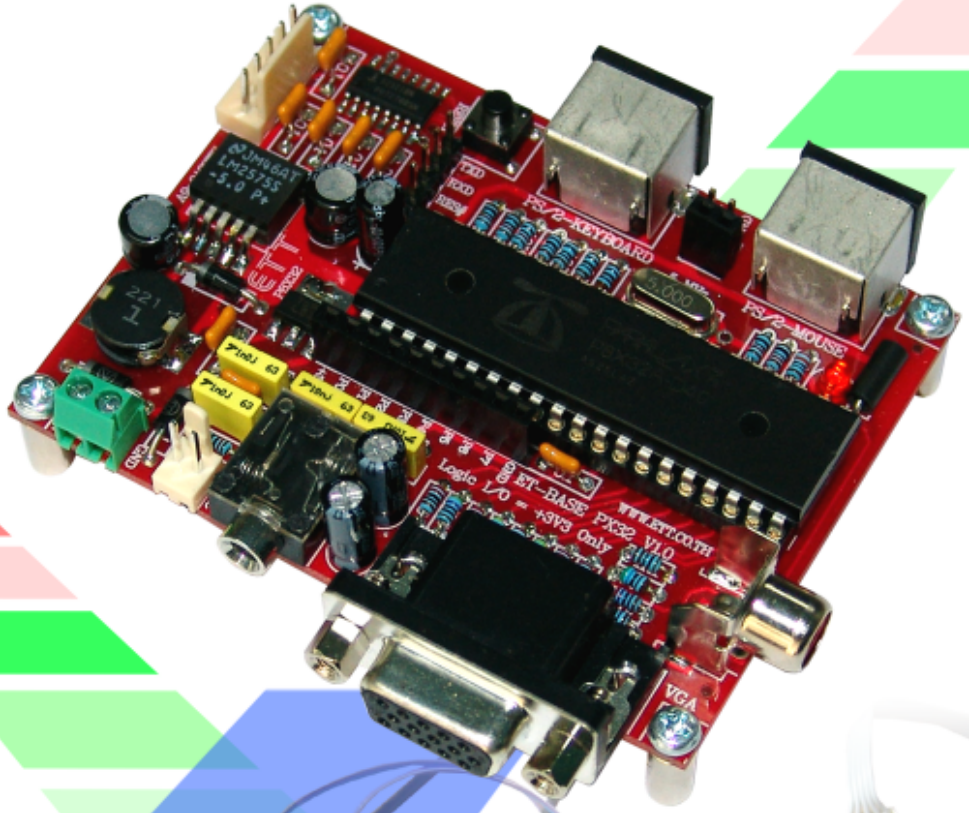
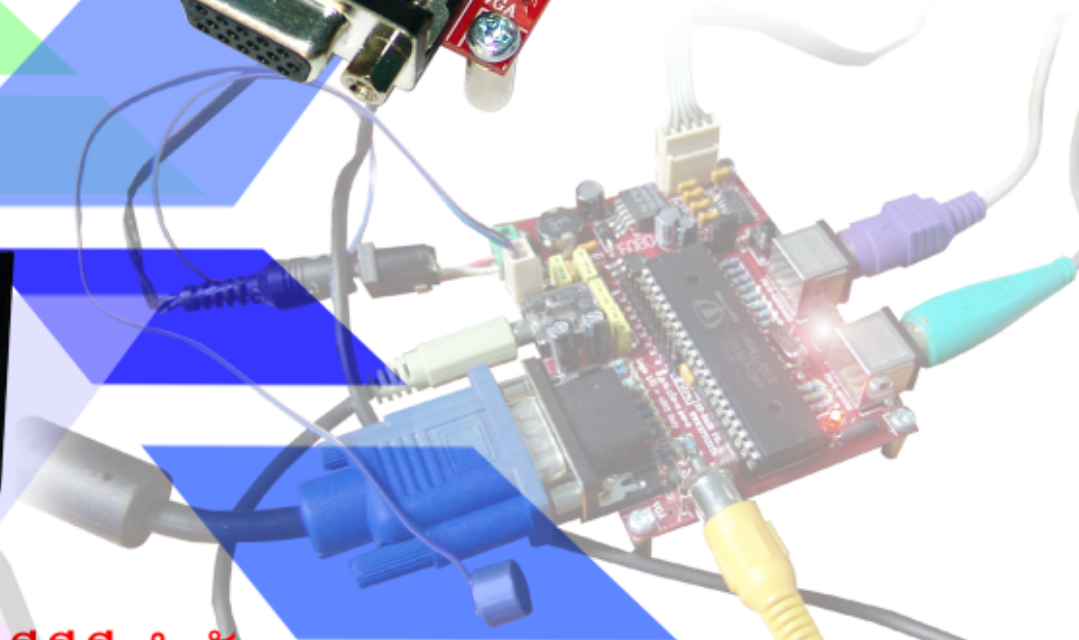
คู่มือการใช้งาน

User's Manual

ET-BASE PX32 V1.0





ETT 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. คุณสมบัติของ MCU P8X32A	2
2. คุณสมบัติของบอร์ด ET- BASE PX32 V1.0	2
3. โครงร้างและบล็อกไดอะแกรมของ MCU P8X32A	3
4. โครงสร้างของบอร์ด ET-BASE PX32 V1.0	4
5. การใช้งานโปรแกรม Propeller เบื้องต้น	6
6. การทดสอบ RUN ตัวอย่างโปรแกรม	9
7. การต่อสาย Download และการต่อสายสื่อสาร RS232	10
8. ข้อกำหนดที่ควรรู้ในการพัฒนาโปรแกรมด้วยภาษา SPIN บน Tool Propeller	12
9. วงจรบอร์ด ET-BASE-PX32 V1.0	16

ET- BASE PX32 V1.0

สำหรับบอร์ด ET-BASE PX32 นี้จะเป็นบอร์ด Training โดยใช้ MCU เบอร์ P8X32A –D40 ของ PARALLAX โดยจะเป็น MCU ที่มีความไวสูง ขนาด 32 บิต 8 Cog Multiprocessor โครงสร้างของ MCU จะเป็นตัวถังแบบ DIP 40 PIN สามารถทำงานได้ที่ความถี่สูงสุด 80 MHz ทำงานที่แรงดัน 2.7-3.6 VDC การพัฒนาโปรแกรมจะใช้ Software tool "Propeller V1.06" ซึ่ง Software ตัวนี้สามารถใช้เขียนโปรแกรม, Compile Code และ Download Code ผ่านทาง RS232 ได้เลย(ไม่สามารถ Debug ดูการทำงานเป็น STEP ได้) โดยภาษาที่ใช้ในการเขียนโปรแกรมจะเป็นภาษา SPIN ซึ่งจะช่วยให้พัฒนาโปรแกรมได้ง่ายและรวดเร็วขึ้น เนื่องจากใน โปรแกรม Propeller นี้ จะมี Library ต่างๆสำหรับใช้ติดต่อระหว่างอุปกรณ์รอบข้างกับตัว MCU P8X32A ไว้ให้เรียบร้อยแล้ว ซึ่งเวลาใช้งานผู้ใช้ก็สามารถเรียก Library มาใช้ได้เลย ตัวอย่างเช่น Library RS232 , Library VGA , Library TV , Library Keyboard , Mouse เป็นต้น

1. คุณสมบัติของ MCU P8X32A

1.1) เป็น MCU 32 bit 8 Cog Multiprocessor (8 CPU ใน Chip เดียว)

1.2) ตัวถังแบบ DIP 40 PIN มี Port I/O 32 Pin

1.3) ตัว MCU ทำงานที่แรงดัน 2.7-3.6 VDC และ I/O Port สามารถขับกระแส Source/Sink ได้ 40 mA ที่ 3.3 VDC

1.4) ทำงานได้ที่ความถี่สูงสุด 80 MHz สามารถเลือกใช้งาน External Clock หรือ Internal Clock ได้ มี PLL อยู่ภายใน

1.5) มี RAM ภายในสำหรับเก็บ Code 32 Kbyte ซึ่งเวลาตัดไฟเลี้ยง MCU ออก Code ก็จะถูกลบ ดังนั้นเวลาใช้งานจริงจะต้องต่อ I2C EEPROM ไว้ภายนอกเพื่อเก็บ Code

1.6) Pin ที่ถูกกำหนดให้ทำงานเป็น Input จะสามารถรับระดับแรงดัน Input ได้ไม่เกิน VDD(2.7-3.6V) เท่านั้น

1.7) ความเร็วในการทำงานภายในของ Chip จะอยู่ที่ 20 MIPS/cog

2. คุณสมบัติของบอร์ด ET- BASE PX32 V1.0

2.1) ใช้ MCU P8X32A ตัวถังแบบ DIP 40 PIN

2.2) มี I2C EEPROM #24LC256 (32Kb) สำหรับใช้เก็บ Code Program

2.3) ตัวบอร์ดรับแรงดัน Input 6-12 VDC โดยมีชุด Regulator ปรับแรงดันลงให้เหลือ 5V และ3.3V อยู่ในบอร์ด

2.4) ใช้ Crystal 5.00 MHz (External) สามารถใช้ PLL ภายในตัว Chip คูณเพิ่มความถี่ให้สูงขึ้นได้ถึง 80 MHz

2.5) พัฒนาโปรแกรมด้วยภาษา SPIN โดยใช้ Tool "Propeller" (Freeware) ซึ่ง Tool นี้จะใช้สำหรับเขียนโปรแกรม , Compile และ Download ภายในตัว และจะมี Library ให้ใช้ในการ Interface ระหว่างตัว MCU กับอุปกรณ์รอบข้างที่จัดไว้ให้บนบอร์ดเรียบร้อยแล้ว และสามารถเข้าไป Download Library เพิ่มเติมได้ที่ http://www.parallax.com/ ในส่วนของการใช้งานภาษา SPIN ก็สามารถดูได้จาก Help ของโปรแกรม Propeller

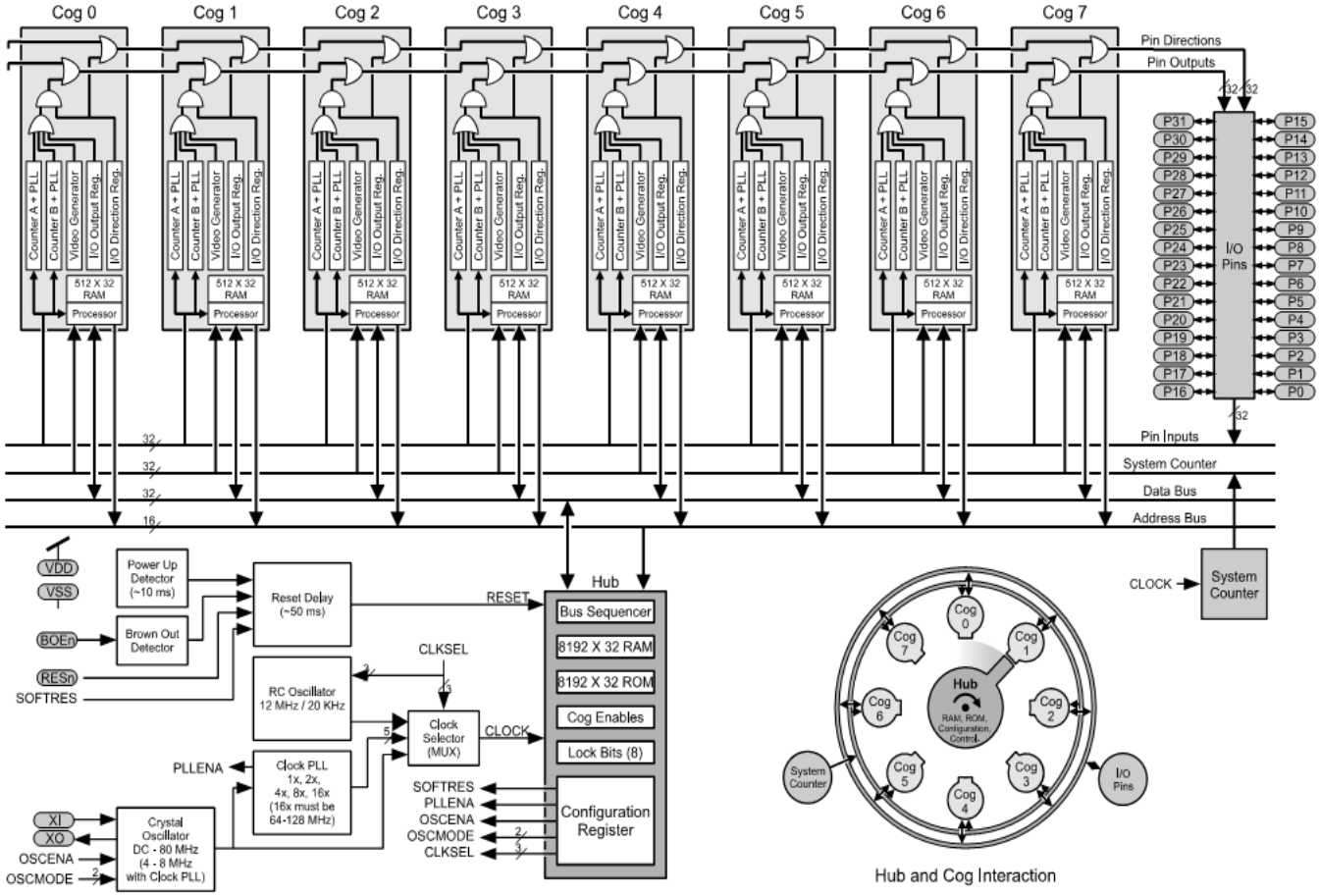
2.4) การ Download Code จะ Download ผ่านทาง RS232 โดยสามารถเลือกจากตัว Propeller ได้ว่า จะ Download Code ไปเก็บไว้ยัง EEPROM (External) หรือ RAM ภายใน MCU

2.5) บอร์ดนี้ได้จัดสรร I/O ไว้ตายตัวสำหรับ Interface กับอุปกรณ์รอบข้างดังนี้

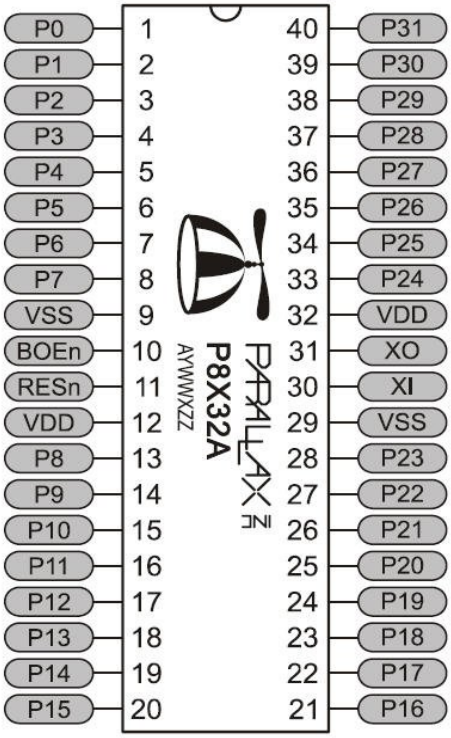
1 . Port Key Board(PS2) , 2. Port Mouse (PS2) , 3. Port RS232 , 4. Port VGA , 5. Port TV(AV)

6. Port MIC , 7. Port Headphone , 8. Port I/O 8 PIN สำหรับใช้งานด้านอื่นๆ

3. โครงร้างและบล็อกไดอะแกรมของ MCU P8X32A



รูปที่ 3.1 แสดงบล็อกไดอะแกรมของ Chip P8X32A



รูปที่ 3.2 แสดงโครงสร้างของ Chip P8X32A(DIP 40 PIN)

จากรูปที่ 3.1 จะเห็นว่าโครงสร้างภายในของ Chip จะประกอบไปด้วย Processor 8 ตัว โดยจะเรียกว่า “Cog” ซึ่งถูกออกแบบให้มีการทำงานที่ความไวสูง สิ้นเปลืองพลังงานต่ำ ตัวถังมีขนาดเล็ก มีความคล่องตัวในการทำงานผ่านทาง Processor

ทั้ง 8 ตัวสูง โดยสามารถทำงานได้ในเวลาเดียวกันพร้อมๆกัน และเป็นอิสระต่อกัน ซึ่ง Chip นี้จะใช้การ Share ทรัพยากรผ่านทางศูนย์กลาง HUB เพื่อให้ Cog แต่ละ Cog สามารถใช้งานทรัพยากรร่วมกันได้ ในส่วนของระบบ Clock จะถูก Share ไปยัง Cog แต่ละ Cog โดยจะอ้างอิงที่ฐานเวลาเดียวกัน ส่วนการ Interrupt จะไม่ถูกใช้กับ Chip นี้ แต่จะใช้การกำหนดตำแหน่งที่จะกระโดดไปทำงานให้กับ Cog ต่างๆโดยตรงเลย

ในรูปที่3.2 จะเป็นโครงสร้างตัวถังแบบ DIP 40 PIN ของ Chip โดยจำแนก PIN ต่างๆได้ดังนี้

ตารางที่3.1 รายละเอียด PIN

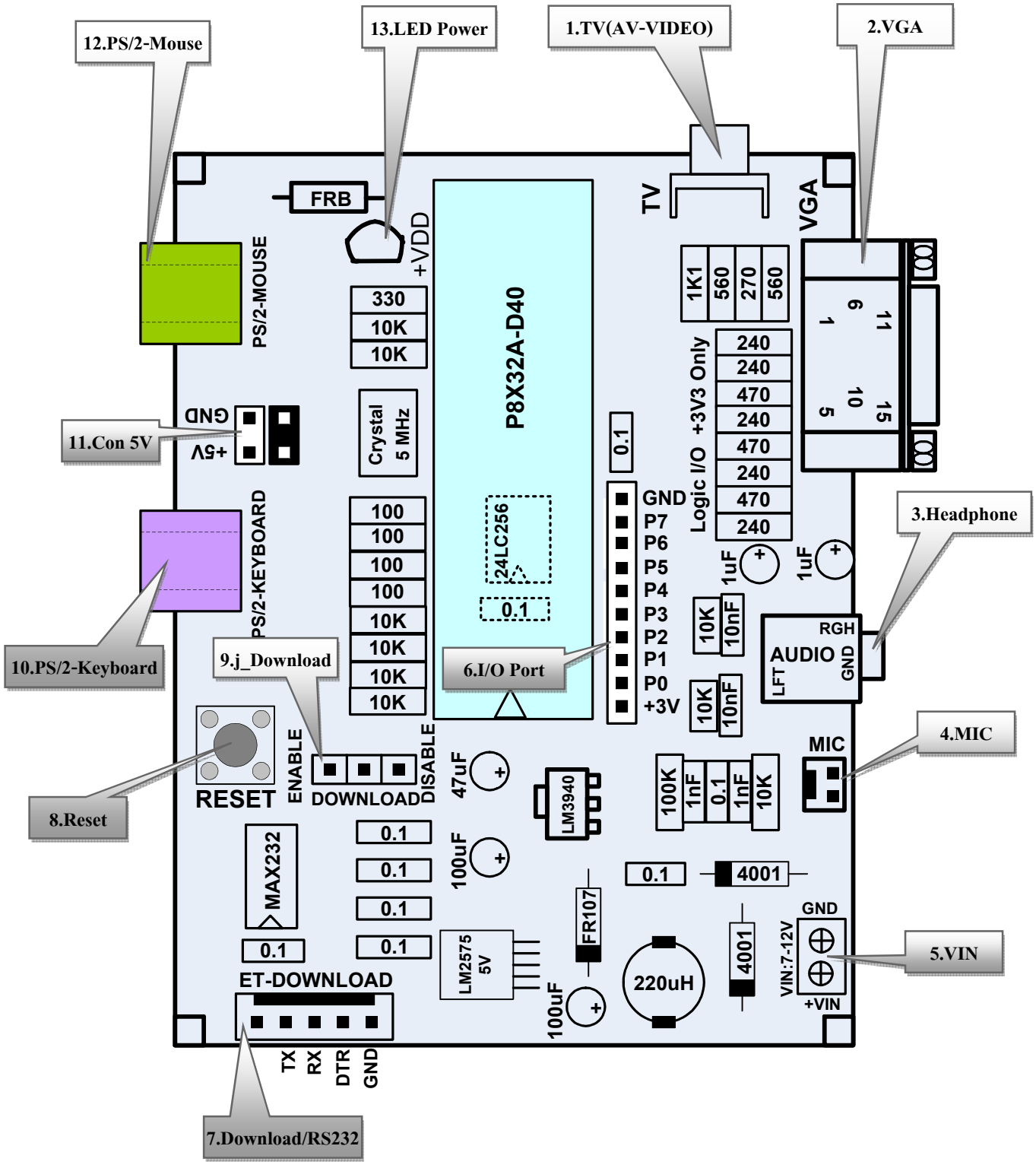
PIN Name	Direction	Description
P0-P31	I/O	เป็น Port I/O โดยมีระดับ Logic อยู่ที่ประมาณครึ่งหนึ่งของ VDD หรือ 1.6 VDC ที่แรงดัน 3.3 VDC ใน 32 Pin นี้จะมีอยู่ด้วยกัน 4 Pin ซึ่งจะถูกกำหนดให้ทำงานในหน้าที่พิเศษหลังจาก Power-up หรือ Reset คือ P28 – I2C SCL ซึ่งจะใช้ต่อไปยัง EEPROM ภายนอก P29 – I2C SDA ซึ่งจะใช้ต่อไปยัง EEPROM ภายนอก P30 – Serial Tx ใช้ต่อสำหรับ Download Code และส่งข้อมูล ผ่านทาง RS232 P31 – Serial Rx ใช้ต่อสำหรับ Download Code และรับข้อมูล ผ่านทาง RS232
VDD	---	3.3 V Power (2.7-3.6 VDC)
VSS	---	Ground
BOEn	I	Brown Out Enable(Active Low) จะใช้ต่อไปยัง VDD หรือ VSS ถ้าขานี้เป็น Low ขา RESn จะกลายเป็น Output แต่จะยังสามารถ Drive Low เพื่อ Reset Chip ได้ ถ้าเป็น high ขา RESn จะทำหน้าที่เป็น Input
RESn	I/O	Reset(Active Low) เมื่อเป็น Low ตัว Chip และ Cog ทั้งหมด จะถูก Disable PIN I/O จะถูกปล่อยลอย และ Chip จะ Restart ภายในเวลา 50ms หลังจาก Logic ที่ RESn เปลี่ยนจาก Low เป็น High
XI	I	Crystal Input
XO	O	Crystal Output

4. โครงสร้างของบอร์ด ET-BASE PX32 V1.0

สำหรับบอร์ด ET-BASE PX32 V1.0 นี้ ได้กำหนด Port ในการ Interface กับอุปกรณ์รอบข้างเฉพาะอย่างไว้ให้เรียบร้อยแล้ว จะเหลือในส่วนของ Port I/O (P0-P7)อีก 8 Pin เท่านั้น ที่จัดไว้สำหรับให้ผู้ใช้นำไปใช้ต่อกับอุปกรณ์อื่นๆเพิ่มเติมได้ เช่น LED , SW, SD Card , LCD เป็นต้น

ตารางที่4.1 แสดง Port ที่ถูกต่อใช้งานบนบอร์ด ET-BASE PX32

Port Number	หน้าที่
P0-P7	ถูกกำหนดให้เป็น I/O ที่ผู้ใช้สามารถนำไปต่อใช้งานอื่นๆเพิ่มเติมได้
P8-P9	ถูกกำหนดให้ใช้สำหรับต่อ Microphone
P10-P11	ถูกกำหนดให้เป็น AUDIO Out Stereo สามารถต่อไปยังเครื่องขยาย หรือ Headphone ได้
P12-P15	ถูกกำหนดให้เป็น Port Video Out (NTSC/PAL)สำหรับต่อ เข้าช่อง AV ของ TV
P16-P23	ถูกกำหนดให้ใช้เป็น Port VGA สำหรับต่อเข้ากับจอ PC หรือ จอ LCD
P24-P25	ถูกกำหนดให้ใช้สำหรับต่อ PS/2 Mouse
P26-P27	ถูกกำหนดให้ใช้สำหรับต่อ PS/2 Key Board
P28-P29[System]	ถูกต่อเข้ากับ I2C-EPPROM เพื่อใช้เก็บ Code Program (P28:Clock ; P29:Data)
P30-P31[System]	ถูกกำหนดให้ใช้เป็น Port RS232 เพื่อ Download Program และรับ-ส่งข้อมูลทาง RS232(P30:Rx;P31:Tx)



รูปที่4.1 แสดงโครงสร้างของบอร์ด ET-BASE PX32 V1.0

หมายเลข1 : TV(AV-VIDEO) เป็นขั้วต่อ VIDEO OUT เพื่อใช้ต่อกับ TV โดยต่อเข้าที่ขั้วต่อ VIDEO IN ของ TV

หมายเลข2 : VGA เป็นขั้วต่อ VGA OUT (DB15) ใช้สำหรับต่อไปยัง Monitor ของ PC เช่น จอ LCD

หมายเลข3 : Headphone เป็นขั้วต่อ AUDIO OUT ใช้สำหรับต่อหูฟัง หรือต่อเข้ากับชุดขยายสัญญาณ

หมายเลข4 : MIC เป็นขั้วต่อ Microphone

หมายเลข5 : VIN เป็นขั้วต่อไฟเลี้ยงบอร์ด DC 7V-12 V ในการต่อจะต้องระวัง ต่อขั้วบวก-ลบ ให้ถูกต้องด้วย

หมายเลข6 : I/O Port เป็น Port I/O 8 Pin จัดไว้สำหรับให้ผู้ใช้สามารถต่อ I/O อื่นๆเพิ่มเติมได้ อาทิเช่น LCD, Key,LED หรือ SD Card เป็นต้น

ข้อควรระวัง สำหรับ I/O Port นี้ในกรณีที่ใช้งานเป็น Input จะรับแรงดันที่เข้ามายัง Port ได้ไม่เกิน VDD หรือ 3.3 V ห้ามใช้แรงดัน 5 V มาต่อเข้าโดยตรงเพราะจะทำให้ Port Pin เสียหายได้ ควรใช้ R มาต่อ Dividerให้เหลือ3.3V เสียก่อน

หมายเลข7 : Download/RS232 เป็นขั้วต่อใช้สำหรับ Download Program และใช้ ในการสื่อสาร รับ-ส่ง ข้อมูลผ่านทาง RS232 โดยมี Line Driver Max232 เป็นตัวปรับระดับสัญญาณ Rx,Tx จาก 3.3V ไปเป็น ±12V เพื่อให้ต่อใช้งานร่วมกับ PC ได้

หมายเลข8 : Reset เป็น SW. สำหรับใช้ Reset การทำงานของ MCU โดยจะทำงานที่ Logic 0

*หมายเลข9 :*Jumper Download เมื่อจะ Download Program ให้ Set jumper มาทางด้าน Enable ซึ่ง Jumper นี้จะทำหน้าที่ต่อขา RES ของ MCU เข้ากับขา DTR ของขั้ว Download ถ้า Set Jumper มาทางด้าน Disable จะเป็นการตัดขา RES ของ MCU ออกจากขา DTR ของขั้ว Download เพื่อป้องกันสัญญาณ Reset จาก PC หลังจากดาวน์โหลดเสร็จแล้ว และยังไม่ได้ถอดสาย Download ออก

หมายเลข10 : PS/2-Keyboard เป็นขั้วต่อ PS/2 ใช้สำหรับต่อ keyboard

หมายเลข11 : Con 5V เป็นขั้วต่อ DC 5 V Output สำหรับใช้ต่อไฟ 5 V จากบอร์ดไปใช้งานภายนอกได้

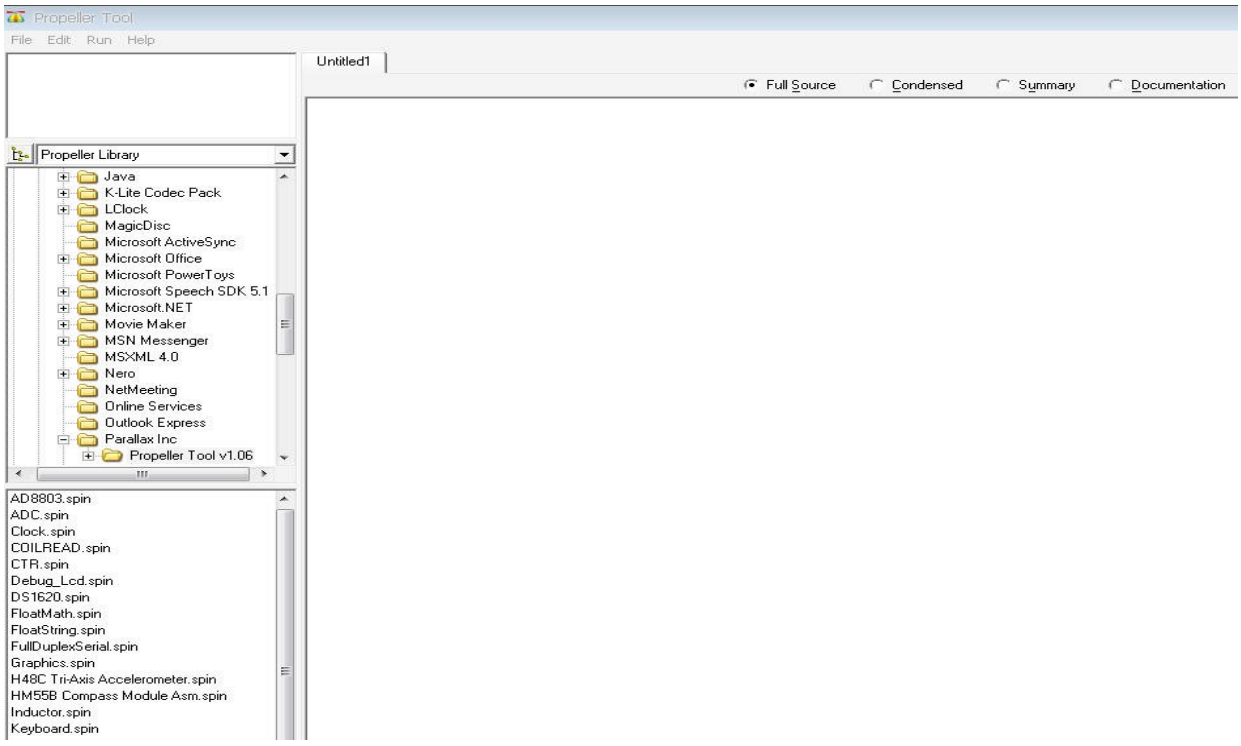
หมายเลข12 : PS/2-Mouse เป็นขั้วต่อ PS/2 ใช้สำหรับต่อ Mouse

หมายเลข13 : LED Power แสดงสะถานะการทำงานของแหล่งจ่ายไฟ

5. การใช้งานโปรแกรม Propeller เบื้องต้น

5.1) ทำการติดตั้งโปรแกรม Propeller V1.06 ลงในเครื่อง

5.2) หลังจากติดตั้งแล้วให้เปิดโปรแกรม Propeller ขึ้นมาจะได้หน้าต่างดังรูปที่ 5.2

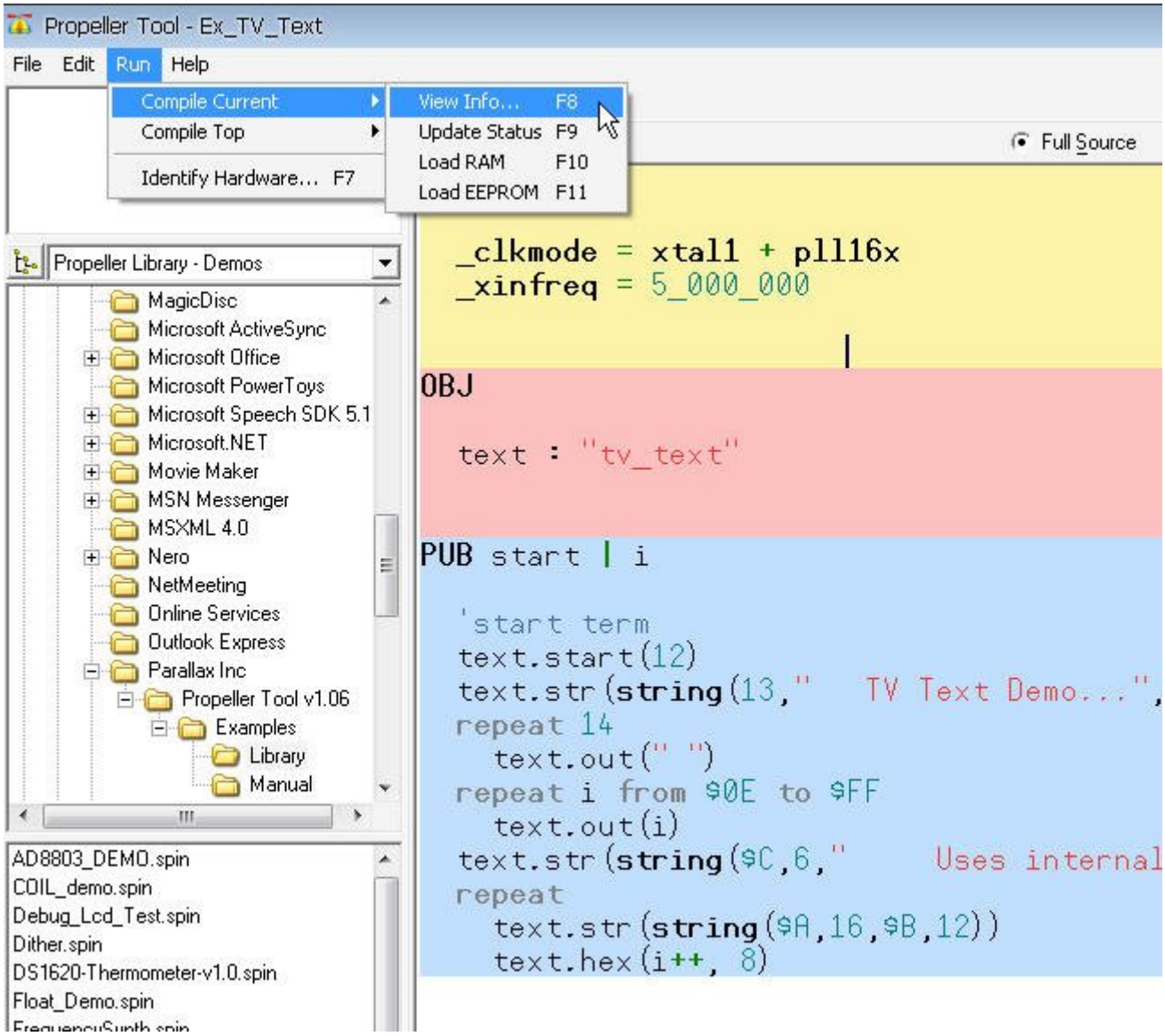


รูปที่5.2

จากรูปในหน้าต่างด้านขวามือ(แท็บUntitled1) จะเป็นพื้นที่ว่างสำหรับใช้เขียนโปรแกรม

5.3) หลังจากเขียนโปรแกรม เสร็จก็ให้ไปที่เมนู File เลือก Save As เพื่อทำการ Save File เป็นนามสกุลจุด spin

5.4) เมื่อ Save File เรียบร้อยแล้ว ก็ให้ไปที่เมนู RUN และเลือกที่ Compile Current จากนั้นก็จะมีเมนูให้ผู้ใช้ เลือกต่ออีก ดังแสดงในรูปที่ 5.3 ถ้าเลือก



รูปที่5.2

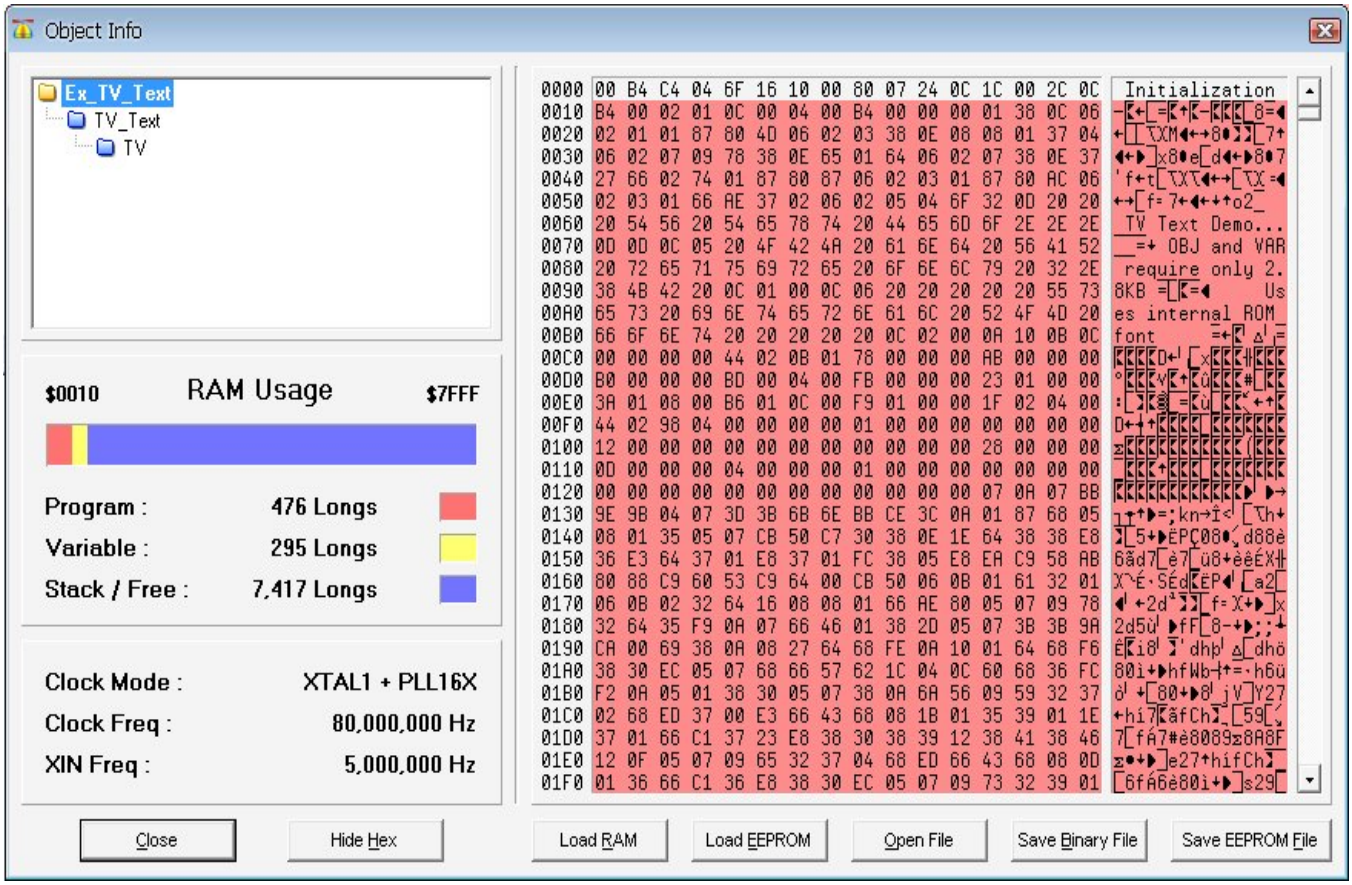
View Info… = จะเป็นการ Compile โปรแกรมอย่างเดียว และจะมีหน้าต่าง Object info รายงานข้อมูลต่างๆของโปรแกรมที่ เขียนขึ้น ดังรูปที่5.3 ซึ่งก็จะแสดงพื้นที่ RAM ที่ถูกใช้ไป แสดงความถี่ Clock ที่ผู้ใช้เลือกใช้งาน ซึ่งข้อมูลของตัวอย่างโปรแกรมในรูปที่ 5.3 นี้สามารถอธิบายได้คือ

- เป็นข้อมูลของไฟล์ที่ชื่อ Ex_TV_Text โดยภายในไฟล์นี้ ได้เรียกใช้งาน Function ที่อยู่ใน File Library ที่ชื่อ TV_Text และ ใน File library TV_Text ก็ได้เรียกใช้ Function ใน File library ที่ชื่อ TV อีกต่อหนึ่ง
- ไฟล์ Ex_TV_Text นี้ ใช้เนื้อที่ RAM ในส่วนที่เป็นโปรแกรมไปทั้งหมด 476 Long และในส่วนที่

เป็นตัวแปร ไปทั้งหมด 295 Long และเหลือพื้นที่ RAM ที่ยังว่างอยู่ทั้งหมด 7,417 Long โดย 1 Long จะมีค่าเท่ากับ 32 bit หรือ 4 byte

- ในโปรแกรมนี้ได้เลือกใช้ Clock Mode ในโหมด XTAL1+PLL16X โดยความถี่ Clock ที่ใช้จริงก็จะเท่ากับ 80 Mhz โดยใช้ Crystal จากภายนอก 5 MHz (5MHz x PLL16 = 80MHz)

หลังจากหน้าต่าง Object Info แสดงขึ้นมาแล้ว ให้ผู้ใช้เลือกคลิกที่ปุ่ม Load Ram หรือ ปุ่ม Load EEPROM เพื่อทำการ Download Program โดยถ้าเลือกปุ่ม Load RAM โปรแกรมจะถูก Load เข้าไปเก็บไว้ยังพื้นที่ RAM ของ MCU โดยตรง ซึ่งการโหลดวิธีนี้ โปรแกรมจะถูกลบเมื่อมีการกด SW. Reset หรือเอาไฟเลี้ยงบอร์ดออก , ถ้าเลือกปุ่ม Load EEPROM โปรแกรมจะถูก Load เข้าไปเก็บไว้ยัง EEPROM ที่ต่ออยู่ภายนอกก่อน จากนั้น ตัว MCU ก็จะทำการดึง Code จาก EEPROM เข้าไป RUN ใน RAM ให้อัตโนมัติ ซึ่งการโหลดวิธี นี้ โปรแกรมจะไม่ถูกลบเมื่อมีการกด SW. Reset หรือเอาไฟเลี้ยงบอร์ดออก



รูปที่5.3

Load RAM = จะเป็นการ Compile และ Download โปรแกรมลงใน RAM เลย จะไม่แสดงหน้าต่าง Object Info ให้เห็น โปรแกรมจะถูกลบ ถ้ามีการกด SW. Reset หรือ ตัดไฟเลี้ยงออก

Load EEPROM = จะเป็นการ Compile และ Download โปรแกรมลงใน EEPROM ที่ต่ออยู่ภายนอกจะไม่แสดงหน้าต่าง Object Infoให้เห็นเช่นกัน โปรแกรมจะยังคงอยู่ เมื่อ มีการ Reset หรือตัดไฟเลี้ยงออก

5.5) สำหรับการ Set Baud Rate หรือ เลือก COM Port เพื่อใช้ในการ Download Code นั้นจะไม่จำเป็นเพราะ ตัวโปรแกรม Propeller นั้นจะทำการ Set Baud Rate และหา Com Port ให้โดยอัตโนมัติ แต่ถ้าผู้ใช้ต้องการจะกำหนด Com Port เองก็สามารถเข้าไปแก้ไขได้โดยไปที่เมนู Edit แล้วเลือก Preferences จะมีหน้าต่างขึ้นมาให้เลือก ที่แท็บ Operation ในช่อง Serial Port Search : ให้เลือก Comport ที่จะใช้งาน (ปกติ = Auto)

5.6) สำหรับตัวโปรแกรม Propeller นี้จะใช้การพัฒนาโปรแกรมด้วยภาษา SPIN ซึ่งรูปแบบการใช้งานของภาษา SPIN นี้ผู้ใช้สามารถดูคำสั่งการใช้งานได้ โดยไปที่เมนู Help แล้วเลือกที่ Propeller Manual (pdf)

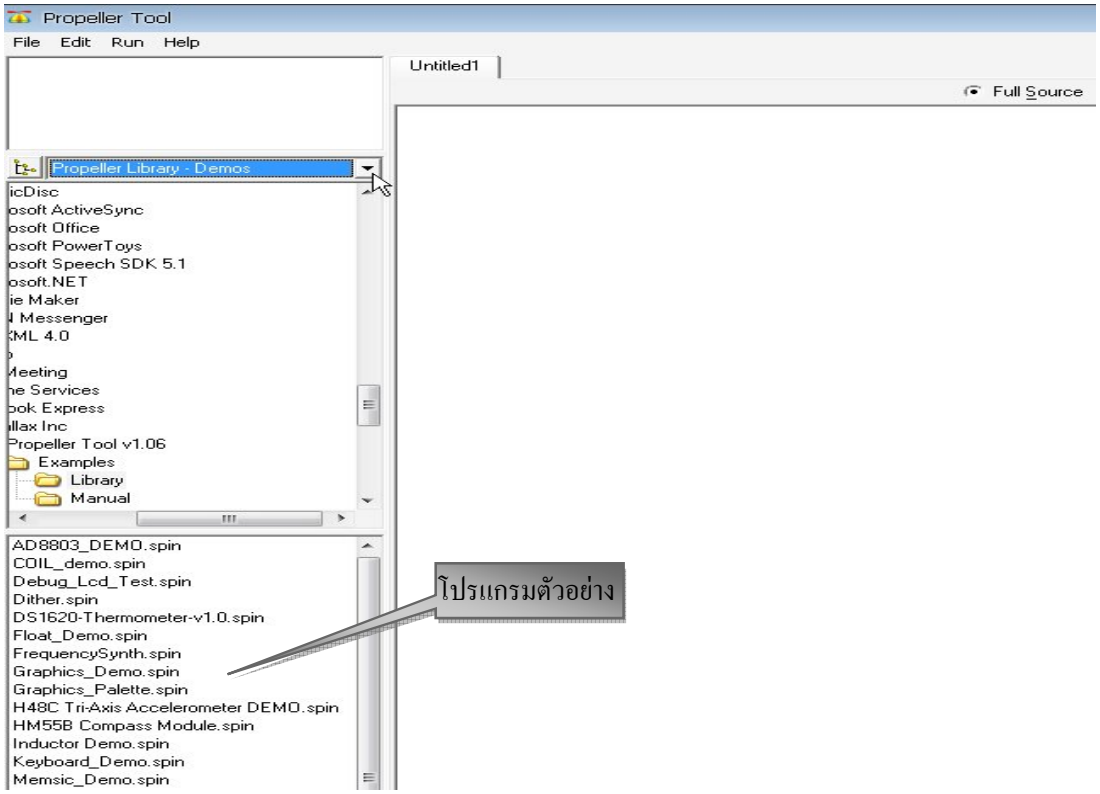
6. การทดสอบ RUN ตัวอย่างโปรแกรม

1) ทำการติดตั้งโปรแกรม Propeller Tool V1.06.exe ให้เรียบร้อย และทำการ Copy ตัวอย่างของ ETT จาก CD ROM มาวางไว้ใน PC ให้เรียบร้อย
2) ต่อสาย Download จาก Com Port ของ PC มาเข้าที่ขั้วต่อ Download ของบอร์ด และ Set jumper Download (หมายเลข9) มาทางด้าน Enable
3) ในที่นี้จะขอทดสอบการแสดงผลด้วย TV ดังนั้น ให้ต่อสายสัญญาณจาก TV OUT ของบอร์ดไปเข้าที่ขั้วต่อ VIDEO IN ของ TV แล้วทำการเปิด TV ในช่อง AV รอไว้ จากนั้นจ่ายไฟเลี้ยงให้กับตัวบอร์ด
4) จากตัวอย่างของ ETT ให้เข้าไปที่ Folder “ Ex1_TV ” จากนั้นเลือกที่ Folder “Ex1.2_TvText _Graphic” แล้วดับเบิ้ลคลิกที่ไฟล์ “Ex1.2_TV_TextGPH.spin” โปรแกรม Propeller ก็จะทำการเปิดไฟล์นี้ขึ้นมา
5) จากนั้นให้ไปที่เมนู RUN แล้วเลือกที่ Compile Current แล้วคลิกเลือก Load RAM หรือ Load EEPROM ก็ได้ โปรแกรมก็จะถูก Compile และ Download ลงบน MCU
6) หลังจาก โหลดโปรแกรมเสร็จให้สังเกตที่หน้าจอ TV จะเห็น ข้อความ “Hello!” แสดงขึ้นที่หน้าจอ
7) ถ้าต้องการทดสอบตัวอย่างอื่นๆอีกก็ให้ทำตามขั้นตอนที่กล่าวไปข้างต้น พร้อมกับหาอุปกรณ์มาต่อเข้ากับ Port ที่จะใช้ทดสอบด้วย

นอกจากตัวอย่างของ ETT ที่เขียนมาให้แล้ว ผู้ใช้ยังสามารถทดสอบ RUN ตัวอย่างที่มีอยู่ในโปรแกรม Propeller บนบอร์ดของ ETT ได้ด้วย โดยทำดังนี้

1) เปิดโปรแกรม Propeller ขึ้นมา ในช่องด้านซ้ายมือ ดูในรูปที่6.1 ในตำแหน่งที่ Mouse ชี้ ให้เลือกที่ Propeller Library - Demos จากนั้นจะเห็นไฟล์ตัวอย่างที่เป็นนามสกุลจุด SPIN แสดงอยู่ในช่องด้านล่างซ้ายมือ
2) ดับเบิลคลิกไฟล์ตัวอย่างที่ต้องการจะ RUN ขึ้นมา โดยควรจะเลือกไฟล์ที่สนับสนุนการ Interface ที่มีอยู่บนบอร์ดของอีทีที ที่จัดไว้ให้ ตัวอย่างไฟล์ที่ Interface ทาง Port VGA เช่น VGA_DEMO.spin , ตัวอย่างไฟล์ที่ Interface ทาง Port Mic , Audio เช่น Microphone_to_Headphones.spin , SingingDemoSeven.spin เป็นต้น
3) ไปที่เมนู RUN แล้วเลือกที่ Compile Current แล้วคลิกเลือก Load RAM หรือ Load EEPROM ก็ได้ โปรแกรมก็จะถูก Compile และ Download ลงบน MCU
4) สังเกตการณ์ทำงานของโปรแกรม จากอุปกรณ์ที่นำมาต่อ Interface

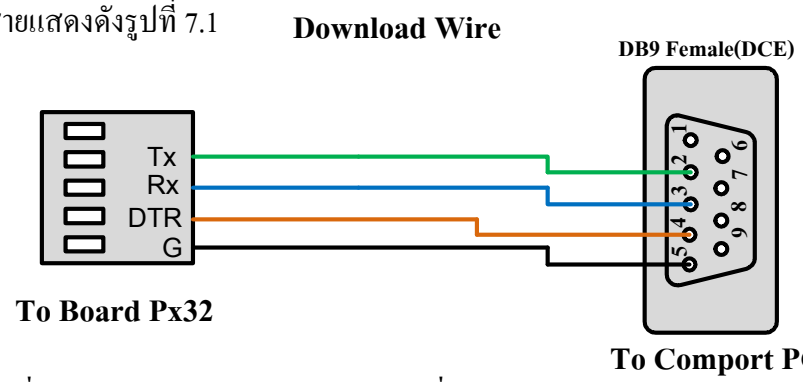
ในกรณีที่ผู้ใช้เปิดโปรแกรมขึ้นมาซ้อนกันหลายๆหน้าต่าง ตัวโปรแกรม Propeller จะทำการ Compile และ Download โปรแกรม เฉพาะในส่วนของไฟล์ที่ถูก Active แสดงให้เห็นอยู่ที่หน้าจอเท่านั้น



รูปที่6.1

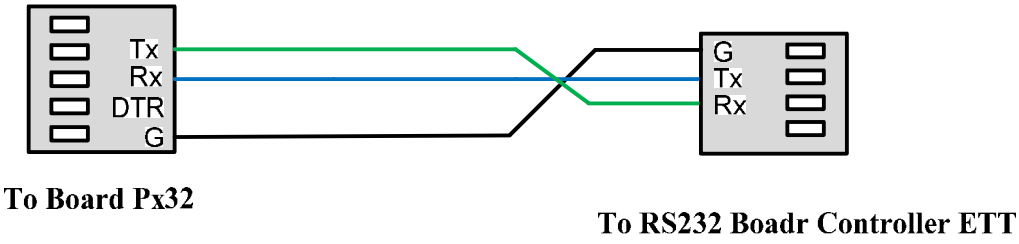
7. การต่อสาย Download และการต่อสายสื่อสาร RS232

-สำหรับสาย Download Program และสายสำหรับรับ-ส่งข้อมูลทาง RS232 ระหว่างบอร์ด PX32 กับ PC จะใช้สายเส้นเดียวกันโดยการเข้าสายแสดงดังรูปที่ 7.1



รูปที่ 7.1 แสดงการเข้าสาย Download & สายสื่อสาร RS232 ระหว่างบอร์ด Px32 และ PC

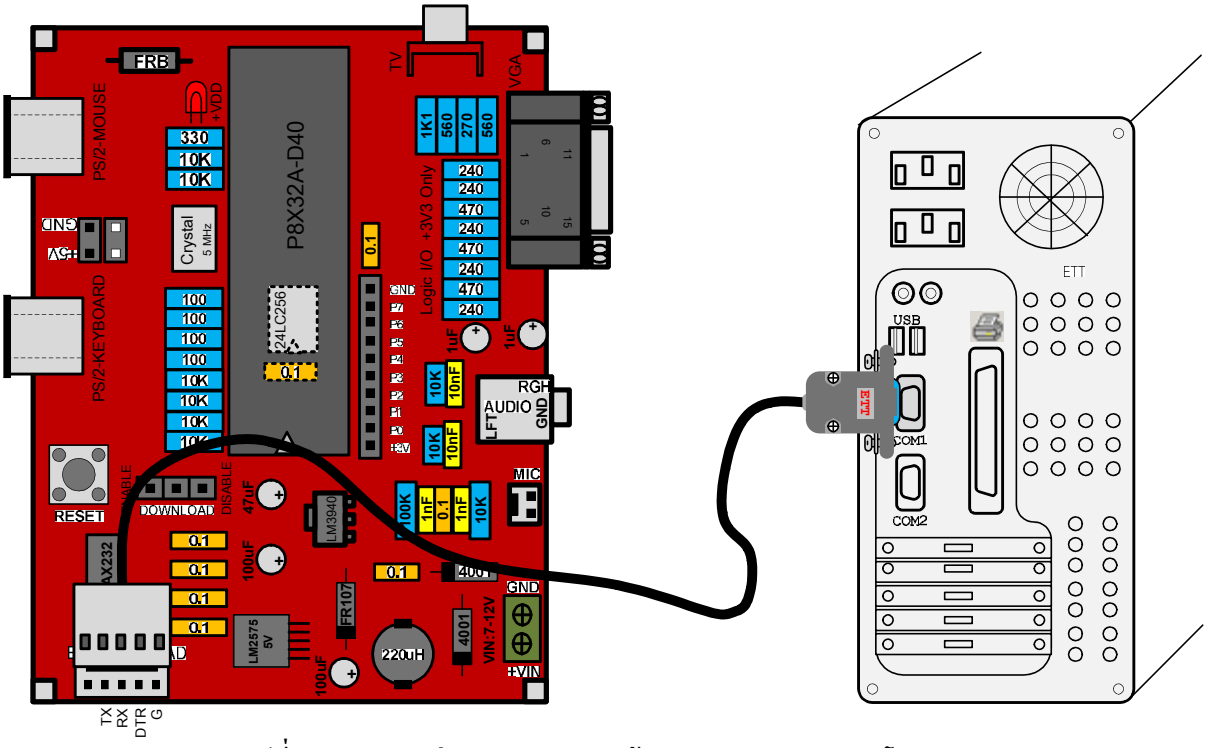
ในกรณีที่จะ รับ-ส่ง ข้อมูลทาง RS232 ระหว่างบอร์ด PX32 กับบอร์ด Controller อื่นๆ สามารถใช้การเข้าสายดังแสดงในรูปที่ 7.2



รูปที่ 7.2 แสดงการเข้าสาย RS232 สำหรับ รับ-ส่ง ข้อมูล ระหว่างบอร์ด PX32 กับบอร์ด Controller อื่นๆ

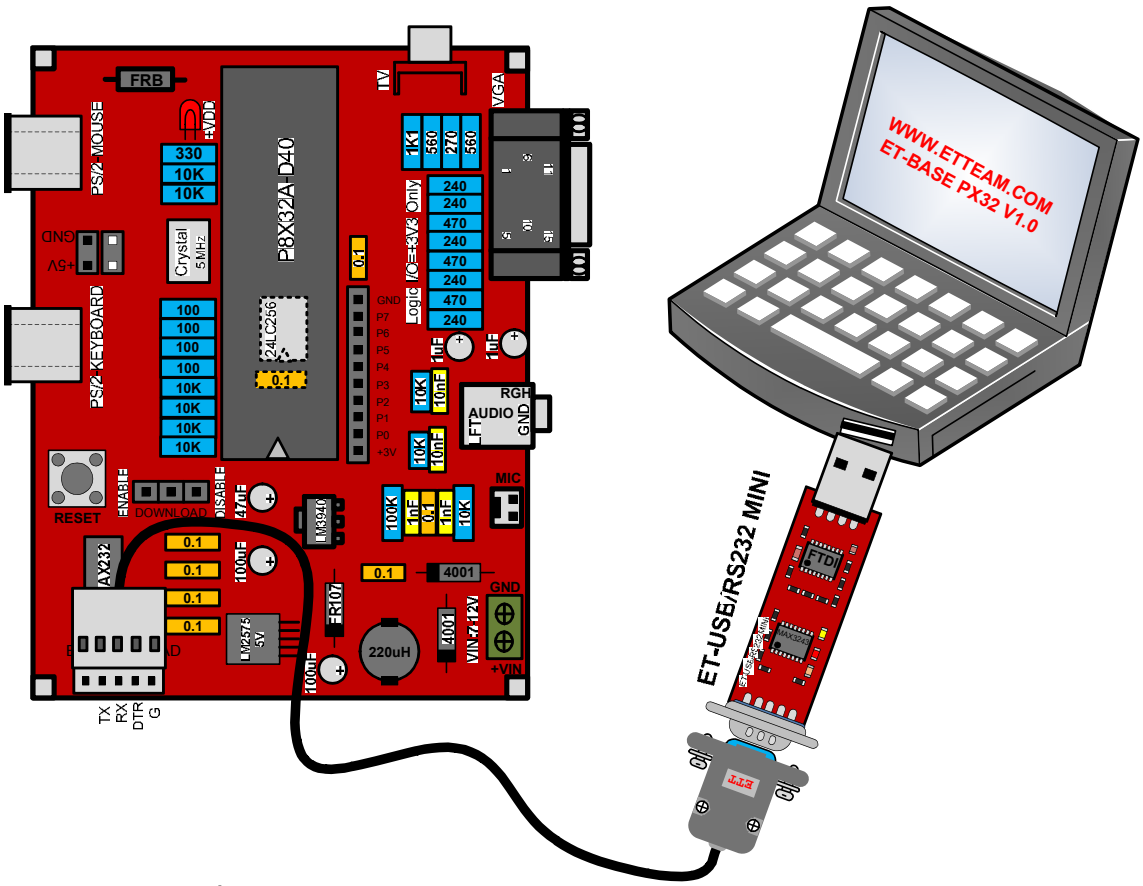
-เมื่อจะทำการ Download Program ลงบนบอร์ด Px32 ก็ให้ทำการต่อสาย download ทางด้าน Connector 5 Pin เข้ากับขั้วต่อของบอร์ด PX32 ส่วนทางด้าน Connector DB9 ก็ให้ต่อเข้ากับ Com Port ของ PC ที่ใช้งานดังรูปที่ 7.3 และทำการ Set Jumper Download(หมายเลข9) บนบอร์ด PX32 มาทางด้าน Enable , เมื่อจะทำการรับ-ส่ง ข้อมูล ผ่านทาง RS232 ของบอร์ด

PX32 กับ PC หรือกับบอร์ด Controller อื่นๆ หลังจาก Download โปรแกรมเรียบร้อยแล้ว ก็ควรจะ Set Jumper Download(หมายเลข9) บนบอร์ด PX32 มาทางด้าน Disable



รูปที่ 7.3 แสดงการต่อสาย Download เข้ากับ Com-Port ของ PC โดยตรง

ในกรณีที่ใช้เครื่อง PC หรือ Note Book ที่ไม่มีขั้วต่อ Com-Port ผู้ใช้จะต้องใช้ชุดแปลงจาก USB ไปเป็น RS232 เสียก่อน โดยถ้าใช้ชุดแปลงของ ETT รุ่น ET-USB/RS232 Mini ก็สามารถต่อใช้งานดังแสดงในรูปที่ 7.4



รูปที่ 7.4 แสดงการต่อสาย Download โดยผ่านตัวแปลงชุด ET-USB/RS232 Mini

8. ข้อกำหนดที่ควรรู้ในการพัฒนาโปรแกรมด้วยภาษา SPIN บน Tool Propeller

8.1) สำหรับภาษา SPIN นี้เวลาใช้งานเราจะมองออกเป็น Block ซึ่งมีอยู่ด้วยกันทั้งหมด 6 Block โดยเวลาจะเขียนก็จะต้องพิมพ์ชื่อเฉพาะของแต่ละ Block ที่ถูกกำหนดไว้ตายตัวด้วย โดยเวลาพิมพ์ชื่อเฉพาะของแต่ละ Block นี้ มีข้อกำหนดว่าจะต้องเริ่มพิมพ์ที่ Colum ซ้ายสุดของหน้าต่างที่จะใช้เขียนโปรแกรมเสมอ โดยชื่อ Block ทั้ง 6 Block มีดังนี้

1.) **CON** - Block นี้จะใช้สำหรับกำหนดค่าคงที่ให้กับตัวแปรที่จะใช้งาน ซึ่งตัวแปรที่ประกาศใน Block นี้สามารถเรียกใช้ได้ภายในไฟล์เดียวกัน ตัวอย่างเช่น

```
CON

        _clkmode        = xtal1 + pll16x
        _xinfreq        = 5_000_000
              tt        = 20
```

2.) **VAR** - Block นี้จะใช้สำหรับประกาศตัวแปรที่จะใช้ในการเขียนโปรแกรม โดยตัวแปรที่ประกาศใน Block นี้สามารถเรียกใช้งานได้ภายใน Project File เดียวกัน ตัวอย่างเช่น

```
VAR
        long    testerror
        long    vlong
        word    vword
        byte    vbyte
```

3.) **OBJ** - Block นี้จะใช้สำหรับกำหนดชื่อไฟล์จากภายนอกเข้ามา เพื่อให้สามารถเรียกใช้ฟังก์ชันที่มีอยู่ในไฟล์จากภายนอกได้ (ไฟล์จากภายนอกที่ไม่ใช่ไฟล์ Library ของตัว Propeller เวลาจะเรียกใช้ฟังก์ชันที่อยู่ในไฟล์นั้นๆ ผู้ใช้จะต้อง Copy File นั้นมาไว้ที่เดียวกับไฟล์ Project ที่ผู้ใช้เขียนอยู่ด้วย) การดูฟังก์ชันที่อยู่ภายใน ไฟล์ Library ของ Propeller ทำได้โดย ดูรูปที่ 6.1 ในช่องที่ Mouse ชี้อยู่ให้เลือก “Propeller Library” จากนั้นในหน้าต่างด้านล่าง ก็จะแสดงชื่อไฟล์ .spin ออกมาให้เห็นซึ่งไฟล์เหล่านี้ก็คือ File Library ของตัว Propeller ที่มี ไว้ให้เรียกใช้ เมื่อดับเบิลคลิกที่ File Library ไฟล์ก็จะถูกเปิดขึ้นมา และจะมีฟังก์ชั่นต่างๆถูกเขียนไว้ ซึ่งฟังก์ชั่นเหล่านั้นผู้ใช้ สามารถเรียกมาใช้ในโปรแกรมที่ผู้ใช้เขียนอยู่ได้ ตัวอย่างเช่น

```
OBJ

        term    : "tv_terminal"
```

เวลาจะเรียกใช้งานฟังก์ชันที่อยู่ในไฟล์ Library “ tv_terminal” ก็ให้แทนด้วย term.*ชื่อฟังก์ชัน* เช่น term.out(12) เป็นต้น

4.) **PUB** - สำหรับ Block นี้จะมีไว้สำหรับให้ผู้ใช้เขียนโปรแกรมที่ต้องการลงไป และสามารถเรียกใช้งานฟังก์ชันที่เขียนอยู่ใน block นี้ ได้ทั้งภายใน Project File เดียวกัน หรือต่าง Project File กันก็ได้ ในทุกๆ Project File ที่สร้างขึ้น จะต้องมี Block PUB อยู่อย่างน้อย 1 Block โดย Block PUB Block แรก ตัว Propeller จะถือเป็น Block

main ของโปรแกรม ส่วน Block PUB ที่อยู่ต่อจาก Block PUB แรกจะถูกมองเป็นเพียงโปรแกรมย่อยมีไว้สำหรับให้โปรแกรมหลักเรียกใช้ โดยใน Block นี้ เมื่อพิมพ์คำว่า PUB แล้ว จะต้องตั้งชื่อให้กับ Block PUB ด้วยซึ่งชื่อนี้ก็จะเป็นเหมือนชื่อของFunction นั่นเอง ตัวอย่างเช่น

```
PUB stop

  vga.stop
```

เวลาจะประกาศตัวแปรใช้ภายใน PUB นั้นๆ ก็ให้ใส่ | ต่อจากชื่อ แล้วจึงตามด้วยชื่อตัวแปร ซึ่งตัวแปรที่ ประกาศนี้จะใช้งานได้เฉพาะภายใน PUB เท่านั้น และตัวแปรที่ประกาศจะมีขนาด 32 bit

เวลาจะรับค่าจากภายนอกเข้ามาใช้ยังฟังก์ชัน ก็ให้ใส่ **(ชื่อตัวแปร1,ตัวแปร2)** ต่อจากชื่อ โดยภายในวงเล็บก็ให้ใส่ชื่อตัวแปรที่ใช้รับการส่งผ่านค่าเข้ามาในฟังก์ชันด้วย

เวลาจะส่งผ่านค่าจากฟังก์ชันออกไปยังภายนอก ก็ให้ใส่ **:** ต่อจากชื่อ แล้วตามด้วยชื่อตัวแปรที่จะส่งผ่านค่าออกไป ดังตัวอย่าง

```
PUB Ticks(Pin) : Microseconds | cnt1, cnt2

  outa[Pin]~
  dira[Pin]~~
  outa[Pin]~~
  outa[Pin]~
  dira[Pin]~

  waitpne(0, |< Pin, 0)
  cnt1 := cnt
  waitpeq(0, |< Pin, 0)
  cnt2 := cnt
  Microseconds := (||(cnt1 - cnt2) / (clkfreq / 1_000_000)) >> 1
```

ในตัวอย่างนี้ชื่อ PUB คือ Ticks ส่วน (Pin) เป็นการประกาศตัวแปรเพื่อใช้รับค่าจากภายนอกเข้ามาเก็บไว้ที่ตัว แปร Pin ส่วน : Microsecond เป็นการประกาศตัวแปรเพื่อใช้ส่งผ่านค่าออกจากฟังก์ชัน และสุดท้าย | cnt1,cnt2 เป็นการประกาศตัวแปรไว้ใช้งานภายใน Block PUB Ticks เวลาใช้งานจริงก็ประกาศในส่วนที่จะใช้งานก็พอ ไม่จำเป็นต้องประกาศจนครบทั้งหมดก็ได้

5.) **PRI** -สำหรับ Block นี้จะมีไว้สำหรับให้ผู้ใช้เขียนโปรแกรมที่ต้องการลงไปเหมือนกับ Block PUB แต่จะต่างตรงที่ Function ที่เขียนอยู่ใน Block นี้จะถูกมองเป็นโปรแกรมย่อยสำหรับให้เรียกใช้ภายใน Project File เดียวกันเท่านั้น ไม่สามารถเรียกใช้งานคนละ Project File ได้ ส่วนการประกาศใช้งานตัวแปรก็จะเหมือนกับ Block PUB

```
PRI bound(i, delta) : b | d

  d := bx_div[i]
  b := bx_min[i] + (bx_acc[i] := bx_acc[i] + delta
```

6.) **DAT** - สำหรับ Block นี้จะมีไว้สำหรับ กำหนด data table หรือ สำหรับ เขียน Assembly Code ตัวอย่างเช่น

```
DAT

'*********************************
'* Assembly language VGA driver *
'*********************************

                        org

entry                   mov     taskptr,#tasks

                        mov     x,#8
:init                   jmpret  taskret,taskptr
                        djnz    x,#:init


vgacolors               long    $C000C000       'red
                        long    $C0C00000
                        long    $08A808A8       'green
                        long    $0808A8A8
                        long    $50005000       'blue
                        long    $50500000
```

Block ทั้ง 6 นี้เวลาจะใช้งานผู้ใช้สามารถประกาศใช้เฉพาะ Block ที่จะใช้งานเท่านั้นก็ได้ไม่จำเป็นต้องประกาศทุก Block และในแต่ละ Block สามารถประกาศใช้ซ้ำกันได้มากกว่า 1 Block โดยใน Block PUB เวลาประกาศใช้ซ้ำกัน ชื่อของฟังก์ชันที่พิมพ์ต่อจาก PUB จะต้องไม่เหมือนกัน

8.2) ตัวอักษรที่พิมพ์ไม่ว่าจะเป็นการพิมพ์คำสั่ง หรือ การพิมพ์ชื่อตัวแปรก็ตาม สามารถพิมพ์โดยใช้ ตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก แทนกันได้ โดยตัว Propeller จะมองเป็นตัวแปรตัวเดียวกัน จะไม่มองแยกกันคนละตัวเหมือนภาษา C

8.3) เวลาแทนค่าเลข ฐาน10 เกิน 3 หลักจะต้องคั่นด้วย _ เสมอ เช่น x := 1_200

8.4) การใช้เครื่องหมายเท่ากับ ถ้าใช้ใน Block CON จะใช้ = ,ถ้าใช้ใน Block OBJ จะใช้ : ,ถ้าใช้ใน Block PUB จะใช้ := ,

8.5) การใช้เลขฐานใน Propeller : $ นำหน้าค่าคงที่ เช่น $56 จะใช้แสดงถึงตัวเลขฐาน 16 , % นำหน้าค่าคงที่(1และ0) เช่น %1001 จะแสดงถึงตัวเลขฐาน2 , %% นำหน้าค่าคงที่ (0,1,2,3) เช่น %%2130 จะมองเป็น Quaternary Number ก็คือให้มองเลข 1หลัก มีขนาด 2 บิต จากตัวอย่าง %%2130 = 10 01 11 00 (ฐาน2) = 9C (ฐาน16)

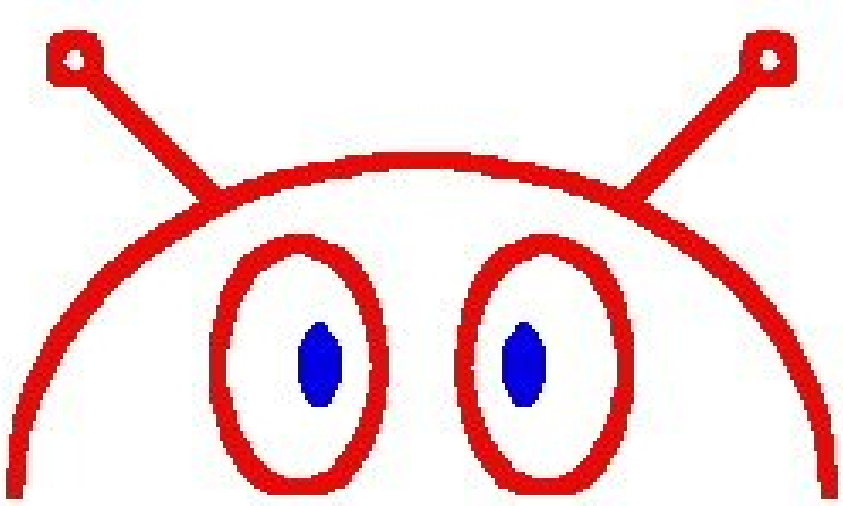
8.6) ในการใช้งานคำสั่งเกี่ยวกับ Loop เช่น repeat หรือ คำสั่งเงื่อนไข เช่น if หลังจากพิมพ์คำสั่งแล้ว ส่วนของโปรแกรมที่จะพิมพ์ในบรรทัดต่อมาเพื่อให้ทำงานภายใต้คำสั่ง repeat หรือ if จะต้องพิมพ์ให้เยื้องกับคำสั่งเหล่านี้มาทางขวามืออย่างน้อย 1 ช่องว่าง ถ้าพิมพ์ให้อยู่ในระดับเดียวกัน หรือเยื้องไปทางด้านซ้ายมือ ตัว Propeller จะถือว่าโปรแกรมในบรรทัดนั้นอยู่ภายนอกคำสั่ง repeat หรือ if เป็นต้น ตัวอย่างเช่น

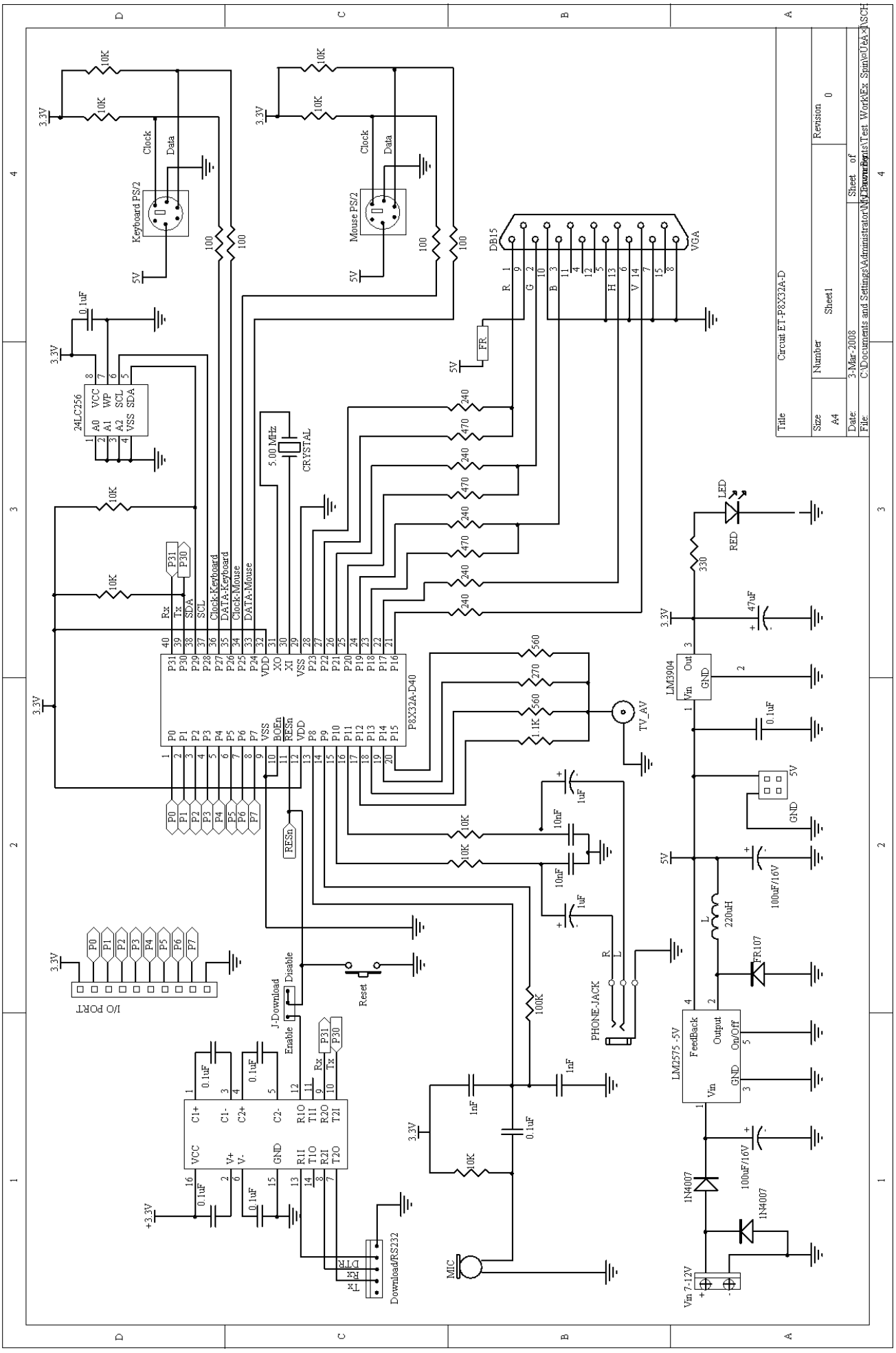
```
PRI BlinkingLED
  Pin := 07
  DirA[Pin] := Out
  Repeat
    OutA[Pin] := High            'LED ON
    WaitCnt(40_000_000 + Cnt)  'ONE-HALF SECOND WAIT
    OutA[Pin] := Low             'LED OFF
    WaitCnt(40_000_000 + Cnt)  'ONE-HALF SECOND WAIT
```

8.7) ในการเขียน Comment อธิบายโปรแกรมจะนำด้วยสัญลักษณ์ ' หรือ '' แล้วตามด้วย Comment หรือเขียนไว้ใน เครื่องหมายปีกกา {comment} หรือ {{comment}}

8.8) ใน Block PUB ที่ใช้เป็น Block main ของโปรแกรม ถ้าโปรแกรมที่เขียนใน Block เป็นแบบ Loop เปิด ไม่มีการทำงานวนซ้ำภายใน Loop ใดๆ ก็ควรจะจบโปรแกรมด้วยคำสั่ง Repeat เพื่อไม่ให้ MCU วนกลับไป Reset เริ่มโปรแกรมใหม่

สำหรับรายละเอียดมากกว่านี้ในการใช้งานคำสั่งต่างๆของ ภาษา SPIN ให้ดูได้ที่ Help ของ โปรแกรม Propeller โดยเข้าไปที่เมนู Help แล้วเลือกที่ Propeller Manual(pdf)





รูปวงจร ET-BASE PX32 V1.0