงในรูปด้านบน สำหรับตัวแปร sp=0x01 นี้จะใช้สำหรับส่งข้อมูลไปแทนตัวอักษรเพื่อต้องการให้เว้นวรรคบนหน้าจอ GLCD ครั้งละ 1 dot เพื่อจัดแต่งตำแหน่งตัวอักษรให้ตรงกัน

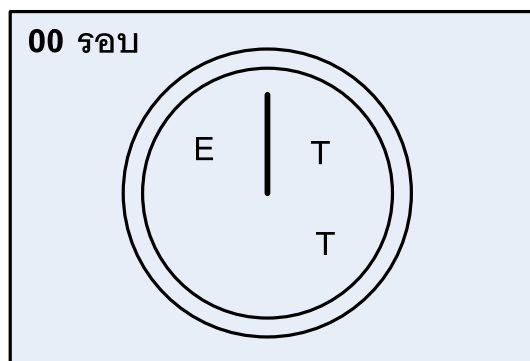
ตัวอย่างที่ 2 ตัวอย่างนี้จะส่งคำสั่งด้วย MCU Z8 Encore(Z8F6422) เขียนด้วยภาษา C บน Compiler ZDS II กำหนดให้ส่งคำสั่งผ่านทาง Uart0(RS232:CH0) , Baud Rate =115200 bit/s และกำหนดให้บอร์ด GLCD Set ID = '00' โดยเขียนโปรแกรมรับค่าจาก Key Matrix 4x3 เข้ามาทาง Port C จากนั้นนำคีย์ที่กดไปแสดงออกหน้าจอ GLCD เมื่อกด key ตัวเลขครบ 10 ตัวก็จะมีเสียงดังขึ้นที่บอร์ด GLCD โดยกำหนดให้ Key * ทำหน้าที่ Clear ตัวเลขที่กดทั้งหมด ส่วน Key # ทำหน้าที่

ลบเลขที่อยู่หน้า Cursor ครั้งละหลัก ดู Source Code ของตัวอย่างนี้ได้ในแผ่น CD ที่ให้มา ในตัวอย่าง Ex2.CmmGlcd และดูวงจรการต่อ Key เข้ากับ MCU Z8F6422 ได้ในรูปแบบที่ 7.2.1



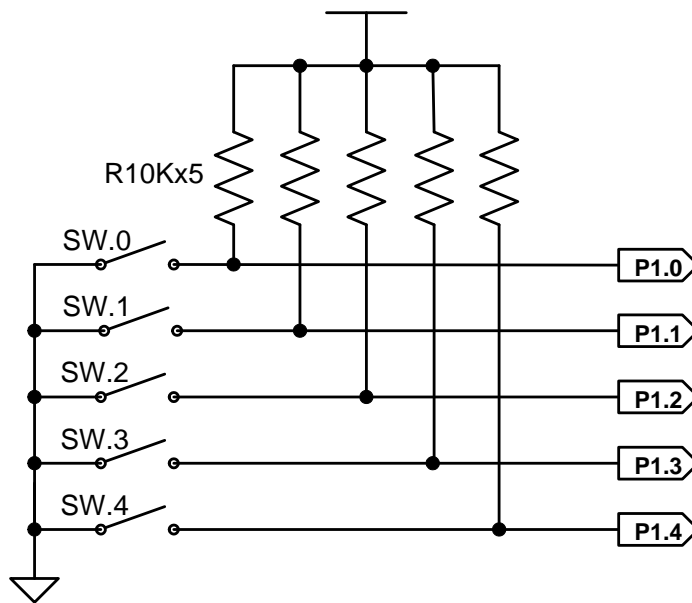
รูปที่ 7.2.1 การต่อ Key Switch 4x3 สำหรับตัวอย่างที่ 2

ตัวอย่างที่ 3 ตัวอย่างนี้จะส่งคำสั่งด้วย MCU AT89C51ED2 เขียนด้วยภาษา C บน Keil μ VISION 3 V3.33 กำหนดให้ส่งคำสั่งผ่านทาง Uart(RS232) , Baud Rate = 115200 bit/s และกำหนดให้บอร์ด GLCD Set ID = '00' โดยจะเขียนโปรแกรมส่งคำสั่ง วาดรูปวงกลมออกหน้าจอและภายในวงกลมก็จะมีเข็มหมุนอยู่ เหมือนนาฬิกา เมื่อหมุนครบรอบก็จะแสดงจำนวนรอบให้เห็นที่มุมบนด้านซ้ายของจอ GLCD ด้วย ดู Source Code ของตัวอย่างนี้ได้ในแผ่น CD ที่ให้มา ในตัวอย่าง Ex3.CmmGlcd

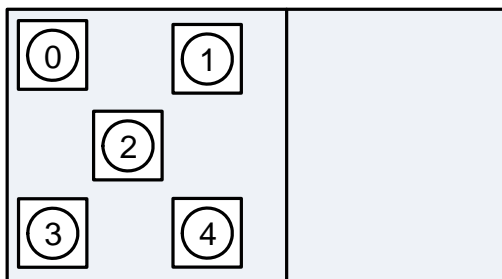


รูปที่ 7.2.2 แสดงตัวอย่างหน้าจอ GLCD เมื่อ Run โปรแกรมตัวอย่างที่ 3

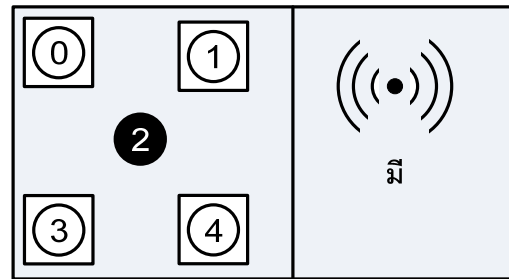
ตัวอย่างที่ 4 ตัวอย่างนี้จะส่งคำสั่งด้วย MCU AT89C51ED2 เขียนด้วยภาษา C บน Keil μ VISION 3 V3.33 กำหนดให้ส่งคำสั่งผ่านทาง Uart(RS232) , Baud Rate =115200 bit/s และกำหนดให้บอร์ด GLCD Set ID = '00' โดยให้ต่อ SW. 5 ตัว เข้าทาง Port P1.0-P1.4 ของ MCU ดังแสดงในรูปที่ 7.2.3 จากนั้นเขียนโปรแกรมแสดงปุ่มสวิตช์บนหน้าจอ GLCD ดังแสดงในรูปที่ 7.2.4(ก) เมื่อมีการกด SW. ปุ่มใดๆจากภายนอก ก็ให้รูปปุ่ม SW.บนหน้าจอ GLCD ของปุ่มที่ถูกกด แสดงสถานะการถูกกดด้วย รวมทั้งมีเสียงตัวโน้ตดังขึ้น ดู Source Code ของตัวอย่างนี้ได้ในแผ่น CD ที่ให้มา ในตัวอย่าง Ex4.CmmGlcd



รูปที่ 7.2.3 แสดงวงจรการต่อ SW.เข้ากับ Port P1.0-P1.4



ก) รูปหน้าจอ ขณะไม่กด SW.



ข) รูปหน้าจอ ขณะกด SW.2

รูปที่ 7.2.4 แสดงรูปหน้าจอ GLCD เมื่อ RUN โปรแกรมตัวอย่างที่ 4

ข้อควรจำ อย่าลืมว่าในขณะที่เริ่มต้นจ่ายไฟให้กับบอร์ดส่งคำสั่งและบอร์ด GLCD ก่อนที่จะเริ่มส่งคำสั่งไปยังบอร์ด GLCD ต้องแน่ใจว่า บอร์ด GLCD นั้นอยู่ในสถานะพร้อมรับคำสั่งเสียก่อน ซึ่งในการเขียนโปรแกรมให้กับบอร์ดส่งคำสั่งก็ควรที่จะ delay ไว้ที่ส่วนต้นของโปรแกรม หรืออาจจะตรวจสอบ Sync Byte ที่ตัวบอร์ด GLCD ส่งออกมาให้ก่อนก็ได้

Appendix A : สรุปตารางคำสั่ง

1) COMMAND '00' (Clear Screen)

Start	ID	Mark1	Command	Mark2	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	00	=	Enter (0x0D)

2) COMMAND '01' (Invert Screen)

Start	ID	Mark1	Command	Mark2	Invert	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	01	=	0-1	Enter (0x0D)

Invert = '0' : Normal (default) , '1' : Invert

3) COMMAND '02' (Display Screen)

Start	ID	Mark1	Command	Mark2	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	02	=	Enter (0x0D)

4) COMMAND '03' (On/Off Back Light)

Start	ID	Mark1	Command	Mark2	Back Light	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	03	=	0-1	Enter (0x0D)

Back Light = '0' : OFF , '1' : ON

5) COMMAND '04' (Sound)

Start	ID	Mark1	Command	Mark2	Note	Mark3	Delay	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	1byte	1 byte
Esc (0x1B)	00-FF	#	04	=	0-7	,	0-9	Enter (0x0D)

Note = '0' : No Sound , '1' : เสียง โด (524 Hz) , '2' : เสียง เร (587 Hz) , '3' : เสียง มี (659 Hz)

'4' : เสียง ฟา (698Hz) , '5' : เสียง ซอล (784 Hz) , '6' : เสียง ลา (880 Hz) , '7' : เสียง ที (988 Hz)

Delay = '0' : ความยาวเสียงสั้นสุด และความยาวเสียงเพิ่มขึ้นเรื่อยๆตามหมายเลข โดย '9' ความยาวเสียงยาวสุด

6) COMMAND '10' (Plot Circle)

Start	ID	Mk1	Command	Mk2	Cx	Mk3	Cy	Mk4	Radius	Mk5	Fill	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	10	=	000-127	,	00-63	,	00-31	,	0-1	Enter (0x0D)

7) COMMAND '11' (Plot Ellipse)

Start	ID	Mk1	Command	Mk2	Cx	Mk3	Cy	Mk4	Rx	Mk5	Ry	Mk6	Fill	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	11	=	000-127	,	00-63	,	00-64	,	00-32	,	0-1	Enter (0x0D)

8) COMMAND '12' (Plot Dot)

Start	ID	Mark1	Command	Mark2	X	Mark3	Y	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte
Esc (0x1B)	00-FF	#	12	=	000-127	,	00-63	Enter (0x0D)

9) COMMAND '13' (Plot Triangle)

Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	Mk6
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte
Esc (0x1B)	00-FF	#	13	=	000-127	,	00-63	,	000-127	,	00-63	,

(ต่อ)

X3	Mk7	Y3	Mk8	Fill	END
3 byte	1 byte	2byte	1 byte	1 byte	1 byte
000-127	,	00-63	,	0-1	Enter (0x0D)

10) COMMAND '14' (Plot Rectangle)

Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	Mk6	Fill	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	14	=	000-127	,	00-63	,	000-127	,	00-63	,	0-2	Enter (0x0D)

Fill '0' = รูปเป็นลายเส้น , '1' = รูปเป็นสีดำทั้งรูป , '2' = รูปเป็นสีขาวทั้งรูป ไม่มีลายเส้น

11) COMMAND '15' (Plot Line)

Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte
Esc (0x1B)	00-FF	#	15	=	000-127	,	00-63	,	000-127	,	00-63	Enter (0x0D)

12) COMMAND '16' (Plot Line Thick)

Start	ID	Mk1	Command	Mk2	X1	Mk3	Y1	Mk4	X2	Mk5	Y2	Mk6	Thick	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2byte	1 byte	3 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	16	=	000-127	,	00-63	,	000-127	,	00-63	,	0-4	Enter (0x0D)

13) COMMAND '17' (Plot Polygon)

Start	ID	Mk1	Command	Mk2	NumPoint	Mk3	Position (x,y)	Mk6	Thick	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	X1,Y1, ... ,X9,Y9	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	17	=	3-9	,	000-127,00-63, ... ,000-127,00-63	,	0-4	Enter (0x0D)

NumPoint = จำนวนจุดหรือด้านของรูปที่จะวาดกำหนดได้ตั้งแต่ '3'-'9'

Position(x,y) = ค่าตำแหน่งพิกัดจุด X : 3 byte('000-127') , Y : 2 byte('00-63')

Thick = ความหนาของเส้นที่จะวาด กำหนดได้ตั้งแต่ '0'-'4'

14) COMMAND '20' (Write Picture)

Start	ID	Mark1	Command	Mark2	Data File BMP	END
1 byte	2 byte	1 byte	2 byte	1 byte	1024 byte	1 byte
Esc (0x1B)	00-FF	#	20	=	Hex Code	Enter (0x0D)

Hex Code = ข้อมูลของไฟล์ภาพ ที่ได้จากการ Convert ขนาด 1024 byte

15) COMMAND '21' (Demo Test GLCD)

Start	ID	Mark1	Command	Mark2	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	21	=	Enter (0x0D)

16) COMMAND '22' (Display Wall paper)

Start	ID	Mark1	Command	Mark2	Picture	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	22	=	'1-7'	Enter (0x0D)

Picture = เลือกรูปที่จะให้แสดงออกหน้าจอ GLCD (เลือกค่า '1-7')

17) COMMAND '30' (Set Cursor Position)

Start	ID	Mark1	Command	Mark2	X	Mark3	Y	END
1 byte	2 byte	1 byte	2 byte	1 byte	3 byte	1 byte	2 byte	1 byte
Esc (0x1B)	00-FF	#	30	=	000-127	,	00-63	Enter (0x0D)

18) COMMAND '31' (Write Message1)

Start	ID	Mark1	Command	Mark2	Mode	Mark3	Data Text	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	1-200 byte	1 byte
Esc (0x1B)	00-FF	#	31	=	0-1	:	ข้อความ	Enter (0x0D)

Mode = '0' : แสดงอักษรสีขาว , '1':แสดงอักษรสีดำ

Data Text = Data ASCII Thai-Eng :1-200 byte

19) COMMAND '32' (Write Message2)

Start	ID	Mark1	Command	Mark2	Mode	Mark3	Data Text	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte	1-200 byte	1 byte
Esc (0x1B)	00-FF	#	32	=	0-3	:	ข้อความ	Enter (0x0D)

Mode = '0' : แสดงอักษรครั้งละไม่เกิน 1 หน้าจอ และขึ้นบรรทัดใหม่ให้อัตโนมัติ

'1' : แสดงอักษรครั้งละไม่เกิน 1 บรรทัด และจัดข้อความให้ชิดซ้าย

'2' : แสดงอักษรครั้งละไม่เกิน 1 บรรทัด และจัดข้อความให้อยู่กึ่งกลางหน้าจอ

'3' : แสดงอักษรครั้งละไม่เกิน 1 บรรทัด และ จัดข้อความให้ชิดขวา

20) COMMAND '33' (Del Text ขนาด 7x5 dot)

Start	ID	Mark1	Command	Mark2	Mode	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	33	=	0-3	Enter (0x0D)

Mode = '0' : ลบตัวอักษรสีดำในตำแหน่งที่ Cursor ปรากฏ , '1' : ลบตัวอักษรสีขาวในตำแหน่งที่ Cursor ปรากฏ

'2' : ลบตัวอักษรสีดำที่อยู่ในตำแหน่งหน้า Cursor , '3' : ลบตัวอักษรสีขาวที่อยู่ในตำแหน่งหน้า Cursor

21) COMMAND '34' (ON/OFF Cursor)

Start	ID	Mark1	Command	Mark2	OFF/ON	END
1 byte	2 byte	1 byte	2 byte	1 byte	1 byte	1 byte
Esc (0x1B)	00-FF	#	34	=	0-1	Enter (0x0D)

OFF/ON = '0' : Cursor OFF

'1' : Cursor ON

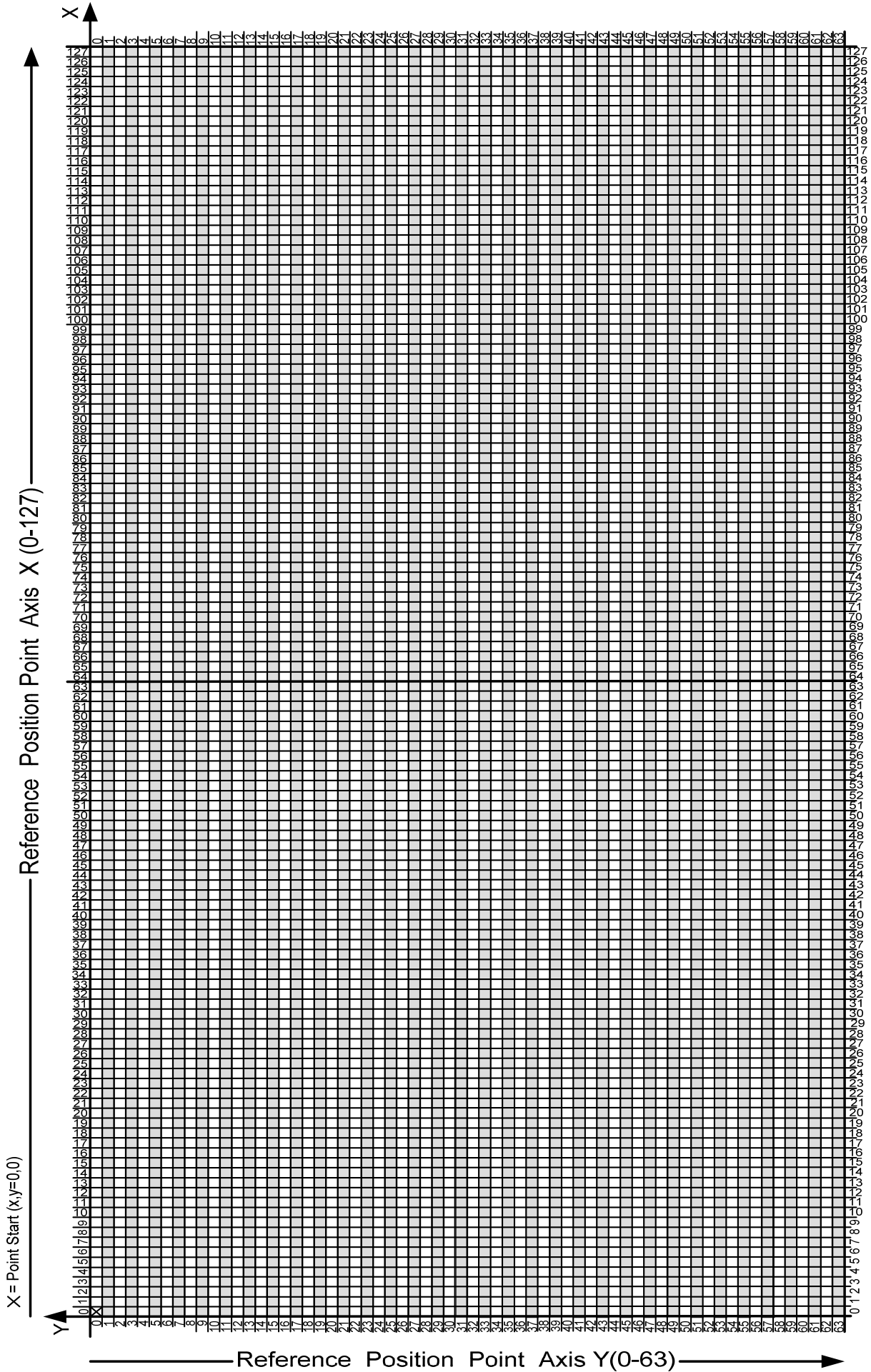
Appendix B: ตาราง ASCII Code สมอ.

Table Font ASCII Code สมอ.

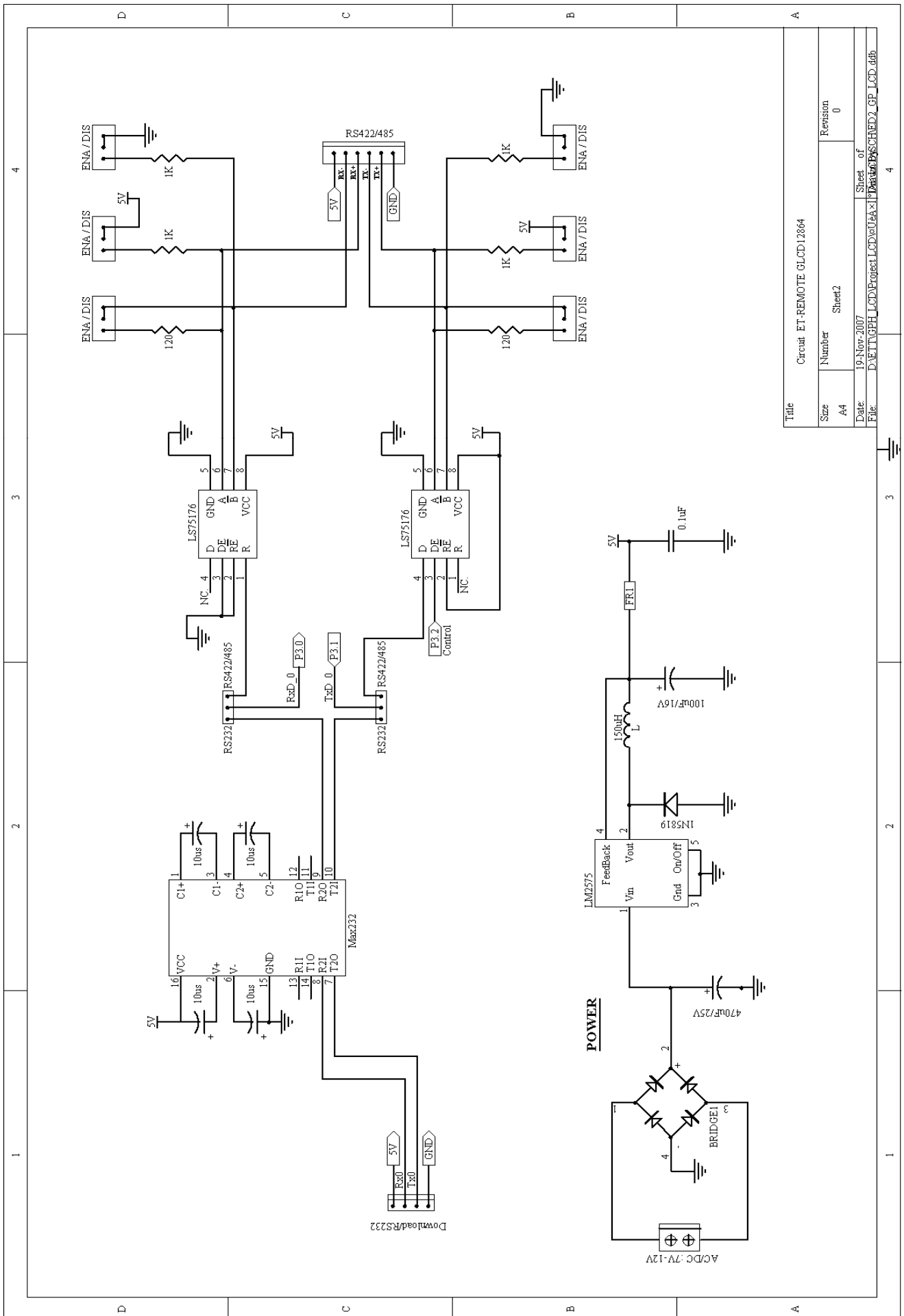
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	×	44	2C	,	88	58	X	132	84	×	176	B0	ฐ	220	DC	┌
1	1	×	45	2D	-	89	59	Y	133	85	...	177	B1	ฑ	221	DD	▼
2	2	×	46	2E	.	90	5A	Z	134	86	×	178	B2	ฒ	222	DE	▲
3	3	×	47	2F	/	91	5B	[135	87	×	179	B3	ณ	223	DF	▩
4	4	×	48	30	0	92	5C	\	136	88	×	180	B4	ด	224	E0	┌
5	5	×	49	31	1	93	5D]	137	89	×	181	B5	ต	225	E1	┌
6	6	×	50	32	2	94	5E	^	138	8A	×	182	B6	ถ	226	E2	┌
7	7	×	51	33	3	95	5F	~	139	8B	×	183	B7	ท	227	E3	┌
8	8	×	52	34	4	96	60	`	140	8C	×	184	B8	ธ	228	E4	┌
9	9	×	53	35	5	97	61	a	141	8D	×	185	B9	น	229	E5	×
10	0A	×	54	36	6	98	62	b	142	8E	×	186	BA	บ	230	E6	×
11	0B	×	55	37	7	99	63	c	143	8F	×	187	BB	ป	231	E7	×
12	0C	×	56	38	8	100	64	d	144	90	×	188	BC	ผ	232	E8	×
13	0D	×	57	39	9	101	65	e	145	91	'	189	BD	ฝ	233	E9	×
14	0E	×	58	3A	:	102	66	f	146	92	'	190	BE	พ	234	EA	×
15	0F	×	59	3B	;	103	67	g	147	93	"	191	BF	ฟ	235	EB	×
16	10	×	60	3C	<	104	68	h	148	94	"	192	C0	ภ	236	EC	×
17	11	◀	61	3D	=	105	69	i	149	95	•	193	C1	ม	237	ED	×
18	12	×	62	3E	>	106	6A	j	150	96	-	194	C2	ย	238	EE	×
19	13	×	63	3F	?	107	6B	k	151	97	_	195	C3	ร	239	EF	×
20	14	×	64	40	@	108	6C	l	152	98	×	196	C4	ฤ	240	F0	×
21	15	×	65	41	A	109	6D	m	153	99	×	197	C5	ล	241	F1	×
22	16	×	66	42	B	110	6E	n	154	9A	×	198	C6	ฎ	242	F2	×
23	17	×	67	43	C	111	6F	o	155	9B	×	199	C7	ว	243	F3	×
24	18	×	68	44	D	112	70	p	156	9C	×	200	C8	ศ	244	F4	×
25	19	×	69	45	E	113	71	q	157	9D	×	201	C9	ษ	245	F5	×
26	1A	×	70	46	F	114	72	r	158	9E	×	202	CA	ส	246	F6	×
27	1B	×	71	47	G	115	73	s	159	9F	×	203	CB	ห	247	F7	×
28	1C	×	72	48	H	116	74	t	160	A0	×	204	CC	ฬ	248	F8	×
29	1D	×	73	49	I	117	75	u	161	A1	ก	205	CD	อ	249	F9	×
30	1E	×	74	4A	J	118	76	v	162	A2	ข	206	CE	ฮ	250	FA	×
31	1F	×	75	4B	K	119	77	w	163	A3	ช	207	CF	ษ	251	FB	×
32	20		76	4C	L	120	78	x	164	A4	ค	208	D0	ะ	252	FC	×
33	21	!	77	4D	M	121	79	y	165	A5	ค	209	D1	ะ	253	FD	×
34	22	"	78	4E	N	122	7A	z	166	A6	ฅ	210	D2	า	254	FE	×
35	23	#	79	4F	O	123	7B	{	167	A7	ง	211	D3	า	255	FF	×
36	24	\$	80	50	P	124	7C		168	A8	จ	212	D4	า			
37	25	%	81	51	Q	125	7D	}	169	A9	ฉ	213	D5	า			
38	26	&	82	52	R	126	7E	~	170	AA	ช	214	D6	า			
39	27	'	83	53	S	127	7F	■	171	AB	ช	215	D7	า			
40	28	(84	54	T	128	80	×	172	AC	ฅ	216	D8	า			
41	29)	85	55	U	129	81	×	173	AD	ฅ	217	D9	า			
42	2A	*	86	56	V	130	82	×	174	AE	ญ	218	DA	า			
43	2B	+	87	57	W	131	83	×	175	AF	ญ	219	DB	▶			

* × : ไม่มีตัวอักษร

Appendix C : DISPLAY GRAPHIC LCD 128x64



BUFFER DISPLAY GRAPHIC LCD 128x64
 For use reference Point(x,y) when sent command



Title			
Circuit ET-REMOTE GLCD12864			
Size	Number	Sheet	Revision
A4	Sheet2	0	0
Date	19-Nov-2007		Sheet of
File	D:\ETT\GPH_LCD\Project\LCD\ET-Remote\PCB\CACHED2_GL_LCD.dtb		

รูป Circuit ET-Remote GLCD12864 Sheet2