ภาคผนวก ก คู่มือบอร์ดไมโครคอนโทรลเลอร์รุ่น ET-ARM7 STAMP LPC2119

(ที่มา : http://www.etteam.com/product/ARM/et-arm_stamp_lpc2119.htm)

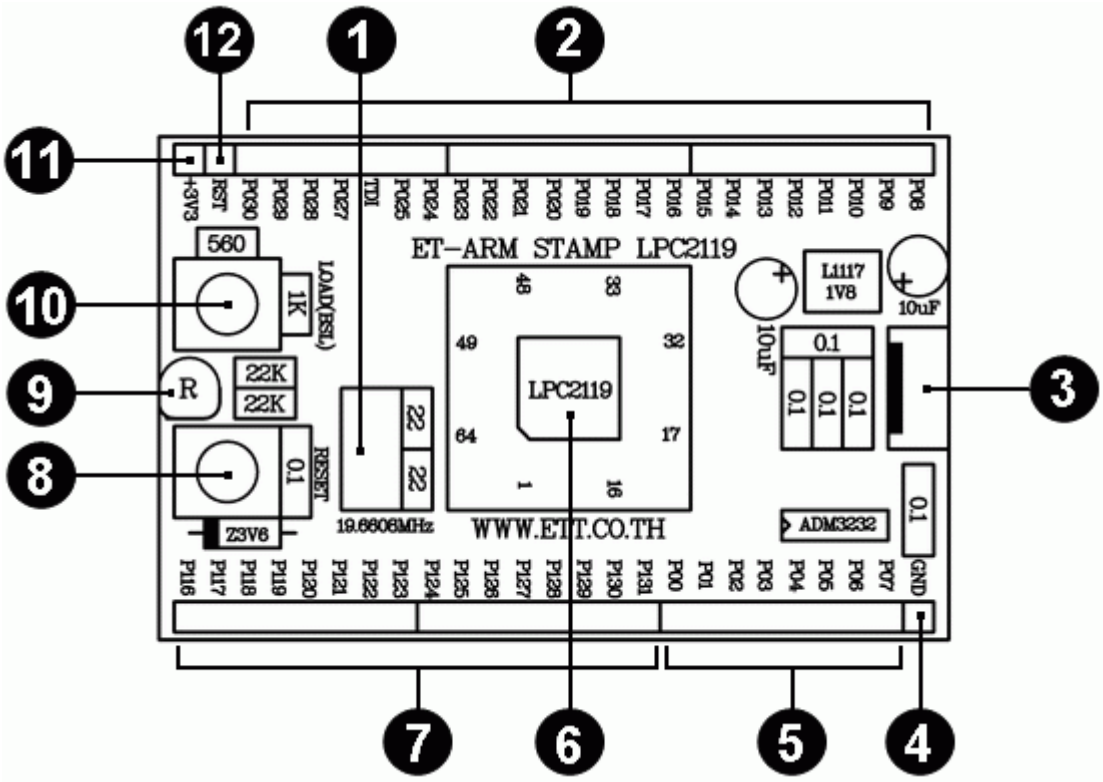
ET-ARM7 STAMP LPC2119

ET-ARM7 STAMP LPC2119 เป็นบอร์ดไมโครคอนโทรเลอร์ในตระกูล ARM7TDMI-S Core ซึ่งเลือกใช้ไมโครคอนโทรเลอร์ 16/32-Bit ขนาด 64 Pin แบบใช้พลังงานต่ำเป็น MCU ประจำบอร์ด ซึ่งบอร์ดนี้เลือกใช้ MCU เบอร์ LPC2119 ของ Philips โดยการออกแบบโครงสร้างของบอร์ดนั้นจะเน้นเรื่องการจัดวางบอร์ดให้มีขนาดเล็กเพื่อให้ง่ายต่อการนำไปประยุกต์ใช้งาน โดยได้นำ MCU มาจัดวงจรร่วมกับอุปกรณ์พื้นฐานที่จำเป็นและจัดขาออกมาให้ใช้งานภายนอก ซึ่งการจัดเรียงขาสัญญาณจะทำการจัดเรียงอย่างเป็นระเบียบเพื่อให้สามารถต่อใช้งานได้โดยสะดวก ตัวบอร์ดใช้ไฟ +3.3V สามารถรองรับ I/O ที่เป็นสัญญาณ 5V ได้ ตัวบอร์ดมี Connector UART0 (RS-232) จำนวน 1 Port สำหรับทำการ Download Hex File หรือใช้งานในการสื่อสาร RS232 ในโปรแกรม Application ที่เขียนขึ้นเอง

คุณสมบัติของบอร์ด

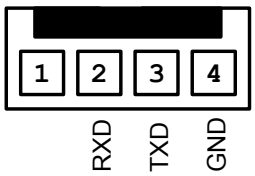
1. ใช้ MCU ตระกูล ARM7TDMI-S เบอร์ LPC2119 ของ Philips ซึ่งเป็น MCU ขนาด 16/32-Bit
2. ใช้ Crystal 19.6608 MHz โดย MCU สามารถประมวลผลด้วยความเร็วสูงสุดที่ 58.9824 MHz เมื่อใช้งานร่วมกับ Phase-Locked Loop (PLL) ภายในตัว MCU เอง
3. รองรับการโปรแกรมแบบ In-System Programming (ISP) และ In-Application Programming (IAP) ผ่านทาง On-Chip Boot-Loader Software ผ่านทาง UART0 (RS232)
4. Power Supply ใช้แรงดันไฟฟ้า +3.3V เท่านั้น (3.0V – 3.6V $\pm$ 10% Error)
5. ภายใน MCU มีหน่วยความจำโปรแกรมแบบ Flash ขนาด 128 KB, หน่วยความจำข้อมูล Static RAM ขนาด 16 KB
6. จำนวน GPIO สูงสุดถึง 46 I/O Pins สามารถเชื่อมต่อกับระบบ I/O ที่เป็นสัญญาณ 5V ได้ ซึ่งขาสัญญาณ GPIO จะมีการใช้งานร่วมกันของ Function อื่นๆอีกดังนี้
   - SPI จำนวน 2 ช่อง ,I2C 1 ช่อง , CAN จำนวน 2 ช่อง , 4-Channel 10 Bit A/D Converter
   - UART แบบ Full-Duplex จำนวน 2 ช่อง คือ UART 0 มาตรฐาน 4 Pin ETT เป็นสัญญาณระดับ RS232 Level และ UART 1 เป็นสัญญาณระดับ TTL Level
   - Timer 32-bit จำนวน 2 ช่อง (4 Input Capture / 4 Output Compare), PWM Output ,Watchdog Timer ,Real Time Clock
7. ทนอุณหภูมิใช้งานระหว่าง -40 to +85°C
8. Dimensions : - PCB Size 1575 x 2559 mil (~ 40 x 65 mm)
   - ระยะขา ความกว้าง 1500 mil ความยาว 2500 mil (~ 38.1 x 63.5 mm)
   - ระยะระหว่างขา 2 x 25 Pins I/O Connector 100 mil (~ 2.54 mm)

โครงสร้างบอร์ด ET-ARM7 STAMP LPC2119



- **หมายเลข** 1 คือ Crystal 19.6608 MHz **หมายเลข** 6 คือ CPU ARM7 LPC2119 ของ Philips
- **หมายเลข 2 และ** 5 คือ GPIO 0 ตั้งแต่ P0.0 – P0.25 และ P0.27 – P0.30 จำนวนทั้งหมด 30 Pins สามารถรองรับอุปกรณ์ที่มีสัญญาณ I/O เป็น 3.3V และ 5V ได้
- **หมายเลข** 3 คือ คือ UART 0 หรือ Serial Port สำหรับติดต่อกับอุปกรณ์มาตรฐาน RS232 และเป็น ISP Download Connector สำหรับโปรแกรม Hex file ลงบอร์ด

ET-RS232



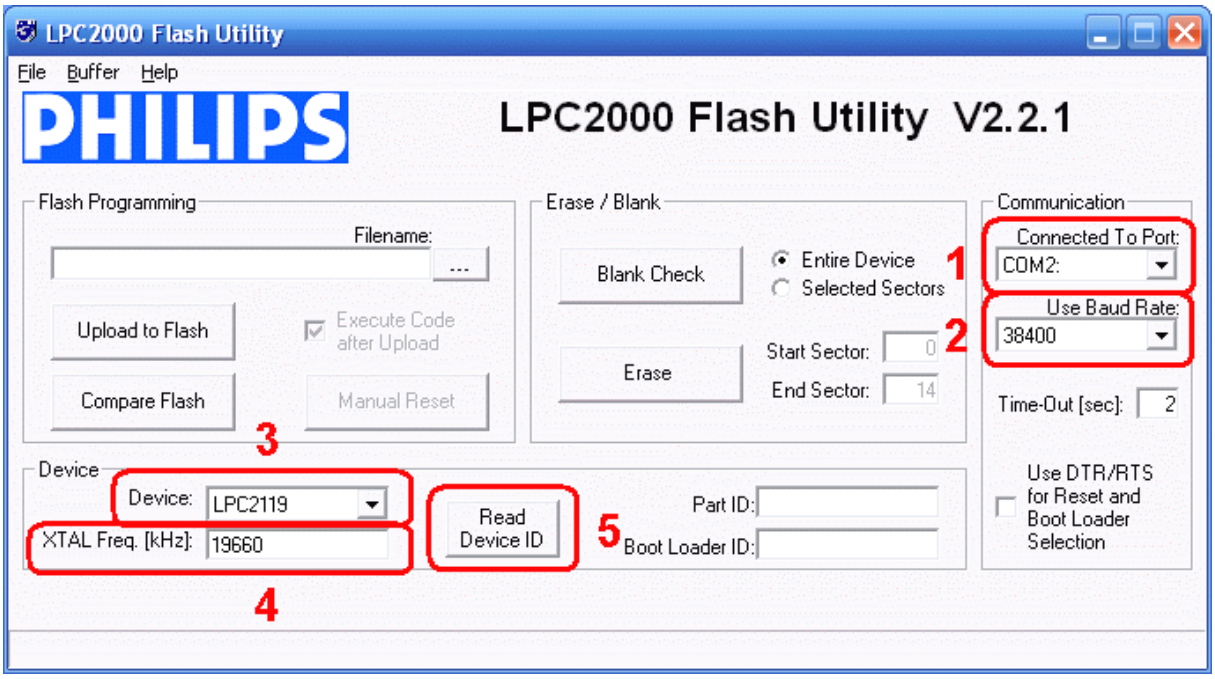
- **หมายเลข** 4 คือ จุดต่อ GND ส่วน **หมายเลข** 11 คือ จุดต่อ Power Supply +3.3V ของบอร์ด
- **หมายเลข** 7 คือ GPIO 1 ตั้งแต่ P1.16 – P1.31 จำนวนทั้งหมด 16 Pins สามารถรองรับอุปกรณ์ที่มีสัญญาณ I/O เป็น 3.3V และ 5V ได้
- **หมายเลข** 8 คือ สวิตช์ RESET ส่วน **หมายเลข** 10 คือ สวิตช์ LOAD (BSL)
- **หมายเลข** 9 คือ LED สีแดง แสดงสถานการณ์ทำงานของ Power Supply
- **หมายเลข** 12 **คือ** จุดต่อสัญญาณ RESET สำหรับ Reset อุปกรณ์ภายนอก

การ Download Hex file ให้กับ MCU ของบอร์ด

การ Download Hex File ให้กับหน่วยความจำ Flash ของ MCU ในบอร์ดนั้น จะใช้โปรแกรมชื่อ LPC2000 Flash Utility ของ Philips ซึ่งจะติดต่อกับ MCU ผ่าน Serial Port ของคอมพิวเตอร์ PC โดยโปรแกรมดังกล่าวสามารถดาวน์โหลดโปรแกรมได้ที่ www.semiconductors.philips.com

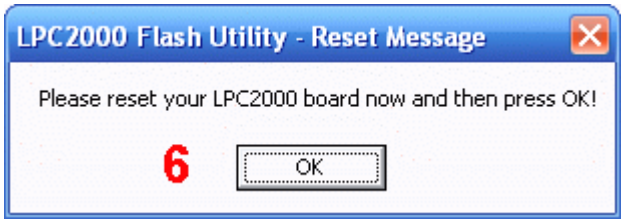
ขั้นตอนการ Download HEX File ให้กับ MCU

1. ต่อสายสัญญาณ RS232 ระหว่างพอร์ตสื่อสารอนุกรม RS232 ของ PC และบอร์ด (ET-RS232)
2. จ่ายไฟเลี้ยงวงจรขนาด +3.3V ให้กับบอร์ด ซึ่งจะสังเกตเห็น LED สีแดง (PWR) ติดสว่างให้เห็น
3. สั่ง Run โปรแกรม LPC2000 Flash Utility ของ Philips ซึ่งจะได้ผลดังรูป



4. เริ่มต้นกำหนดค่าตัวเลือกต่างๆให้กับโปรแกรมตามต้องการ ซึ่งในกรณีที่ใช้กับ LPC2119 ของบอร์ด ET-ARM STAMP LPC2119 ของอีทีที ให้เลือกกำหนดค่าต่างๆให้โปรแกรมดังนี้
   1) เลือก COM Port ให้ตรงกับหมายเลข Com Port ที่ใช้งานจริง (ในตัวอย่าง COM2)
   2) ตั้งค่า Baud Rate อยู่ที่ระหว่าง 4800 – 38400 ซึ่งเป็นค่าที่ทดสอบแล้วใช้ได้โดยไม่เกิดปัญหา หรือใช้ค่าความเร็วมาตรฐานคือ 9600
   3) เลือกกำหนดเบอร์ MCU ในการติดต่อ ในที่นี้คือ LPC2119
   4) กำหนดค่าคริสตอล ออสซิลเลเตอร์ ให้ตรงกับที่ใช้ในจริงภายในบอร์ด โดยกำหนดให้มีหน่วยเป็น KHz และห้ามใส่ค่าเกิน 5 หลัก ในที่นี้ใช้ค่า 19.6608MHz ซึ่งเท่ากับ 19660

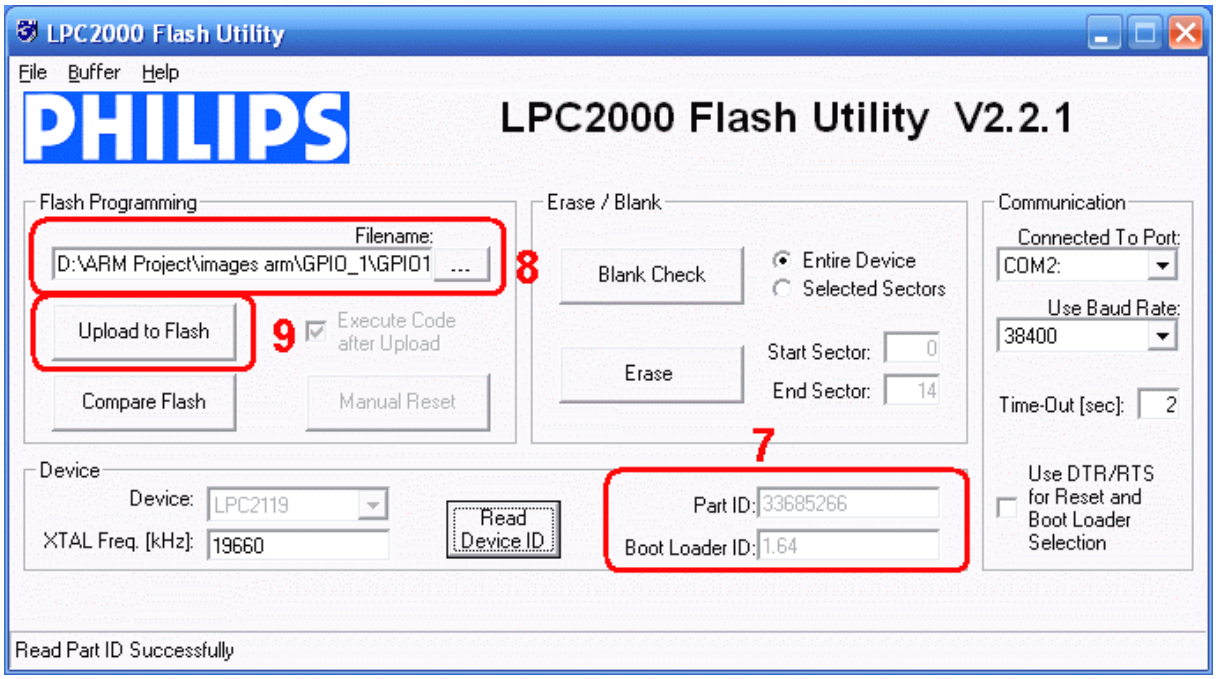
5) คลิกเมาส์ที่ปุ่มคำสั่ง Read Device ID เพื่อติดต่อกับ CPU ซึ่งจะมีข้อความขึ้นมาเตือนให้เข้าสู่ Boot Mode ดังแสดงในรูป



6) ให้กดสวิตช์ RESET และ LOAD (BSL) ที่บอร์ด ET-ARM STAMP LPC2119 เพื่อทำการ Reset ให้ MCU ทำงานใน Boot Loader ตามขั้นตอนดังต่อไปนี้

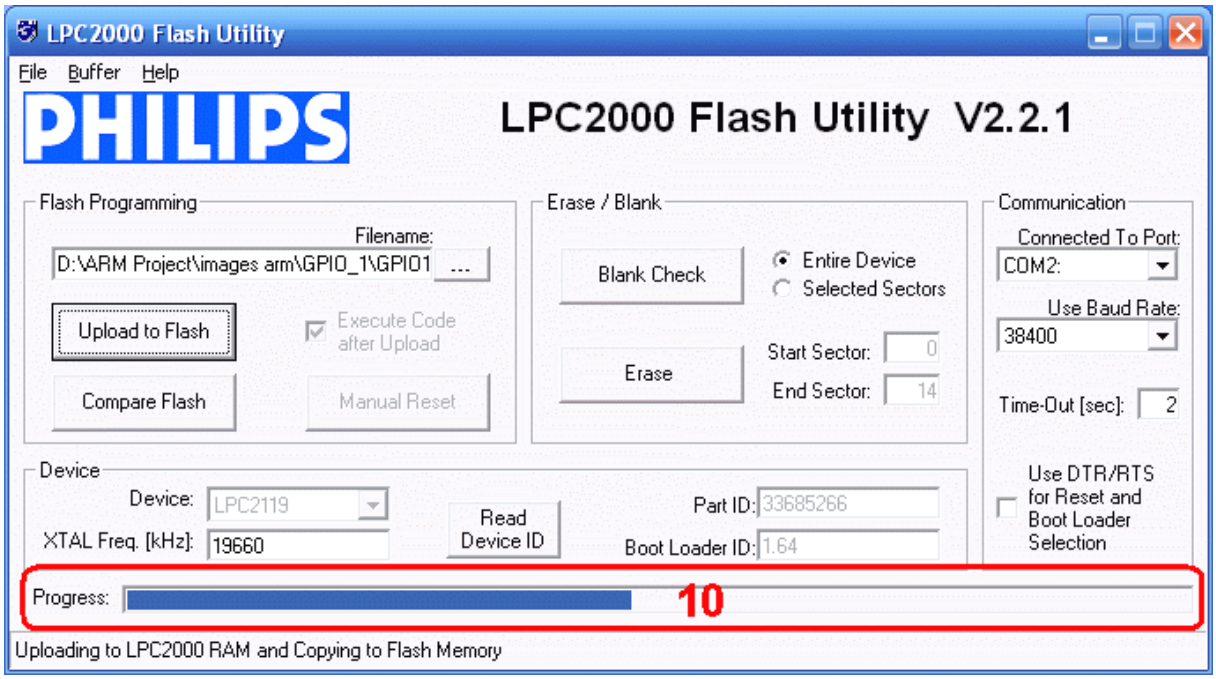
- กดสวิตช์ LOAD (BSL) ค้างไว้
- กดสวิตช์ RESET โดยที่สวิตช์ LOAD (BSL) ยังกดค้างอยู่
- ปล่อยสวิตช์ RESET โดยที่สวิตช์ LOAD (BSL) ยังกดค้างอยู่
- ปล่อยสวิตช์ LOAD (BSL) เป็นลำดับสุดท้าย เสร็จแล้วจึงคลิกเมาส์ที่ “OK.

7) เมื่อติดต่อกับ CPU ได้ จะปากฏรายละเอียด Part ID และ Boot Loader ID ดังรูป



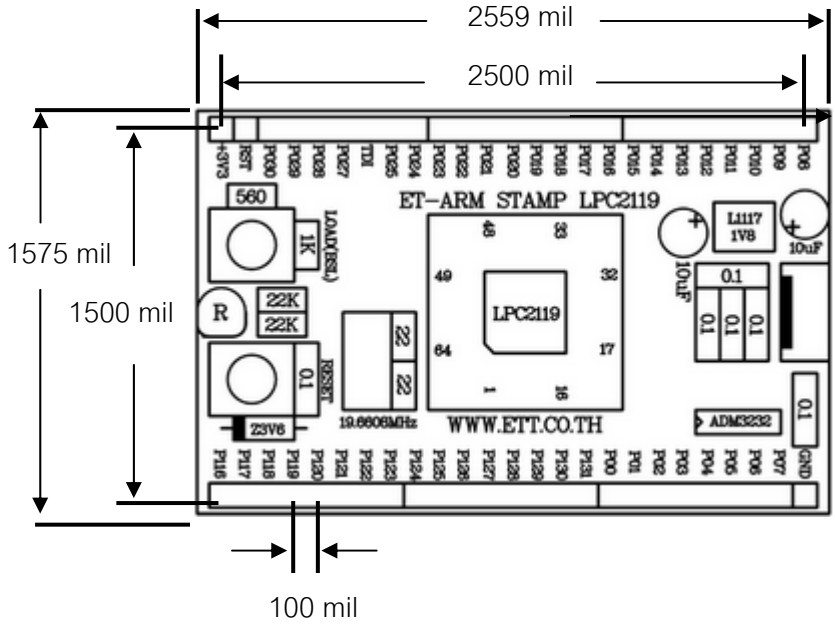
8) ให้ทำการเลือกกำหนด HEX File ที่จะทำการสั่งโปรแกรม

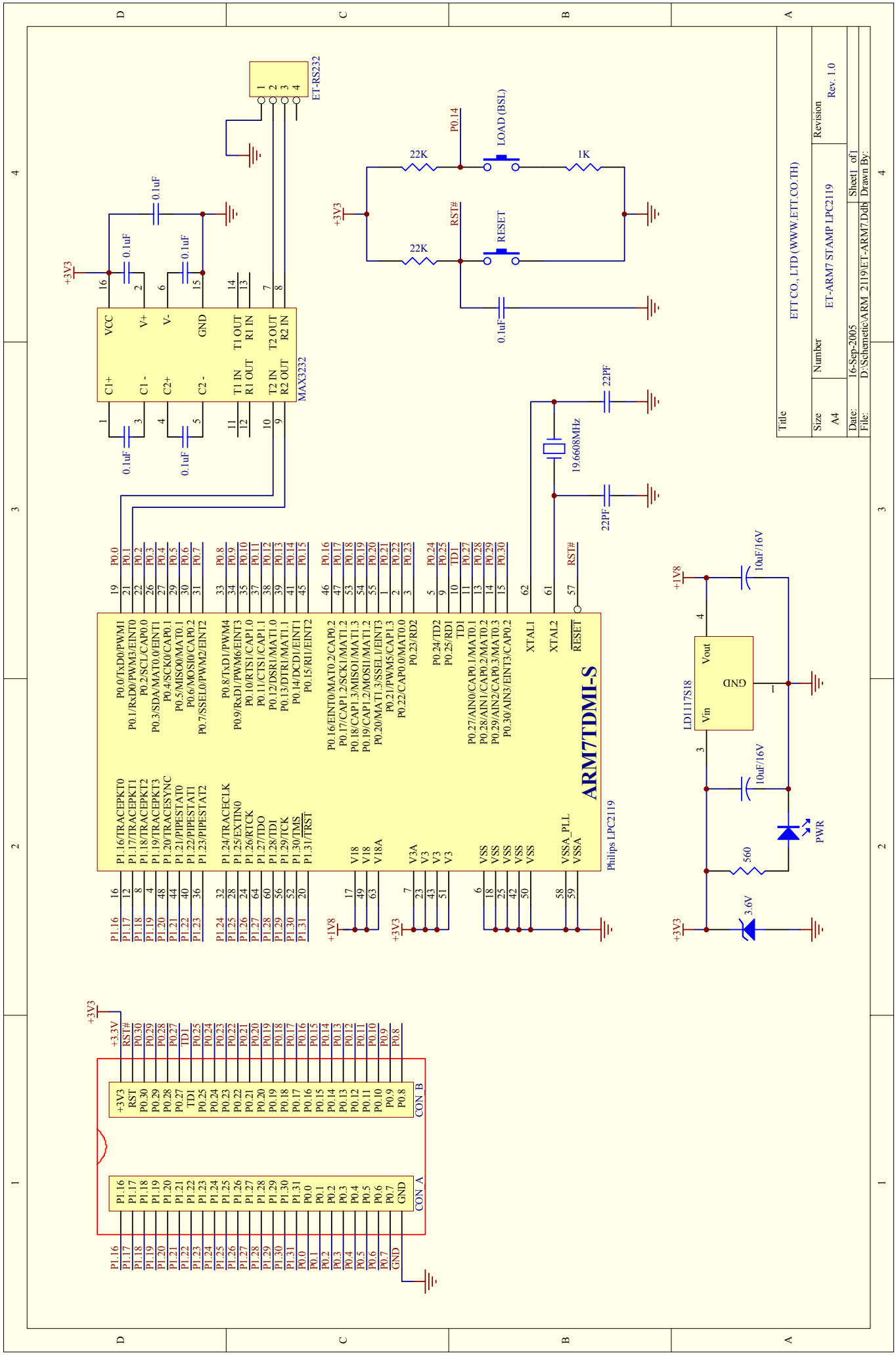
9) ให้ทำการคลิกเมาส์ที่ “Upload to Flash” ซึ่งจะเห็นว่าโปรแกรม LPC2000 จะเริ่มต้นทำการ Download ข้อมูลให้กับ MCU ทันที โดยในขั้นตอนนี้ให้รอจนกว่าการทำงานของโปรแกรมจะเสร็จสมบูรณ์ ดังรูป



10) เมื่อการทำงานของโปรแกรมเสร็จเรียบร้อยแล้ว ให้กดสวิตช์ Reset ที่บอร์ด ซึ่ง MCU จะเริ่มต้นทำงานตามโปรแกรมที่สั่ง Download ให้ทันที

Board Dimension




ARM7TDMI-S
Philips LPC2119
MAX3232
ET-RS232
LD1117S18
CON_A
CON_B
LOAD (BSL)
RESET
PWR
19.6608MHz
ETT CO., LTD (WWW.ETT.CO.TH)
ET-ARM7 STAMP LPC2119
Rev. 1.0
16-Sep-2005
D:\Schematic\ARM 2119\ET-ARM7.Ddb

ภาคผนวก ข คู่มือ ET-ARM7 START KIT V1

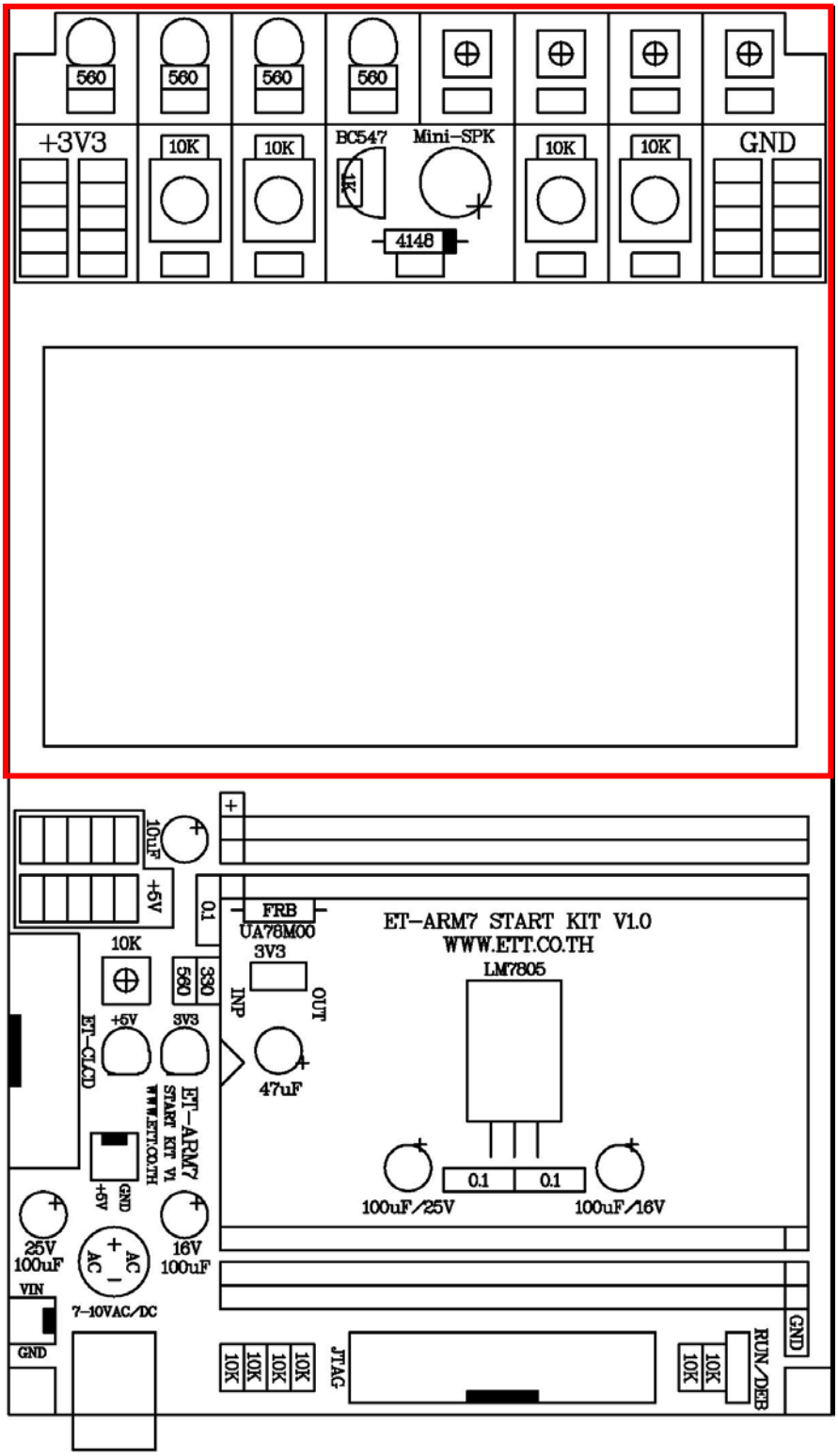
(ที่มา : http://www.etteam.com/product/ARM/et-arm7start_kit_v1exp.htm)

ET-ARM7 START KIT V1.0 / EXP

ET-ARM7 START KIT V1.0 / EXP เป็นชุด “ARM Base Socket” ใช้สำหรับสนับสนุนการใช้งานร่วมกับชุดโมดูล “ET ARM STAMP LPC2119” หรือโมดูล ARM อื่นๆที่มีขนาดเดียวกัน โดยในส่วนของชุด “ARM Base Socket” หรือ ET-ARM7 START KIT V1.0 และ ET-ARM7 START KEI V1.0 EXP จะประกอบไปด้วย วงจรพื้นฐานที่จำเป็นสำหรับการศึกษาเรียนรู้และทดลองใช้งานทรัพยากรต่างๆของ MCU ตระกูล ARM โดยภายในบอร์ดได้จัดเตรียมวงจรใช้งานที่จำเป็นไว้ให้ใช้งานอย่างครบถ้วนได้แก่

- วงจรแหล่งจ่ายไฟ แบบ Bridge Rectifier ขนาด 1A พร้อมวงจร Filter สามารถใช้กับแหล่งจ่ายไฟได้ทั้ง AC และ DC ขนาด 7-12V
- วงจร Regulate ขนาด +3.3V / 500mA สำหรับใช้งานเป็นแหล่งจ่ายไฟเลี้ยงวงจรให้กับโมดูล “ET-ARM STAMP LPC2119” และวงจร I/O ต่างๆที่ใช้กับแหล่งจ่ายขนาด 3.3V พร้อม LED แสดงสถานะสีเขียว และจุด Connector เชื่อมต่อใช้งาน ทั้งตัวผู้และตัวเมีย
- วงจร Regulate ขนาด +5V / 1A สำหรับใช้งานเป็นแหล่งจ่ายไฟเลี้ยงวงจรให้กับจอแสดงผล LCD และอุปกรณ์ I/O ต่างๆที่ใช้กับแหล่งจ่ายขนาดขนาด +5V พร้อม LED แสดงสถานะสีแดง และจุด Connector เชื่อมต่อใช้งาน ทั้งตัวผู้และตัวเมีย
- วงจรเชื่อมต่อจอแสดงผล LCD แบบ Character พร้อม VR ปรับความสว่าง โดยใช้สัญญาณ GPIO1[16..21] ในการเชื่อมต่อวงจรกับ LCD แบบ 4 Bit Interface
- ขั้วต่อ ARM JTAG ขนาด 20Pin (HEADER IDE 2 x 10) สำหรับใช้เชื่อมต่อกับชุด Hardware ARM JTAG ในการ Debug การทำงานของ MCU โดยจัดเรียงสัญญาณอ้างอิงตามมาตรฐานของ “Multi-ICE System” ของ ARM ซึ่งสามารถต่อใช้งานร่วมกับชุด Hardware Debugger ของ ARM ที่ออกแบบโดยอ้างอิงมาตรฐานเดียวกันนี้ได้ทั้งหมด
- วงจร LED แสดงผลแบบ Sink Current ใช้ไฟเลี้ยง 3.3V โดยใช้ LED สีแดงขนาด 3mm. จำนวน 4 ชุด สำหรับใช้ในการทดสอบการทำงานของ Output ต่างๆ
- วงจรปรับแรงดัน 0-3.3V โดยใช้ตัวต้านทานปรับค่าได้แบบเกือกม้าแบบมีแกนปรับ จำนวน 4 ชุด สำหรับใช้ในการทดสอบการทำงานของ A/D
- วงจร Push Button Switch จำนวน 4 ชุด สำหรับใช้ทดสอบการทำงานของ Input ต่างๆ
- วงจร Mini Speaker สำหรับใช้ทดสอบการกำเนิดเสียง Beep หรือเสียงอื่นๆ
- พื้นที่สำหรับบัดกรีวงจรเพิ่มเติมขนาด 8cm x 4.5cm หรือใช้เป็นพื้นที่ติดตั้ง Photo Board รุ่น AD100 ขนาด 360 จุด
- ขั้วต่อ Header สำหรับรองรับโมดูล “ET-ARM STAMP LPC2119” หรือโมดูลอื่นๆที่มีขนาดเท่ากันพร้อม Connector สำหรับต่อไปยังวงจรทดลองต่างๆทั้งแบบตัวผู้และตัวเมีย

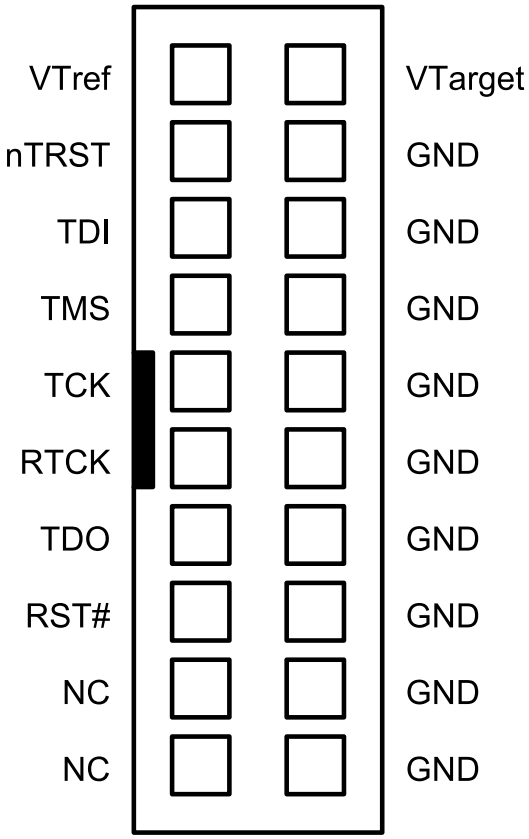
โครงสร้างบอร์ด ET-ARM7 START KIT V1.0 / EXP



รูปแสดงโครงสร้างของ ET-ARM7 START KIT V1.0 / EXP

หมายเหตุ อุปกรณ์ภายในกรอบสีแดงจะไม่มีในรุ่น “ET-ARM7 START KIT V1.0” แต่จะมีอยู่เฉพาะในรุ่น “ET-ARM7 START KIT V1.0 EXP” เท่านั้น

ขั้วต่อ ARM JTAG

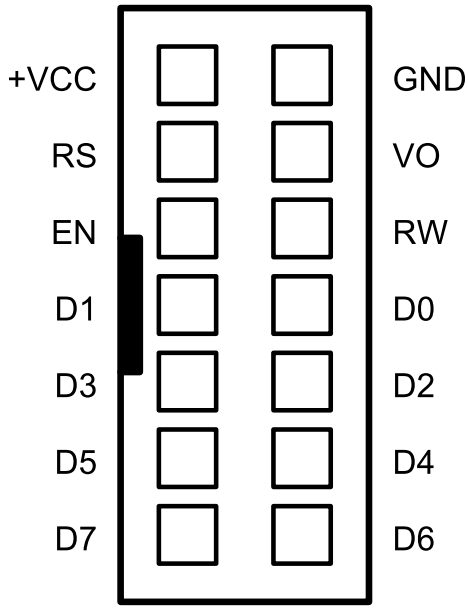


ARM JTAG Pin Connector

ขั้วต่อ “ARM JTAG” นี้ จะใช้สำหรับเชื่อมต่อกับอุปกรณ์ “ARM JTAG Debugger“ ยี่ห้อต่างๆ ที่จัดเรียงสัญญาณแบบเดียวกันกับชุด ARM JTAG ของ Multi-ICE System ซึ่งความสามารถในการทำ Debug กับบอร์ดนั้น จะขึ้นอยู่กับความสามารถของ “ARM JTAG Debugger” และโปรแกรมรองรับ ที่จะนำมาเชื่อมต่อด้วยเป็นหลัก โดยขั้วต่อนี้ ทาง อีทีที เป็นเพียงผู้ทำการออกแบบและจัดเรียงสัญญาณจากขาสัญญาณของ MCU ออกมาจัดเตรียมไว้ให้เพื่ออำนวยความสะดวกในการใช้งานเท่านั้น

ซึ่งในปัจจุบัน อุปกรณ์ที่ใช้สำหรับทำการ Debug การทำงานของ ARM ผ่านทางวงจรของ ARM JTAG นั้น จะมีราคาค่อนข้างแพงมาก

ขั้วต่อ LCD



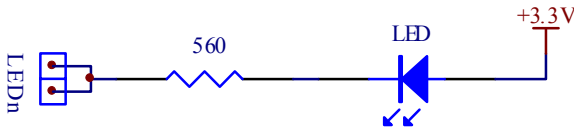
ET-CLCD

ขั้วต่อ LCD นี้ สามารถเชื่อมต่อกับ LCD แบบ Character ทุกรุ่น ที่ใช้ Controller ของ Hitachi เบอร์ HD44780 หรือเทียบเท่า โดยขั้วต่อนี้จะใช้สำหรับทำหน้าที่ เป็นจุดเชื่อมระหว่าง “ET-ARM STAMP LPC2119” กับ LCD แบบ Character

โดยการเชื่อมต่อสัญญาณของ MCU ไปยัง Character LCD นั้นจะออกแบบโดยใช้วงจรในการเชื่อมต่อเป็นแบบ 4 บิต โดยใช้สัญญาณ GPIO1[16..21] จาก MCU ในการควบคุมการแสดงผลของ LCD

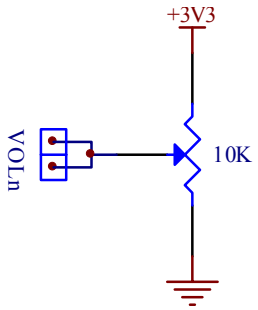
การใช้งาน LED แสดงผล

LED แสดงผลของบอร์ด จะต่อวงจรแบบรับกระแส (Sink Current) โดยใช้กับแหล่งจ่าย +3.3V ทำงานด้วยโลจิก “0” (0V) และหยุดทำงานด้วยโลจิก “1” (+3.3V) โดยมีทั้งหมด 4 ชุด โดยวงจรในส่วนนี้จะใช้สำหรับทดสอบการทำงานของ GPIO ต่างๆที่ทำงานให้ผลเป็น Output แบบโลจิก



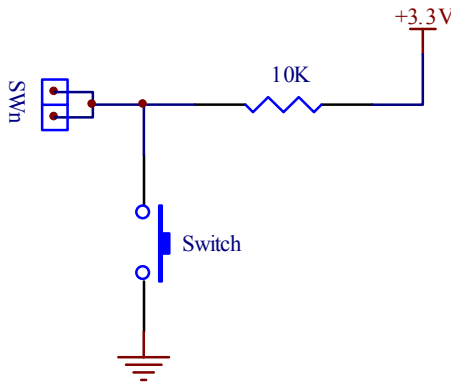
การใช้งานวงจรปรับแรงดัน (0V-3.3V)

วงจรปรับแรงดันของบอร์ดจะใช้ตัวต้านทานปรับค่าได้แบบเกือกม้า ชนิดมีแกนหมุนสำหรับปรับค่า โดยวงจรนี้ใช้กับแหล่งจ่าย +3.3V โดยจะให้ Output เป็นแรงดันซึ่งมีค่าระหว่าง 0V ถึง +3.3V ตามการปรับค่าของตัวต้านทาน ซึ่งมีทั้งหมด 4 ชุด ใช้สำหรับสร้างแรงดัน Input เพื่อทดสอบการทำงานของวงจร A/D



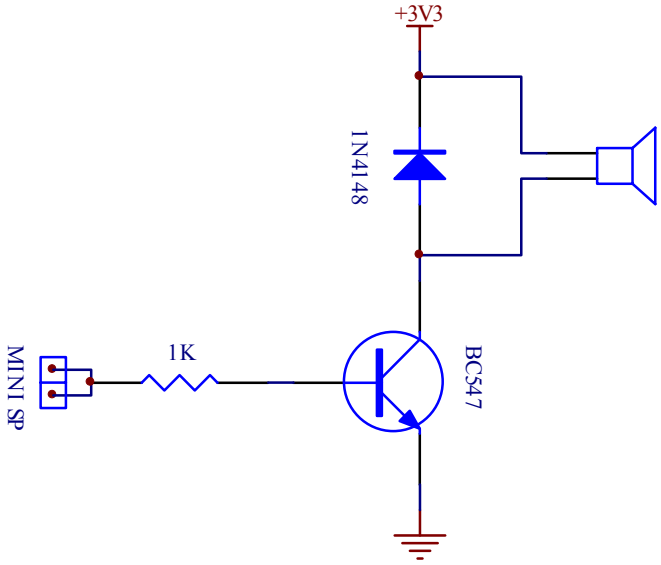
การใช้งานวงจร Push Button Switch

วงจร Push Button Switch จะใช้วงจร Switch แบบ กดติด-ปล่อยดับ (Push Button) พร้อมวงจร Pull-Up ใช้กับแหล่งจ่าย +3.3V โดยในขณะที่สวิตช์ยังไม่ถูกกดจะให้ค่าสถานะเป็นโลจิก “1” แต่เมื่อสวิตช์ถูกกดอยู่จะให้สถานะเป็นโลจิก “0” โดยวงจรส่วนนี้จะมีอยู่ด้วยกัน 4 ชุด ใช้สำหรับทดสอบการทำงานของ GPIO ต่างๆที่ต้องควบคุมการทำงานของวงจรด้วย Input แบบโลจิก



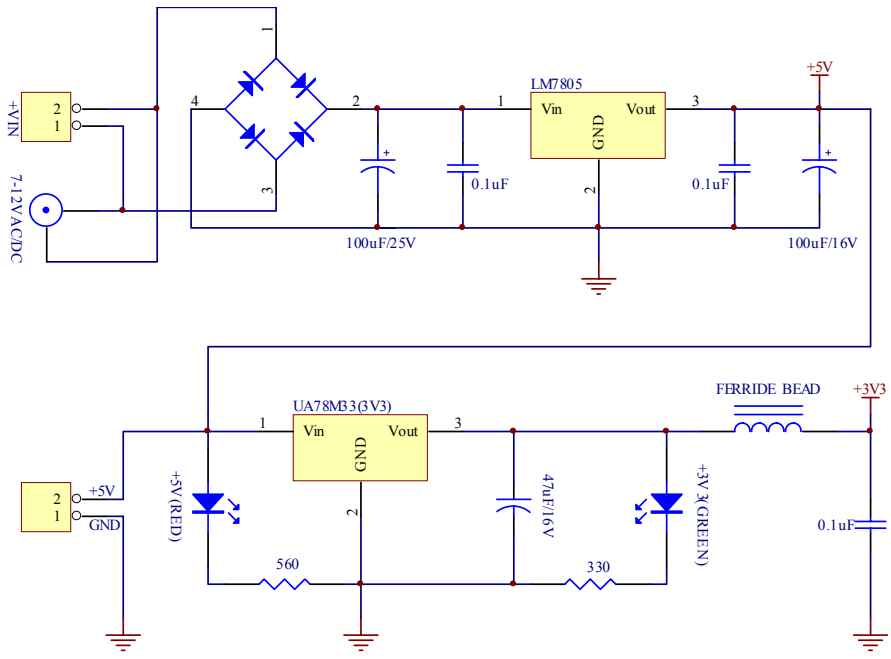
การใช้งาน วงจรกำเนิดเสียง

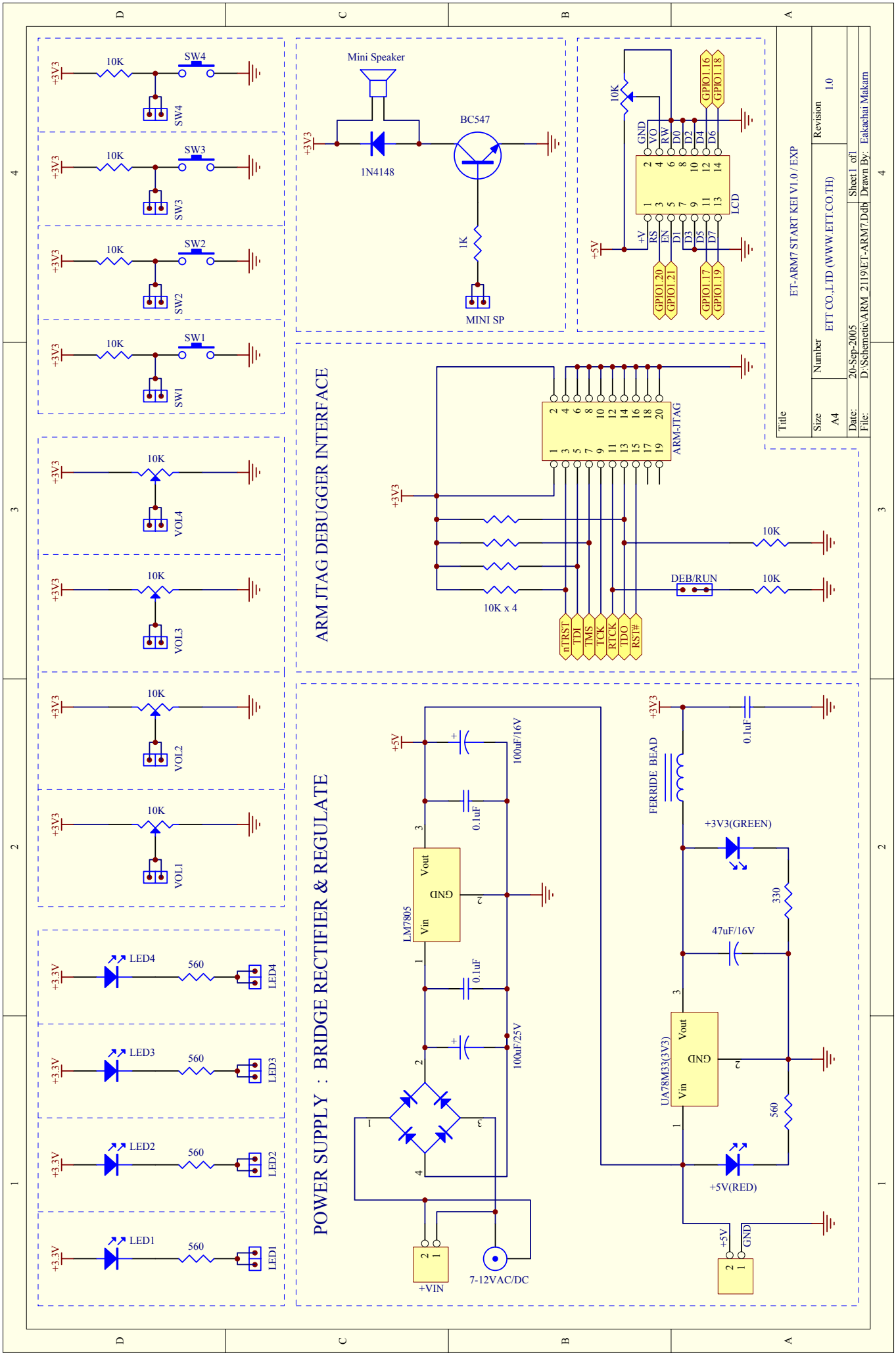
วงจรกำเนิดเสียง จะใช้ลำโพงขนาดเล็ก (Mini Speaker) พร้อมด้วยวงจรทรานซิสเตอร์แบบ NPN สำหรับขับกระแสให้กับลำโพง ใช้กับแหล่งจ่ายขนาด +3.3V ทำงานด้วยโลจิก “1” และหยุดทำงานด้วยโลจิก “0” โดยในการทำงานนั้นต้องส่งสัญญาณโลจิกที่เป็นความถี่ต่างๆให้กับลำโพงเพื่อสร้างเป็นความถี่เสียงตามต้องการ



วงจรแหล่งจ่ายไฟ

วงจรแหล่งจ่ายไฟสามารถใช้งานได้กับไฟ AC และ DC ขนาด 7-12V ได้ทันที โดยวงจรภาคแหล่งจ่ายไฟในส่วนที่เป็นวงจร Regulate นั้นจะมีทั้งส่วนที่เป็น +5V และ +3.3V




ET-ARM7 START KEI V1.0 / EXP
ETT CO.,LTD (WWW.ETT.CO.TH)
Title
Size A4
Number
Revision 1.0
Date: 20-Sep-2005
Sheet 1 of 1
File: D:\Schemetic\ARM 2119\ET-ARM7.Ddb
Drawn By: Eakachai Makarn
ARM JTAG DEBUGGER INTERFACE
ARM-JTAG
10K x 4
DEB/RUN
nTRST
TDI
TMS
TCK
RTCK
TDO
RST#
POWER SUPPLY : BRIDGE RECTIFIER & REGULATE
LM7805
UA78M33(3V3)
100uF/16V
100uF/25V
0.1uF
47uF/16V
FERRIDE BEAD
+3V3(GREEN)
+5V(RED)
330
560
+VIN
7-12VAC/DC
Mini Speaker
1N4148
BC547
1K
MINI SP
LCD
GPIO1.16
GPIO1.18
GPIO1.20
GPIO1.21
GPIO1.17
GPIO1.19
SW1
SW2
SW3
SW4
VOL1
VOL2
VOL3
VOL4
LED1
LED2
LED3
LED4

ภาคผนวก ค ฟังก์ชันและข้อมูลพื้นฐานของระบบปฏิบัติการ uC/OS-II

ฟังก์ชันและข้อมูลพื้นฐานของระบบปฏิบัติการ uC/OS-II

1. DATA STRUCTURES

1.1 Events

ตาราง ค-1 DATA STRUCTURE (Events : OS_EVENT)

Structure	OS_EVENT	EVENT CONTROL BLOCK
void *	OSEventPtr	Pointer to message or queue structure
INT8U	OSEventTbl [OS_EVENT_TBL_SIZE]	List of tasks waiting for event to occur
INT16U	OSEventCnt	Count of used when event is a semaphore
INT8U	OSEventType	OS_EVENT_TYPE_MBOX, OS_EVENT_TYPE_Q or OS_EVENT_TYPE_SEM
INT8U	OSEventGrp	Group corresponding to tasks waiting for event to occur

ตาราง ค-2 DATA STRUCTURE (Events : OS_MBOX_DATA)

Structure	OS_MBOX_DATA	MESSAGE MAILBOX DATA
void *	OSMsg	Pointer to message in mailbox
INT8U	OSEventTbl [OS_EVENT_TBL_SIZE]	List of tasks waiting for event to occur
INT8U	OSEventGrp	Group corresponding to tasks waiting for event to occur

ตาราง ค-3 DATA STRUCTURE (Events : OS_Q_DATA)

Structure	OS_Q_DATA	MESSAGE QUEUE DATA
void *	OSMsg	Pointer to next message to be extracted from queue
INT16U	OSNMsgs	Number of messages in message queue
INT16U	OSQSize	Size of message queue
INT8U	OSEventTbl [OS_EVENT_TBL_SIZE]	List of tasks waiting for event to occur
INT8U	OSEventGrp	Group corresponding to tasks waiting for event to occur

ตาราง ค-4 DATA STRUCTURE (Events : OS_SEM_DATA)

Structure	OS_SEM_DATA	SEMAPHORE DATA
INT16U	OSCnt	Semaphore count
INT8U	OSEventTbl [OS_EVENT_TBL_SIZE]	List of tasks waiting for event to occur
INT8U	OSEventGrp	Group corresponding to tasks waiting for event to occur

1.2 Memory Partition

ตาราง ค-5 DATA STRUCTURE (Memory partition : OS_MEM)

Structure	OS_MEM	MEMORY CONTROL BLOCK
void *	OSMemAddr	Pointer to beginning of memory partition
void *	OSMemFreeList	Pointer to list of free memory blocks
INT32U	OSMemBlkSize	Size (in bytes) of each block of memory
INT32U	OSMemNBlks	Total number of blocks in this partition
INT32U	OSMemNFree	Number of memory blocks remaining in this partition

ตาราง ค-6 DATA STRUCTURE (Memory partition : OS_MEM_DATA)

Structure	OS_MEM_DATA	MEMORY CONTROL BLOCK DATA
void *	OSMemAddr	Pointer to beginning of memory partition
void *	OSMemFreeList	Pointer to list of free memory blocks
INT32U	OSMemBlkSize	Size (in bytes) of each block of memory
INT32U	OSMemNBlks	Total number of blocks in this partition
INT32U	OSNFree	Number of memory blocks free
INT32U	OSNUsed	Number of memory blocks used

1.3 Task

ตาราง ค-7 DATA STRUCTURE (Task : OS_STK_DATA)

Structure	OS_STK_DATA	TASK STACK DATA
INT32U	OSFree	Number of free bytes on the stack
INT32U	OSUsed	Number of bytes used on the stack

ตาราง ค-8 DATA STRUCTURE (Task : OS_TCB)

Structure	OS_TCB	TASK CONTROL BLOCK
OS_STK *	OSTCBStkPtr	Pointer to current top of stack
void *	OSTCBExtPtr	Pointer to user deÞnable data for TCB extension
OS_STK *	OSTCBStkBottom	Pointer to bottom of stack
INT32U	OSTCBStkSize	Size of task stack (in bytes)
INT16U	OSTCBOpt	Task options as passed by OSTaskCreateExt()
INT16U	OSTCBId	Task ID (0..65535)
struct os_tcb*	OSTCBNext	Pointer to next TCB in the TCB list
struct os_tcb*	OSTCBPrev	Pointer to previous TCB in the TCB list
OS_EVENT	OSTCBEventPtr	Pointer to event control block
void *	OSTCBMsg	Message received from OSMboxPost() or OSQPost()
INT16U	OSTCBDly	Nbr ticks to delay task or, timeout waiting for event
INT8U	OSTCBStat	Task status
INT8U	OSTCBPrio	Task priority (0 == highest, 63 == lowest)

ตาราง ค-8 (ต่อ)

Structure	OS_TCB	TASK CONTROL BLOCK
INT8U	OSTCBX	Bit position in group corresponding to task priority (0..7)
INT8U	OSTCBY	Index into ready table corresponding to task priority
INT8U	OSTCBBitX	Bit mask to access bit position in ready table
INT8U	OSTCBBitY	Bit mask to access bit position in ready group
BOOLEAN	OSTCBDelReq	Indicates whether a task needs to delete itself

2. GLOBAL VARIABLES

ตาราง ค-9 GLOBAL VARIABLES

GLOBAL VARIABLES		
INT32U	OSCtxSwCtr	Counter of number of context switches
OS_EVENT *	OSEventFreeList	Pointer to list of free EVENT control blocks
OS_EVENT	OSEventTbl [OS_MAX_EVENTS]	Table of EVENT control blocks
INT32U	OSIdleCtr	Idle counter
INT8S	OSCPUUsage	Percentage of CPU used
INT32U	OSIdleCtrMax	Maximum value that idle counter can take in 1 sec.
INT32U	OSIdleCtrRun	Value reached by idle counter at run time in 1 sec.

ตาราง ค-9 (ต่อ)

GLOBAL VARIABLES		
BOOLEAN	OSStatRdy	Flag indicating that the statistic task is ready
INT8U	OSIntNesting	Interrupt nesting level
INT8U	OSLockNesting	Multitasking lock nesting level
INT8U	OSPrioCur	Priority of current task
INT8U	OSPrioHighRdy	Priority of highest priority task
INT8U	OSRdyGrp	Ready list group
INT8U	OSRdyTbl [OS_RDY_TBL_SIZE]	Table of tasks which are ready to run
BOOLEAN	OSRunning	Flag indicating that kernel is running
INT8U	OSTaskCtr	Number of tasks created
OS_TCB *	OSTCBCur	Pointer to currently running TCB
OS_TCB *	OSTCBFreeList	Pointer to list of free TCBs
OS_TCB *	OSTCBHighRdy	Pointer to highest priority TCB ready to run
OS_TCB *	OSTCBList	Pointer to doubly linked list of TCBs
OS_TCB *	OSTCBPrioTbl [OS_LOWEST_PRIO+1]	Table of pointers to created TCBs
INT32U	OSTime	Current value of system time (in ticks)
INT8U const	OSMapTbl[]	Priority->Bit Mask lookup table
INT8U const	OSUnMapTbl[]	Priority->Index lookup table

3. FUNCTION PROTOTYPES

3.1 Semaphores

```
INT16U OSSemAccept(OS_EVENT *pevent);
OS_EVENT *OSSemCreate(INT16U cnt);
```

OS_EVENT *OSSemDel(OS_EVENT *pevent, INT8U opt, INT8U *err);

void OSSemPend(OS_EVENT *pevent, INT16U timeout, INT8U *err);

INT8U OSSemPost(OS_EVENT *pevent);

INT8U OSSemQuery(OS_EVENT *pevent, OS_SEM_DATA *pdata);

OSSemDel() opt:

OS_DEL_NO_PEND

OS_DEL_ALWAYS

OS_SEM_DATA:

INT16U OSCnt;

INT8U OSEventTbl[];

INT8U OSEventGrp;

3.2 Mutual Exclusion Semaphores

INT8U OSMutexAccept(OS_EVENT *pevent, INT8U *err);

OS_EVENT *OSMutexCreate(INT8U prio, INT8U *err);

OS_EVENT *OSMutexDel (OS_EVENT *pevent, INT8U opt, INT8U *err);

void OSMutexPend(OS_EVENT *pevent, INT16U timeout, INT8U *err);

INT8U OSMutexPost(OS_EVENT *pevent);

INT8U OSMutexQuery(OS_EVENT *pevent, OS_MUTEX_DATA *pdata);

OSMutexDel() opt:

OS_DEL_NO_PEND

OS_DEL_ALWAYS

OS_MUTEX_DATA:

INT8U OSEventTbl[];

INT8U OSEventGrp;

INT8U OSValue;

INT8U OSOwnerPrio;

INT8U OSMutexPIP;

3.3 Event Flags

OS_FLAGS OSFlagAccept(OS_FLAG_GRP *pgrp, OS_FLAGS flags, INT8U wait_type, INT8U *err);

OS_FLAG_GRP *OSFlagCreate(OS_FLAGS flags, INT8U *err);

OS_FLAG_GRP *OSFlagDel(OS_FLAG_GRP *pgrp, INT8U opt, INT8U *err);

OS_FLAGS OSFlagPend(OS_FLAG_GRP *pgrp, OS_FLAGS flags, INT8U wait_type, INT16U timeout, INT8U *err);

OS_FLAGS OSFlagPost(OS_FLAG_GRP *pgrp, OS_FLAGS flags, INT8U operation, INT8U *err);

OS_FLAGS OSFlagQuery(OS_FLAG_GRP *pgrp, INT8U *err);

OSFlagDel() opt:

OS_DEL_NO_PEND

OS_DEL_ALWAYS

operation:

OS_FLAG_CLR

OS_FLAG_SET

wait_type:

OS_FLAG_WAIT_CLR_ALL

OS_FLAG_WAIT_CLR_AND

OS_FLAG_WAIT_CLR_ANY

OS_FLAG_WAIT_CLR_OR

OS_FLAG_WAIT_SET_ALL

OS_FLAG_WAIT_SET_AND

OS_FLAG_WAIT_SET_ANY

OS_FLAG_WAIT_SET_OR

+ OS_FLAG_CONSUME

3.4 Message Mailboxes

void *OSMboxAccept(OS_EVENT *pevent);

OS_EVENT *OSMboxCreate(void *msg);

OS_EVENT *OSMboxDel(OS_EVENT *pevent, INT8U opt, INT8U *err);

void *OSMboxPend(OS_EVENT *pevent, INT16U timeout, INT8U *err);

INT8U OSMboxPost(OS_EVENT *pevent, void *msg);

INT8U OSMboxPostOpt(OS_EVENT *pevent, void *msg, INT8U opt);

INT8U OSMboxQuery(OS_EVENT *pevent, OS_MBOX_DATA *pdata);

OSMboxDel() opt:

OS_DEL_NO_PEND

OS_DEL_ALWAYS

OSMboxPostOpt() opt:

OS_POST_OPT_NONE

OS_POST_OPT_BROADCAST

OS_MBOX_DATA:

void *OSMsg;

INT8U OSEventTbl[];

INT8U OSEventGrp;

3.5 Message Queues

void *OSQAccept(OS_EVENT *pevent);

OS_EVENT *OSQCreate(void **start, INT16U size);

OS_EVENT *OSQDel(OS_EVENT *pevent, INT8U opt, INT8U *err);

INT8U OSQFlush(OS_EVENT *pevent);

void *OSQPend(OS_EVENT *pevent, INT16U timeout, INT8U *err);

INT8U OSQPost(OS_EVENT *pevent, void *msg);

INT8U OSQPostFront(OS_EVENT *pevent, void *msg);

INT8U OSQPostOpt(OS_EVENT *pevent, void *msg, INT8U opt);

INT8U OSQQuery(OS_EVENT *pevent, OS_Q_DATA *pdata);

OSQDel() opt:

OS_DEL_NO_PEND

OS_DEL_ALWAYS

OSQPostOpt() opt:

OS_POST_OPT_NONE

OS_POST_OPT_BROADCAST

OS_POST_OPT_FRONT

OS_Q_DATA:

void *OSMsg;

INT16U OSNMsgs;

INT16U OSQSize;

INT8U OSEventTbl[];

INT8U OSEventGrp;

3.6 Memory Management

OS_MEM *OSMemCreate(void *addr, INT32U nblks, INT32U blksize, INT8U *err);

void *OSMemGet(OS_MEM *pmem, INT8U *err);

INT8U OSMemPut(OS_MEM *pmem, void *pblk);

```
INT8U OSMemQuery(OS_MEM *pmem, OS_MEM_DATA *pdata);

OS_MEM_DATA:
        void *OSAddr;
        void *OSFreeList;
        INT32U OSBlkSize;
        INT32U OSNBlks;
        INT32U OSNFree;
        INT32U OSNUsed;
```

3.7 Task Management

```
INT8U OSTaskChangePrio(INT8U oldprio, INT8U newprio);
INT8U OSTaskCreate(void (*task)(void *pd), void *pdata, OS_STK *ptos,
INT8U prio);
INT8U OSTaskCreateExt(void (*task)(void *pd),
                void *pdata,
                OS_STK *ptos,
                INT8U prio,
                INT16U id,
                OS_STK *pbos,
                INT32U stk_size,
                void *pext,
                INT16U opt);
INT8U OSTaskDel(INT8U prio);
INT8U OSTaskDelReq(INT8U prio);
INT8U OSTaskResume(INT8U prio);
INT8U OSTaskSuspend(INT8U prio);
INT8U OSTaskStkChk(INT8U prio, OS_STK_DATA *pdata);
INT8U OSTaskQuery(INT8U prio, OS_TCB *pdata);
```

OSTaskCreateExt() opt:

OS_TASK_OPT_STK_CHK

OS_TASK_OPT_STK_CLR

OS_TASK_OPT_SAVE_FP

OS_STK_DATA:

INT32U OSFree;

INT32U OSUsed;

OS_TCB:

OS_STK *OSTCBStkPtr;

void *OSTCBExtPtr;

OS_STK *OSTCBStkBottom;

INT32U OSTCBStkSize;

INT16U OSTCBOpt;

INT16U OSTCBId;

OS_TCB *OSTCBNext;

OS_TCB *OSTCBPrev;

OS_EVENT *OSTCBEventPtr;

void *OSTCBMsg;

OS_FLAG_NODE *OSTCBFlagNode;

OS_FLAGS OSTCBFlagsRdy;

INT16U OSTCBDly;

INT8U OSTCBStat;

INT8U OSTCBPrio;

INT8U OSTCBX;

INT8U OSTCBY;

INT8U OSTCBBitX;

INT8U OSTCBBitY;

BOOLEAN OSTCBDelReq;

3.8 Time Management

```
void OSTimeDly(INT16U ticks);
INT8U OSTimeDlyHMSM(INT8U hr, INT8U min, INT8U sec, INT16U ms);
INT8U OSTimeDlyResume(INT8U prio);
INT32U OSTimeGet(void);
void OSTimeSet(INT32U ticks);
```

3.9 Miscellaneous

```
void OSInit(void);
void OSIntEnter(void);
void OSIntExit(void);
void OSSchedLock(void);
void OSSchedUnlock(void);
void OSStart(void);
void OSStatInit(void);
INT16U OSVersion(void);
```

4. ERROR CODES

ตาราง ค-10 ERROR CODES

#define	**0**	
OS_NO_ERR	0	No error
OS_ERR_EVENT_TYPE	1	bad event type for *pevent
OS_ERR_PEND_ISR	2	bloquing function call from ISR forbiden
OS_TIMEOUT	10	timeout reached
OS_TASK_NOT_EXIST	11	task is not created or already deleted
OS_MBOX_FULL	20	message box is full

ตาราง ค-10 (ต่อ)

#define	**0**	
OS_Q_FULL	30	queue is full
OS_PRIO_EXIST	40	this priority level is already used by a existing task
OS_PRIO_ERR	41	Task do not exist
OS_PRIO_INVALID	42	priority >= OS_LOWEST_PRIO
OS_SEM_OVF	50	semaphore exceed 65535 (overßow)
OS_TASK_DEL_ERR	60	can not delete a nonexistant task
OS_TASK_DEL_IDLE	61	can not delete the IDLE task
OS_TASK_DEL_REQ	62	if an other task request this running task to delete itself (this is not an error)
OS_TASK_DEL_ISR	63	task deletion from an ISR isforbiden
OS_NO_MORE_TCB	70	no more Task Control Block
OS_TIME_NOT_DLY	80	Try to resume an unsuspended task
OS_TIME_INVALID_MINUTES	81	minutes > 59
OS_TIME_INVALID_SECONDS	82	seconds > 59
OS_TIME_INVALID_MILLI	83	milli > 999
OS_TIME_ZERO_DLY	84	delay is zero
OS_TASK_SUSPEND_PRIO	90	trying to suspend a nonexistant task
OS_TASK_SUSPEND_IDLE	91	trying to suspend the IDLE task
OS_TASK_RESUME_PRIO	100	trying to resume a nonexistant task
OS_TASK_NOT_SUSPENDED	10	trying to resume a non-suspended task
OS_MEM_INVALID_PART	110	no more partition
OS_MEM_INVALID_BLKS	111	number of block < 2
OS_MEM_INVALID_SIZE	112	block size < sizeof (void *)
OS_MEM_NO_FREE_BLKS	113	no more free blocks in this partition

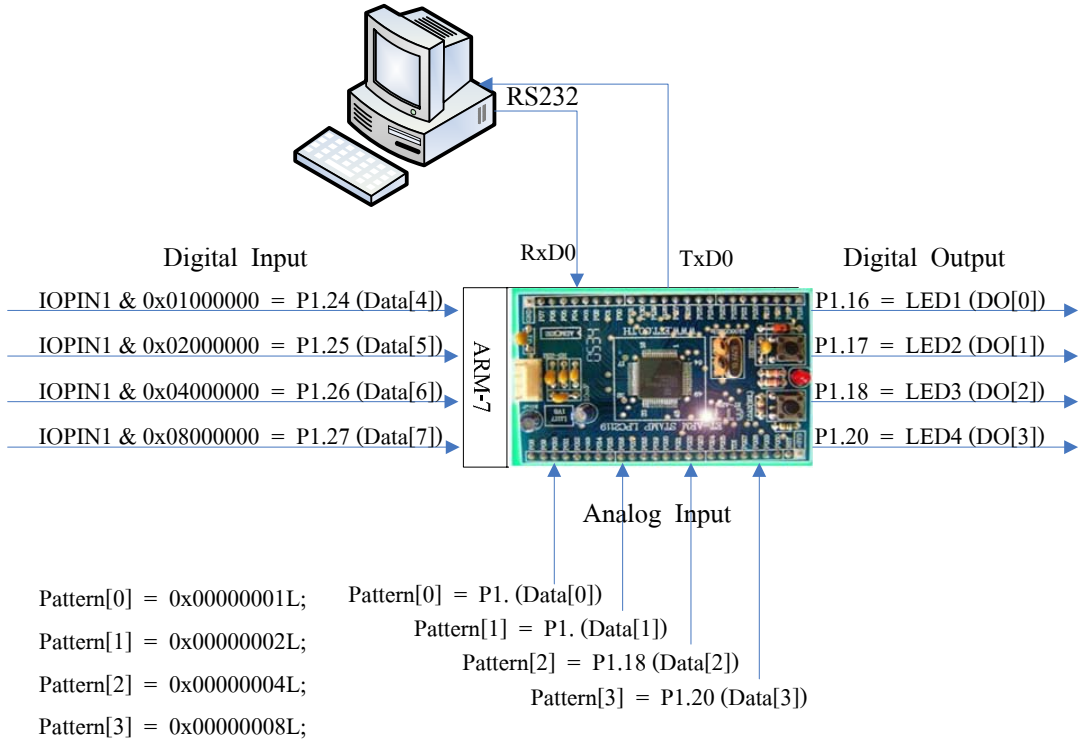
ตาราง ค-10 (ต่อ)

#define	**0**	
OS_MEM_FULL	114	number of return blocks exceed the number blocks in this partition Ñ something goes wrong !
OS_TASK_OPT_ERR	130	OSTaskStkChk() need a task created with OSTaskCreatedExt() and OS_TASK_OPT_STK_CHK on

ภาคผนวก ง งาน (Tasks) ของ Modbus slave ใน uC/OS-II บน ARM-7

งาน (Tasks) ของ Modbus slave ใน uC/OS-II บน ARM-7

งานที่ทำหน้าที่เกี่ยวกับ Modbus slvae ใน uC/OS-II บน ARM-7 ประกอบด้วย 2 งาน คือ RxTask และ ReadADCTask โดย block diagram ของ ARM-7 ที่ใช้อ้างอิงในโปรแกรมแสดงดังภาพประกอบ ง-1 และ source code ของทั้ง 2 งานแสดงดังภาพประกอบ ง-2 และ ง-3 ตามลำดับ



ภาพประกอบ ง-1 block diagram ของ ARM-7

```
static  void  RXTask (void *p_arg)
{
        int crc;            /* CRC */
        (void)p_arg;        /* avoid compiler warning */
        int i, j, k;

        for(;;)
        {
                if (MyRxIndex >= 8)             /* check packet in receive buffer */
                {
                        LED_On(3);
                        MyRxIndex = 0;          /* clear index of receive buffer */

                        for (i = 0; i < 8; i++)     /* copy data from receive buffer to rxbuf */
                                rxbuf[i] = MyRxBuf[i];

                        rxbuf[i] = '\0';
                        OSTimeDlyHMSM(0, 0, 0, 10);

                        txbuf[0] = rxbuf[0];
                        txbuf[1] = rxbuf[1];

                        /* covert 2 bytes of start address to ParNo */
                        ParNo = (unsigned int)rxbuf[2] * 256 + (unsigned int)rxbuf[3];

                        /* convert 2 bytes of data to write (used when function is write) */
                        Data2Write = (unsigned int)rxbuf[4] * 256 + (unsigned int)rxbuf[5];
```

ภาพประกอบ ง-2 โปรแกรม RXTask

```
/* if function is reading, PointNo is number of point to read from start
   address that the same data of Data2Write*/
PointNo = Data2Write;

/* check start address, 8-11 is DO  (function is writing) */
if ((ParNo >= 8) && (ParNo <= 11))
{
        /*slave will send same data that received if function is write */
         txbuf[2] = rxbuf[2];
         txbuf[3] = rxbuf[3];
         txbuf[4] = rxbuf[4];
         txbuf[5] = rxbuf[5];

         /* begin calculate CRC */
         crc =0xffff;
         for(i=0;i<6;i++)
                  crc=crc_calc(crc, (unsigned int)txbuf[i]);

         /* copy CRC to the last 2 bytes of transmit buffer */
         txbuf[6] = (unsigned char)(crc & 0x00ff);
         txbuf[7] = (unsigned char)((crc >> 8) & 0x00ff);

         for (i = 0; i < 8; i++)       /* send data to master */
                  putchar((int)txbuf[i]);
}
```

ภาพประกอบ ง-2 (ต่อ)

```
        else    /* function is reading */
        {
                txbuf[2] = PointNo * 0x02;      /* no. of byte sent to master */
                k = 0;
                for (j = 0; j < (PointNo*2); j=j+2)  /* read data */
                {
                        txbuf[j+3] = (char)((Data[ParNo+k] >> 8) & 0xffL);
                        txbuf[j+4] = (char)(Data[ParNo+k] & 0xffL);
                        k++;
                }
                crc =0xffff;
                k = txbuf[2]+3;

                for(i = 0; i < k; i++)          /* calculate CRC */
                        crc=crc_calc(crc, (unsigned int)txbuf[i]);

                /* copy CRC to the last 2 bytes of transmit buffer */
                txbuf[k] = (unsigned char)(crc & 0x00ff);
                txbuf[k+1] = (unsigned char)((crc >> 8) & 0x00ff);

                OSTimeDlyHMSM(0, 0, 0, 5);

                for (i = 0; i < k+2; i++)    /* send data to master */
                        putchar((int)txbuf[i]);
        }
        TxFinish = 1;
}
```

ภาพประกอบ ง-2 (ต่อ)

```
                else    /* packet in receive buffer not ready to use */
                {
                        OSTimeDlyHMSM(0, 0, 0, 7);
                        LED_Off(3);
                }
        }
}
```

ภาพประกอบ ง-2 (ต่อ)

```
static  void  ReadADCTask (void *p_arg)
{
        (void)p_arg;
        int i;

        for(;;)
        {
                for (i = 0; i < 4; i++)
                {
                        /* clear Ain3:0 is to be sampled and converted */
                        ADCR &= 0xffffff00;
                        ADCR |= Pattern[i];       /* selece Ain pin */

                        OSTimeDlyHMSM(0, 0, 0, 100);
```

ภาพประกอบ ง-3 โปรแกรม ReadADCTask

```
            do      /* Loop Read ADC        */
            {
                    Data[i] = ADDR;             /* read A/D data register  */
            }while((Data[i] & 0x80000000) == 0); /*  wait A/C
                                                    conversion complete */

            Data[i] = (Data[i] >> 6) & 0x03FF;  /* shift ADC result to integer   */
            OSTimeDlyHMSM(0, 0, 0, 100);
    }
    /* read DI */
    if (!(IOPIN1 & 0x01000000))             /* check PORT1.24 */
            Data[4] = 0;
    else
            Data[4] = 1;

    if (!(IOPIN1 & 0x02000000))             /* check PORT1.25 */
            Data[5] = 0;
    else
            Data[5] = 1;

    if (!(IOPIN1 & 0x04000000))             /* check PORT1.26 */
            Data[6] = 0;
    else
            Data[6] = 1;

    if (!(IOPIN1 & 0x08000000))             /* check PORT1.27 */
            Data[7] = 0;
    else
            Data[7] = 1;
```

ภาพประกอบ ง-3 (ต่อ)

```
OSTimeDlyHMSM(0, 0, 0, 100);

/* check DO address and write data to Digital Output buffer */
if (ParNo == 8)
        DO[0] = Data2Write;
if (ParNo == 9)
        DO[1] = Data2Write;
if (ParNo == 10)
        DO[2] = Data2Write;
if (ParNo == 11)
        DO[3] = Data2Write;

/*check DO and update */
if (DO[0])
        LED_On(1);
else
        LED_Off(1);


if (DO[1])
        LED_On(2);
else
        LED_Off(2);

if (DO[2])
        LED_On(3);
else
        LED_Off(3);
```

ภาพประกอบ ง-3 (ต่อ)

```
                if (DO[3])
                        LED_On(5);
                else
                        LED_Off(5);
        }
}
```

ภาพประกอบ ง-3 (ต่อ)

ภาคผนวก จ การพัฒนาเครื่องมือที่ใช้ในการทดสอบ

การพัฒนาเครื่องมือที่ใช้ในการทดสอบ

เครื่องมือที่พัฒนาขึ้นมาใช้ในการทดสอบในงานวิจัยนี้มีทั้งหมด 2 ตัว คือ External Interrupt Generator เป็นเครื่องมือที่พัฒนาด้วย TURBO C และ Modbus Slave Tester เป็นเครื่องมือที่พัฒนาด้วย Microsoft Visual Basic 6.0

1. External Interrupt Generator

External Interrupt Generator เป็นเครื่องมือที่พัฒนาด้วย TURBO C ใช้ในการสร้างสัญญาณจากภายนอกเพื่อเป็น external interrupt ให้กับไมโครโพรเซสเซอร์ ARM-7 โดยมีการติดต่อสื่อสารระหว่างขา P0.16 ของ ARM-7 และ printer port ของเครื่องไมโครโพรเซสเซอร์ ดังนั้นในหัวข้อนี้จะกล่าวถึงความรู้ทั่วไปในการโปรแกรมเพื่อติดต่อสื่อสารผ่าน printer port และ Source code ที่พัฒนาขึ้นสำหรับรับ/ส่งสัญญาณผ่าน printer port

1.1 ความรู้ทั่วไปในการโปรแกรมเพื่อติดต่อสื่อสารผ่าน printer port

parallel port หรือ printer port เป็นอุปกรณ์มาตรฐานที่ใช้ในการติดต่อกับเครื่องพิมพ์ของคอมพิวเตอร์มีความสามารถเป็นได้ทั้ง Input Port หรือ Output Port สำหรับขาต่าง ๆ ของ Parallel Port แสดงดังตาราง จ-1

ตาราง จ-1 ขาและลักษณะการใช้งานของ Parallel Port

Pin No (D-Type 25)	SPP Signal	Direction In/Out	Register	Hardware Inverted
1	nStrobe	In/Out	Control	Yes
2	Data 0	Out	Data	
3	Data 1	Out	Data	
4	Data 2	Out	Data	
5	Data 3	Out	Data	
6	Data 4	Out	Data	

ตาราง จ-1 (ต่อ)

Pin No (D-Type 25)	SPP Signal	Direction In/Out	Register	Hardware Inverted
7	Data 5	Out	Data	
8	Data 6	Out	Data	
9	Data 7	Out	Data	
10	nAck	In	Status	
11	Busy	In	Status	Yes
12	Paper-Out	In	Status	
13	Select	In	Status	
14	nAuto-LineFeed	In/Out	Control	Yes
15	nError/nFault	In	Status	
16	nInitialize	In/Out	Control	
17	nSelect-Printer	In/Out	Control	Yes
18-25	Ground	Ground		

ตาราง จ-2 ค่า Register ต่าง ๆ ของ Data Port

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write/Read*	Bit 7	Data 7 (Pin 9)
			Bit 6	Data 6 (Pin 8)
			Bit 5	Data 5 (Pin 7)
			Bit 4	Data 4 (Pin 6)
			Bit 3	Data 3 (Pin 5)
			Bit 2	Data 2 (Pin 4)
			Bit 1	Data 1 (Pin 3)
			Bit 0	Data 0 (Pin 2)

Parallel Port แต่ละ Port ในโหมดการทำงานแบบ SPP จะมี Register อยู่ทั้งหมด 3 ชุดใช้ในการรับส่งข้อมูล ประกอบด้วย Control Port, Data Port และ Status Port แต่ละ Port ทำหน้าที่ดังตาราง จ-2 ถึง จ-4

* สำหรับ Data Port นี้โดยปรกติจะอยู่ในสถานะสำหรับเขียนแต่หากมีการกำหนดค่าใน Control Port บิตที่ 5 Data Port จะสามารถใช้ในการอ่านได้ด้วย

ตาราง จ-3 ค่า Register ต่าง ๆ ของ Status Port

Offset	Name	Read/Write	Bit No.	Properties
Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ(Not)
			Bit 1	Reserved
			Bit 0	Reserved

ตาราง จ-4 ค่า Register ต่าง ๆ ของ Control Port

Offset	Name	Read/Write	Bit No.	Properties
Base + 2	Control Port	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable Bi-Directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

การอ่านหรือเขียนค่า Register ของ Parallel Port จะอ่านโดยใช้คำสั่ง inportb(io_port) สำหรับอ่านค่าจาก Register และ outportb(io_port, char var) สำหรับกำหนดค่า var ให้ Register โดยที่ค่า io_port จะเป็นค่าตำแหน่งของที่อยู่ (Port Address) ที่ใช้อ้างถึงค่า Register ต่าง ๆ โดยปรกติสำหรับ Parallel Port แล้วจะกำหนดค่า Port Address ตรงกับ Register ดังตาราง จ-5 ส่วนตำแหน่ง IRQ นั้นปรกติจะมีค่าเป็น 5 หรือ 7 แต่อาจไม่ตรงกันในเครื่องคอมพิวเตอร์แต่ละเครื่องต้องเข้าไปตรวจดูที่ BIOS อีกครั้งหนึ่ง

ตาราง จ-5 Parallel Port Address

Address	Notes:
378h - 37Fh	Usual Address For LPT 1
278h - 27Fh	Usual Address For LPT 2

ค่าแรกในตารางจะเป็นค่า Base Port Address ของ Parallel Port นั้นๆ จากตารางที่ 2, 3 และ 4 ในช่อง Offset จะบอกค่า Port Address ต่าง ๆ ของ Register เช่น สำหรับ LPT1 มีค่า Base Port Address เท่ากับ 378h แสดงว่าสามารถติดต่อ Data Port Register ได้ที่ Port Address 378h, Status Port Register ได้ที่ Port Address 379h และ Control Port Register ได้ที่ Port Address 37Ah

1.2 Source code ที่พัฒนาขึ้นสำหรับรับ/ส่งสัญญาณผ่าน printer port

```
#include        <stdio.h>
#include        <conio.h>
main()
{
        long    i, LoopCnt = 0L;
        outportb(0x378, 0xff);                  /* no wake up */
        for (;;)
        {
                while (inportb(0x379) == 0x7f)  /* wait for '0' (sleep) */  (1)
```

```
                ;
        for (LoopCnt = 0L; LoopCnt < 5000000L; LoopCnt++)      (2)
                ;
        outportb(0x378, 0x00);              /* press SW */          (3)
        while (inportb(0x379) == 0x6f)      /* wait if '1'(Run) */  (4)
                ;
        outportb(0x378, 0xff);                                      (5)
    }
}
```

(1) รอสัญญาณ idle mode ที่ส่งมาจาก ARM-7

(2) Delay ช่วงระยะเวลาหนึ่งก่อนที่จะส่งสัญญาณออกผ่าน printer port

(3) ส่งสัญญาณออกผ่าน printer port ซึ่งเปรียบเสมือนการกดสวิตช์เพื่อเป็นสัญญาณ interrupt ให้กับ ARM-7

(4) รอสัญญาณที่ ARM-7 อยู่ในสถานการณ์ทำงานปกติกลับมาเพื่อส่งสัญญาณเคลียร์ค่า interrupt กลับไปยัง ARM-7

(5) ส่งสัญญาณเคลียร์ค่า interrupt

เพื่อความสะดวกในการทดสอบและสามารถสังเกตความเปลี่ยนแปลงที่เกิดขึ้นอันเนื่องมาจาก delay ได้จึงทำการปรับโปรแกรมให้รับค่า delay จากผู้ใช้ก่อนรันและสามารถเปลี่ยนแปลงค่านั้น ๆ ได้ด้วยการกดคีย์บอร์ด 4 ค่า คือ “d”, “u”, “j”, และ “k” แต่ละตัวมีความหมายดังนี้

“d” หมายถึง กำหนดให้ค่า delay ที่ทำงาน ณ เวลานั้นมีค่าลดลง 1 ค่า

“u” หมายถึง กำหนดให้ค่า delay ที่ทำงาน ณ เวลานั้นมีค่าเพิ่มขึ้น 1 ค่า

“j” หมายถึง กำหนดให้ค่า delay ที่ทำงาน ณ เวลานั้นมีค่าลดลง 10 ค่า

“k” หมายถึง กำหนดให้ค่า delay ที่ทำงาน ณ เวลานั้นมีค่าเพิ่มขึ้น 10 ค่า

โปรแกรมที่พัฒนาขึ้นให้สามารถปรับเปลี่ยนค่า delay ด้วยการกดคีย์ 4 ตัว ดังนี้

```
#include        <stdio.h>
#include        <conio.h>
#include        <dos.h>

main(argc, argv)
int     argc;
char    *argv[];
{
        unsigned int DelayVal;
        unsigned char   v;
        long LoopCnt = 0;

        if (argc != 2)
        {  /* ext_int is executable program of External  Interrupt  Generator */
                printf("\nUsage : ext_int  DelayValue \n");
                exit(0);
        }
        /* assign DelayVal value from argv[1] (passing value from user) */
        DelayVal = (unsigned int)atoi(argv[1]);
        outportb(0x378, 0xff);    /* no wake up */

        for (;;)
        {
                if (kbhit())  /* check keyboard to increase or decrease delay */
                {
                        v = getch();
                        if (v == 'd')       /* DelayVal is decrease 1 point */
                                DelayVal--;
                        if (v == 'u')       /* DelayVal is increase 1 point */
                                DelayVal++;
```

```
                    if (v == 'j')       /* DelayVal is decrease 10 points */
                            DelayVal = DelayVal - 10;
                    if (v == 'k')       /* DelayVal is increase 10 points */
                            DelayVal = DelayVal + 10;
            }
            while (inportb(0x379) == 0x7f)    /* wait for '0' (sleep) */
                    ;
            printf("\rDelay = %05d LoopCnt = %09ld ?", DelayVal,
                    LoopCnt++);

            /* delay by DelayVal before wake-up signal sent */
            GoToDelay(DelayVal);

            outportb(0x378, 0x00);              /* press SW */

            while (inportb(0x379) == 0x6f)    /* wait if '1'(Run) */
                    ;
            outportb(0x378, 0xff);
    }
}

/*  function delay */
GoToDelay(n)
int     n;
{
    int     i;

    while (n--)  /* go to loop while to delay and decrease n until n = 0 */
    {
            for (i = 0; i < 100; i++)
```

```
                                    ;
                }
        }
```

GoToDelay เป็นฟังก์ชันสำหรับ delay ด้วยค่าต่าง ๆ ที่ส่งมาจากฟังก์ชันที่เรียกใช้โดยรับค่านั้น ๆ ด้วยตัวแปร n

2. Modbus Slave Tester

Modbus Slave Tester เป็นเครื่องมือที่พัฒนาด้วย Microsoft Visual Basic 6.0 ทำหน้าที่เป็น Modbus master ซึ่งจะตรวจสอบการทำงานเชิงเวลาจริงของระบบโดยตรวจสอบความถูกต้องของเวลาและคำตอบหรือตรรกะโดยมีการโปรแกรมดังภาพประกอบ จ-1 ถึง จ-7

```
Private Sub Form_Load()

   TimeOutCnt = 0                                (1)

   MSComm1.CommPort = 1                          (2)
   MSComm1.PortOpen = True                       (3)
   MSComm1.Settings = "9600,N,8,1"               (4)
   MSComm1.InputLen = 1
   MSComm1.RThreshold = 1
   MSComm1.RTSEnable = True
   MSComm1.DTREnable = True

   Timer2.Enabled = False
   Timer1.Enabled = True
   DataErrCnt = 0                                (5)
   CountTx = 0                                   (6)
End Sub
```

ภาพประกอบ จ-1 Form_Load

(1) กำหนดค่าเริ่มต้นของ TimeOutCnt ให้มีค่าเท่ากับศูนย์ ซึ่งทำหน้าที่เป็นตัวนับความผิดพลาดที่เกิดขึ้นอันเนื่องจากเวลาในการตอบสนอง

(2) กำหนด port ที่ใช้ให้เป็น CommPort1

(3) เปิด port Comm1

(4) กำหนดค่าเริ่มต้นในการรับ/ส่งข้อมูลให้มีค่าอัตราเร็วเป็น 9600 bps ไม่มี parity bit ขนาดของข้อมูล 8 บิต และมี 1 stop bit

(5) กำหนดค่าเริ่มต้นของ DataErrCnt ให้มีค่าเท่ากับศูนย์ศูนย์ ซึ่งทำหน้าที่เป็นตัวนับความผิดพลาดที่เกิดขึ้นอันเนื่องจากข้อมูลหรือตรรกะ

(6) กำหนดค่าเริ่มต้นของ CountTx ให้มีค่าเท่ากับศูนย์ศูนย์ ซึ่งทำหน้าที่เป็นตัวนับจำนวนครั้งที่มีการส่งข้อมูล

```
Private Sub Timer1_Timer()
   Timer2.Enabled = True

   Str_in = ""                          ' clear receive buffer

   CountTx = CountTx + 1
   Text10.Text = CountTx
   MSComm1.Output =  Chr(1) & Chr(3) & Chr(0) & Chr(2) & Chr(0) & Chr(1) &
                     Chr(&H25) & Chr(&HCA)                              (7)

End Sub
```

ภาพประกอบ จ-2 Timer1_Timer

Timer1_Timer เป็นตัวกำหนดเวลาสำหรับส่งข้อมูลไปยัง slave โดยรูปแบบข้อมูลที่ส่งแสดงดัง (7)

Timer2 เป็น object ที่ใช้ในการจับเวลาในการรอการตอบกลับจาก slave ถ้า slave ไม่สามารถตอบกลับได้ทันเวลาที่ตั้งไว้ใน Timer2 ก็จะมีการเรียกใช้ฟังก์ชัน Timer2_Timer เพื่อเพิ่มค่าตัวนับความผิดพลาดอันเนื่องจากเวลาการตอบสนอง

```
Private Sub Timer2_Timer()
   TimeOutCnt = TimeOutCnt + 1
   Text1.Text = TimeOutCnt
   Timer2.Enabled = False
End Sub
```

ภาพประกอบ จ-3 Timer2_Timer

```
Private Sub ReadModbus()
   Dim ch As Variant
      In_Cnt = In_Cnt + 1            ' increment receive counter
      ch = MSComm1.Input             ' read in character form serial port
      Str_in = Str_in + ch

     If In_Cnt = 7 Then              ' receive the whole packet
        Tempt = Asc(Mid$(Str_in, 5, 1)) + ((Asc(Mid$(Str_in, 4, 1))) * 256)
        TankTempt = CDbl(Tempt) / 10
        ControlPanel.TemptT.Text = TankTempt
        CheckErr                                    ' check error in receive packet
        Timer2.Enabled = False
      End If
End Sub
```

ภาพประกอบ จ-4 ReadModbus

ReadModbus ทำหน้าที่รับข้อมูลที่ตอบกลับมาจาก slave และตรวจสอบความถูกต้องของข้อมูลที่ได้รับโดยการเรียกฟังก์ชัน CheckErr

```
Private Sub CheckErr()
   Dim crc As Long
   Dim str_In6 As Integer
   Dim str_In7 As Integer
   Dim crc_HO As Integer
   Dim crc_LO As Integer

   str_In6 = Asc(Mid$(Str_in, 6, 1))                'crc in (HO)
   str_In7 = Asc(Mid$(Str_in, 7, 1))                'crc in (LO)
   crc = mbus_CalcCRCString(Str_in, 5)

   Text8.Text = str_In6
   Text9.Text = str_In7

   crc_HO = crc And 255
   crc_LO = Right$((crc \ &H100), 6)

   Text6.Text = crc_HO
   Text7.Text = crc_LO

   If ((crc_HO <> str_In6) Or (crc_LO <> str_In7)) Then
         DataErrCnt = DataErrCnt + 1
         Text4.Text = DataErrCnt
   End If
End Sub
```

ภาพประกอบ จ-5 CheckErr

CheckErr เป็นฟังก์ชันที่ทำหน้าที่ตรวจสอบความถูกต้องของข้อมูลด้วยวิธี CRC ทั้ง 2 ไบต์โดยมีการเรียกใช้ฟังก์ชัน mbus_CalcCRCString เพื่อคำนวณค่า CRC

```
' Given a String, Calc a modbus style CRC-16 by look-up table
Function mbus_CalcCRCString(sBuf As String, Length As Integer) As Long
  Dim x As Long
  Dim crc As Long
  Dim i As Integer

  If (usCRC(1) <> &HC0C1) Then
    mbus_FillCRCTable                              (8)
  End If

  crc = 65535                ' start with all 1's for a reverse CRC

  For i = 1 To Length        ' process each character in the message */
    x = Asc(Mid(sBuf, i, 1))   ' get next char
    x = (crc Xor x) And 255    '
    x = usCRC(x) And 65535     '
    crc = (crc \ 256) Xor x
  Next i
  mbus_CalcCRCString = crc
End Function
```

ภาพประกอบ จ-6 mbus_CalcCRCString

(8) เป็นการนำค่าจาก look-up table มาใช้ในการคำนวณ CRC ซึ่ง look-up table มีค่าดังนี้

```
Sub mbus_FillCRCTable()
  ' This is a very silly function, but I cannot think
  ' of a better way to init this array (other than .DLL)
  ' VB doesn't have the DATA/READ statement used by earlier BASIC

Debug.Print "Filling CRC Table"

  usCRC(0) = &H0: usCRC(1) = &HC0C1: usCRC(2) = &HC181: usCRC(3) = &H140
  usCRC(4) = &HC301: usCRC(5) = &H3C0: usCRC(6) = &H280: usCRC(7) = &HC241
  usCRC(8) = &HC601: usCRC(9) = &H6C0: usCRC(10) = &H780: usCRC(11) = &HC741
  usCRC(12) = &H500: usCRC(13) = &HC5C1: usCRC(14) = &HC481: usCRC(15) = &H440
  usCRC(16) = &HCC01: usCRC(17) = &HCC0: usCRC(18) = &HD80: usCRC(19) = &HCD41
  usCRC(20) = &HF00: usCRC(21) = &HCFC1: usCRC(22) = &HCE81: usCRC(23) = &HE40
  usCRC(24) = &HA00: usCRC(25) = &HCAC1: usCRC(26) = &HCB81: usCRC(27) = &HB40
  usCRC(28) = &HC901: usCRC(29) = &H9C0: usCRC(30) = &H880: usCRC(31) = &HC841
  usCRC(32) = &HD801: usCRC(33) = &H18C0: usCRC(34) = &H1980: usCRC(35) = &HD941
  usCRC(36) = &H1B00: usCRC(37) = &HDBC1: usCRC(38) = &HDA81: usCRC(39) = &H1A40
  usCRC(40) = &H1E00: usCRC(41) = &HDEC1: usCRC(42) = &HDF81: usCRC(43) = &H1F40
  usCRC(44) = &HDD01: usCRC(45) = &H1DC0: usCRC(46) = &H1C80: usCRC(47) = &HDC41
  usCRC(48) = &H1400: usCRC(49) = &HD4C1: usCRC(50) = &HD581: usCRC(51) = &H1540
  usCRC(52) = &HD701: usCRC(53) = &H17C0: usCRC(54) = &H1680: usCRC(55) = &HD641
  usCRC(56) = &HD201: usCRC(57) = &H12C0: usCRC(58) = &H1380: usCRC(59) = &HD341
  usCRC(60) = &H1100: usCRC(61) = &HD1C1: usCRC(62) = &HD081: usCRC(63) = &H1040
  usCRC(64) = &HF001: usCRC(65) = &H30C0: usCRC(66) = &H3180: usCRC(67) = &HF141
  usCRC(68) = &H3300: usCRC(69) = &HF3C1: usCRC(70) = &HF281: usCRC(71) = &H3240
  usCRC(72) = &H3600: usCRC(73) = &HF6C1: usCRC(74) = &HF781: usCRC(75) = &H3740
  usCRC(76) = &HF501: usCRC(77) = &H35C0: usCRC(78) = &H3480: usCRC(79) = &HF441
  usCRC(80) = &H3C00: usCRC(81) = &HFCC1: usCRC(82) = &HFD81: usCRC(83) = &H3D40
  usCRC(84) = &HFF01: usCRC(85) = &H3FC0: usCRC(86) = &H3E80: usCRC(87) = &HFE41
  usCRC(88) = &HFA01: usCRC(89) = &H3AC0: usCRC(90) = &H3B80: usCRC(91) = &HFB41
```

ภาพประกอบ จ-7 mbus_FillCRCTable

```
usCRC(92) = &H3900: usCRC(93) = &HF9C1: usCRC(94) = &HF881: usCRC(95) = &H3840
usCRC(96) = &H2800: usCRC(97) = &HE8C1: usCRC(98) = &HE981: usCRC(99) = &H2940
usCRC(100) = &HEB01: usCRC(101) = &H2BC0: usCRC(102) = &H2A80: usCRC(103) = &HEA41
usCRC(104) = &HEE01: usCRC(105) = &H2EC0: usCRC(106) = &H2F80: usCRC(107) = &HEF41
usCRC(108) = &H2D00: usCRC(109) = &HEDC1: usCRC(110) = &HEC81: usCRC(111) = &H2C40
usCRC(112) = &HE401: usCRC(113) = &H24C0: usCRC(114) = &H2580: usCRC(115) = &HE541
usCRC(116) = &H2700: usCRC(117) = &HE7C1: usCRC(118) = &HE681: usCRC(119) = &H2640
usCRC(120) = &H2200: usCRC(121) = &HE2C1: usCRC(122) = &HE381: usCRC(123) = &H2340
usCRC(124) = &HE101: usCRC(125) = &H21C0: usCRC(126) = &H2080: usCRC(127) = &HE041
usCRC(128) = &HA001: usCRC(129) = &H60C0: usCRC(130) = &H6180: usCRC(131) = &HA141
usCRC(132) = &H6300: usCRC(133) = &HA3C1: usCRC(134) = &HA281: usCRC(135) = &H6240
usCRC(136) = &H6600: usCRC(137) = &HA6C1: usCRC(138) = &HA781: usCRC(139) = &H6740
usCRC(140) = &HA501: usCRC(141) = &H65C0: usCRC(142) = &H6480: usCRC(143) = &HA441
usCRC(144) = &H6C00: usCRC(145) = &HACC1: usCRC(146) = &HAD81: usCRC(147) = &H6D40
usCRC(148) = &HAF01: usCRC(149) = &H6FC0: usCRC(150) = &H6E80: usCRC(151) = &HAE41
usCRC(152) = &HAA01: usCRC(153) = &H6AC0: usCRC(154) = &H6B80: usCRC(155) = &HAB41
usCRC(156) = &H6900: usCRC(157) = &HA9C1: usCRC(158) = &HA881: usCRC(159) = &H6840
usCRC(160) = &H7800: usCRC(161) = &HB8C1: usCRC(162) = &HB981: usCRC(163) = &H7940
usCRC(164) = &HBB01: usCRC(165) = &H7BC0: usCRC(166) = &H7A80: usCRC(167) = &HBA41
usCRC(168) = &HBE01: usCRC(169) = &H7EC0: usCRC(170) = &H7F80: usCRC(171) = &HBF41
usCRC(172) = &H7D00: usCRC(173) = &HBDC1: usCRC(174) = &HBC81: usCRC(175) = &H7C40
usCRC(176) = &HB401: usCRC(177) = &H74C0: usCRC(178) = &H7580: usCRC(179) = &HB541
usCRC(180) = &H7700: usCRC(181) = &HB7C1: usCRC(182) = &HB681: usCRC(183) = &H7640
usCRC(184) = &H7200: usCRC(185) = &HB2C1: usCRC(186) = &HB381: usCRC(187) = &H7340
usCRC(188) = &HB101: usCRC(189) = &H71C0: usCRC(190) = &H7080: usCRC(191) = &HB041
usCRC(192) = &H5000: usCRC(193) = &H90C1: usCRC(194) = &H9181: usCRC(195) = &H5140
usCRC(196) = &H9301: usCRC(197) = &H53C0: usCRC(198) = &H5280: usCRC(199) = &H9241
```

ภาพประกอบ จ-7 (ต่อ)

```
usCRC(200) = &H9601: usCRC(201) = &H56C0: usCRC(202) = &H5780: usCRC(203) = &H9741
usCRC(204) = &H5500: usCRC(205) = &H95C1: usCRC(206) = &H9481: usCRC(207) = &H5440
usCRC(208) = &H9C01: usCRC(209) = &H5CC0: usCRC(210) = &H5D80: usCRC(211) = &H9D41
usCRC(212) = &H5F00: usCRC(213) = &H9FC1: usCRC(214) = &H9E81: usCRC(215) = &H5E40
usCRC(216) = &H5A00: usCRC(217) = &H9AC1: usCRC(218) = &H9B81: usCRC(219) = &H5B40
usCRC(220) = &H9901: usCRC(221) = &H59C0: usCRC(222) = &H5880: usCRC(223) = &H9841
usCRC(224) = &H8801: usCRC(225) = &H48C0: usCRC(226) = &H4980: usCRC(227) = &H8941
usCRC(228) = &H4B00: usCRC(229) = &H8BC1: usCRC(230) = &H8A81: usCRC(231) = &H4A40
usCRC(232) = &H4E00: usCRC(233) = &H8EC1: usCRC(234) = &H8F81: usCRC(235) = &H4F40
usCRC(236) = &H8D01: usCRC(237) = &H4DC0: usCRC(238) = &H4C80: usCRC(239) = &H8C41
usCRC(240) = &H4400: usCRC(241) = &H84C1: usCRC(242) = &H8581: usCRC(243) = &H4540
usCRC(244) = &H8701: usCRC(245) = &H47C0: usCRC(246) = &H4680: usCRC(247) = &H8641
usCRC(248) = &H8201: usCRC(249) = &H42C0: usCRC(250) = &H4380: usCRC(251) = &H8341
```

ภาพประกอบ จ-7 (ต่อ)