ระบบการจัดการให้บริการเครื่องใช้ไฟฟ้าแบบหยอดเหรียญ

Multiple Management for Vending Machines

โดย

ผศ.บุณย์ชนะ ภู่ระหงษ์

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีงบประมาณเงินรายได้ 2553

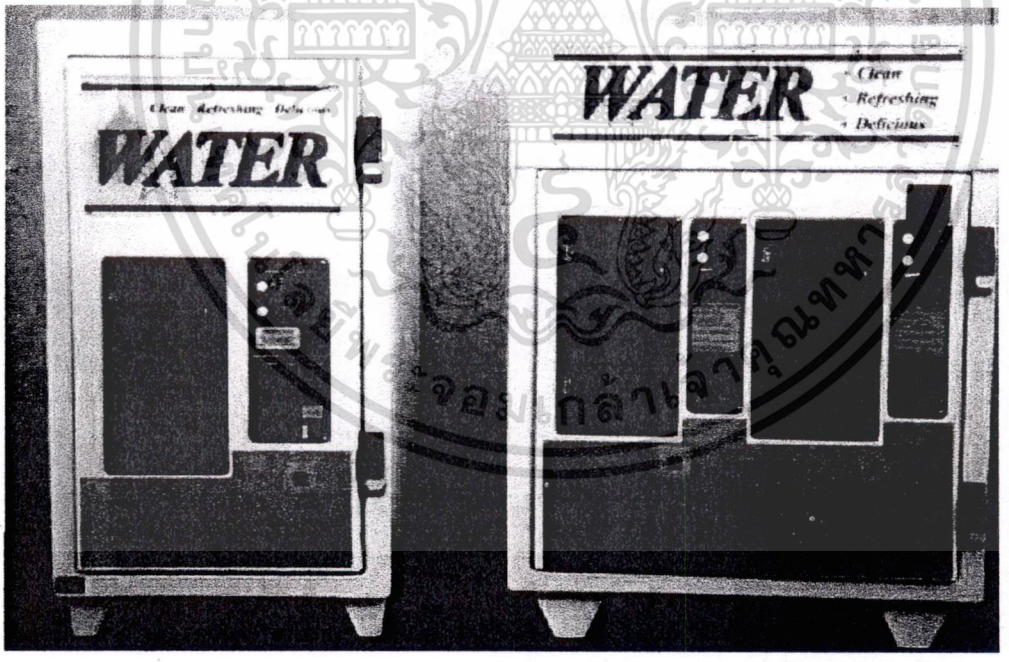
บทที่ 1
บทนำ

1. ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันธุรกิจเครื่องหยอดเหรียญได้รับความนิยมเพิ่มมากขึ้น มีผู้ประกอบการธุรกิจหลายรายได้นำอุปกรณ์ไฟฟ้าที่ใช้ในชีวิตประจำวัน มารวมกับเครื่องหยอดเหรียญเพื่อเปิดให้บริการกับผู้ใช้ในสถานที่ต่างๆเช่น โรงแรม หอพัก แหล่งชุมชน เป็นต้น ในส่วนของเครื่องใช้ไฟฟ้าที่นิยมมาประยุกต์ใช้กับเครื่องหยอดเหรียญมีดังนี้

<u>ตู้น้ำดื่มหยอดเหรียญ</u>

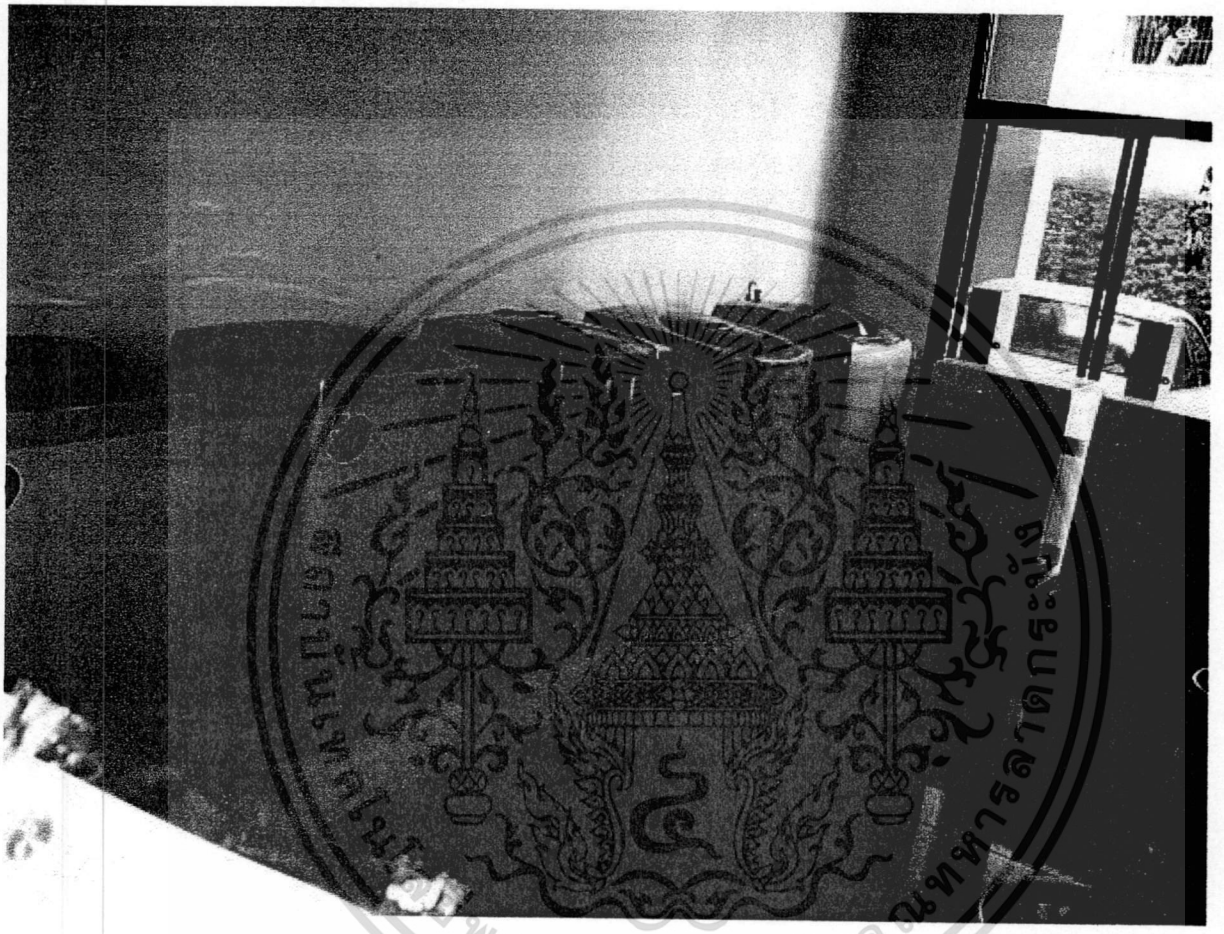
เป็นการให้บริการโดยการนำเอาเครื่องกรองน้ำ เครื่องจ่ายน้ำ และเครื่องหยอดเหรียญ มารวมกัน ลักษณะการให้บริการคือ ผู้ใช้ที่ต้องการใช้น้ำจะนำภาชนะมารองน้ำ แล้วทำการหยอดเหรียญ เครื่องจะทำการกรองน้ำและจ่ายน้ำให้กับผู้ใช้ตามจำนวนเงินที่ผู้ใช้หยอดให้กับหยอดเหรียญ



รูปที่ 1.1 ตู้น้ำดื่มหยอดเหรียญ (ที่มา www.anunbeatableprice.com)

<u>เครื่องซักผ้าหยอดเหรียญ</u>

ให้บริการโดย ผู้ใช้หยอดเหรียญให้กับเครื่อง แล้วเครื่องซักผ้าจะเริ่มต้นกระบวนการซักผ้าอย่างอัตโนมัติตั้งแต่การ ใส่น้ำ ซักผ้า แช่ผ้า และปั่นแห้ง



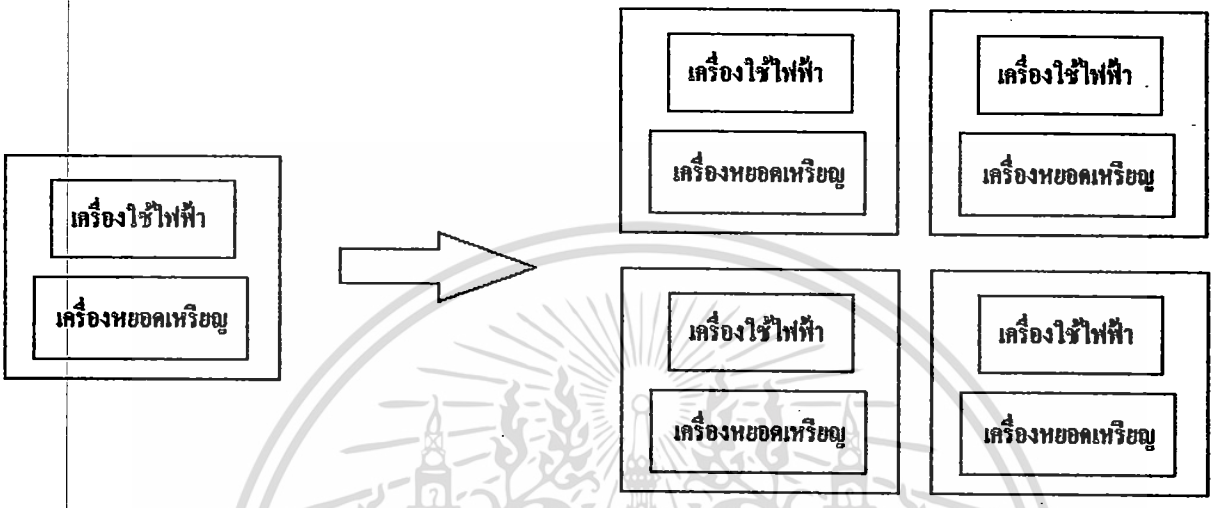
รูปที่ 1.2 เครื่องซักผ้าหยอดเหรียญ

ปัญหา

จากภาพรวมพบว่าในการใช้เครื่องหยอดเหรียญ มาประยุกต์ใช้กับ เครื่องใช้ไฟฟ้าและเครื่องอำนวยความสะดวกต่างๆ มีข้อจำกัดอยู่ คือ

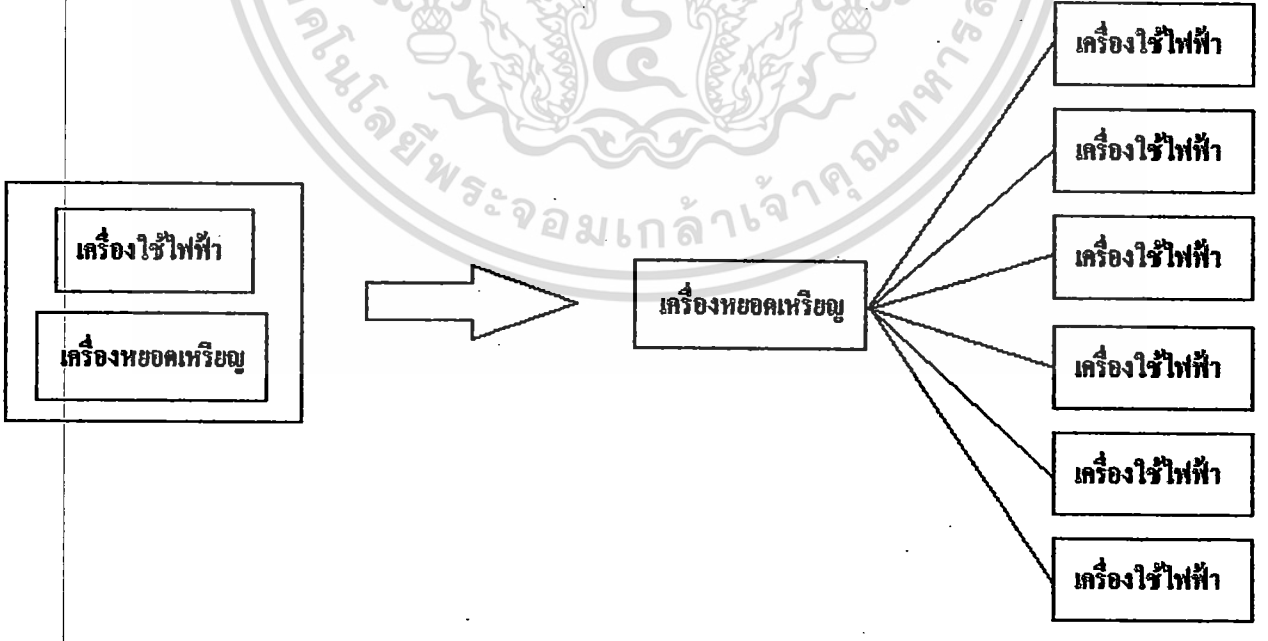
หากต้องการเพิ่มจำนวนของเครื่องที่ให้บริการ จำเป็นต้องเพิ่มจำนวนของเครื่องหยอดเหรียญไปด้วย ทำให้ต้นทุนสูงตามไปด้วย ทำอย่างไรจึงจะสามารถ ขยายจำนวนของเครื่องใช้ไฟฟ้าได้ โดยที่ไม่ต้องเพิ่มจำนวน

ของเครื่องหยอดเหรียญตาม ทำให้ต้นทุนที่ใช้ในการขยายการให้บริการลดลง ง่ายต่อการควบคุม และง่ายต่อการรักษาความปลอดภัยจากการโจรกรรม



รูปที่ 1.3 การขยายการให้บริการแบบปกติ

จากแนวคิดเกี่ยวการแก้ไขปัญหานี้ คือ ทำอย่างไรจึงสามารถควบคุมเครื่องใช้ไฟฟ้าหลายเครื่อง จากเครื่องหยอดเหรียญเครื่องเดียวได้



รูปที่ 1.4 แนวคิดเกี่ยวกับการแก้ไขปัญหา

นอกจากประโยชน์ในการลดต้นทุนการขยายอุปกรณ์แล้วข้อดีอีกข้อของแนวคิดนี้คือ เรื่องความปลอดภัยที่เพิ่มมากขึ้นเนื่องจาก เครื่องหยอดเหรียญแต่ละเครื่องจะเก็บเหรียญจากผู้ใช้ไว้ภายในเครื่องนั้นเอง ยิ่งจำนวนเครื่องหยอดเหรียญมีมาก ก็ยิ่งมีความเสี่ยงในการโจรกรรมมากขึ้น แต่จากแนวคิดใหม่นี้จำนวนเครื่องหยอดเหรียญในระบบจะเหลือเพียงเครื่องเดียวจึงง่ายต่อการรักษาความปลอดภัย

2 วัตถุประสงค์ในการวิจัย

2.1 เพื่อสะดวกต่อผู้รับบริการ

2.2 ทำการจัดการเพื่อต้นทุนของผู้ให้บริการ

3 ประโยชน์ที่คาดว่าจะได้รับ

เพื่อความสะดวก ปลอดภัย และค่าบริการทั้งผู้ให้และผู้รับบริการ โดยการนำความรู้ความสามารถที่นักวิจัยมี ทำวิจัยให้เกิดประโยชน์

บทที่ 2
ทฤษฎี

Microprocessor

ก่อนเริ่มการเขียนโปรแกรมบน ไมโครคอนโทรลเลอร์ และพัฒนาไปใช้กับระบบต่างๆนั้น ควรมีความรู้ความเข้าใจเกี่ยวกับเรื่องพื้นฐานของไมโครโปรเซสเซอร์ เช่นเรื่องของ รีจิสเตอร์ เมมโมรี่ และสถาปัตยกรรมภายในแบบต่าง ๆ เสียก่อน เพราะทำให้สามารถเข้าใจถึงหลักการทำงาน เข้าใจถึงโครงสร้างที่อยู่ภายใน รวมทั้งเข้าใจข้อดีข้อเสียของส่วนต่างๆ ได้

Register and Memory

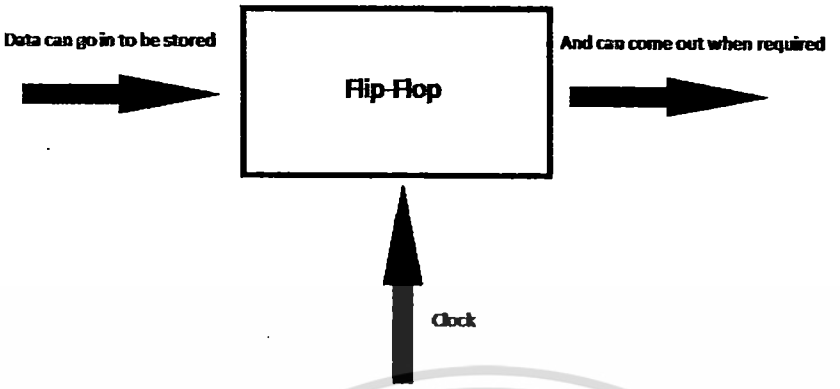
ภายใน Microprocessor ไม่ว่าจะรุ่นไหน ช่วงเวลาไหน หรือถูกผลิตมาจากบริษัทไหนก็ตาม ประกอบไปด้วยวงจรของ Logic Gates จำนวนมากมายมหาศาล เสมอ ซึ่งวงจรของ Logic Gates แต่ละอัน จะทำหน้าที่เก็บข้อมูล เลขฐานสอง จำนวน 1 หลัก วงจรของ Logic Gates ในแต่ละส่วนจะถูกเรียกว่า "Flip-Flop"

Flip-Flop or Bistable

Flip-Flop เป็นวงจรที่สามารถ เก็บข้อมูลเลขฐานสองจำนวน 1 หลักได้ หรืออาจอธิบายได้ว่า สามารถเก็บได้เพียง 0 กับ 1 เท่านั่น Flip-Flop จะมีเงื่อนไขในการทำงานอยู่ที่สัญญาณเวลา หรือ Clock โดยสัญญาณเวลานี้ จะเป็นตัวที่บอกว่าจะต้องเก็บข้อมูลตอนไหน

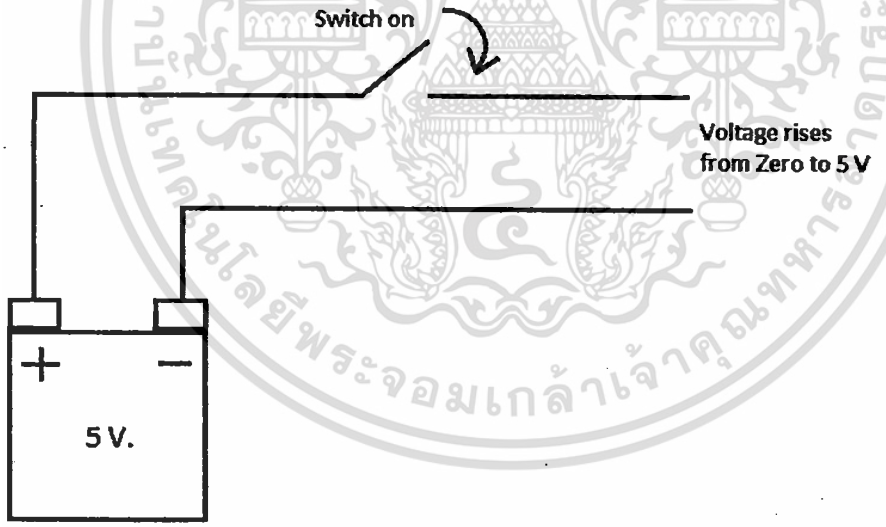
ขั้นตอนการทำงานคร่าวๆของFlip-Flop

1. เตรียมข้อมูลของเลขฐานสองที่ต้องการจะให้ Flip-Flop เก็บไว้
2. รอเวลาอยู่ช่วงประมาณหนึ่ง (ระดับนาโนวินาที) ให้ข้อมูลนั้นพร้อมที่จะถูกจัดเก็บ
3. ส่งสัญญาณเวลาเพื่อบอก Flip-Flop ให้เก็บข้อมูลที่ได้เตรียมไว้ ในส่วนของ Input

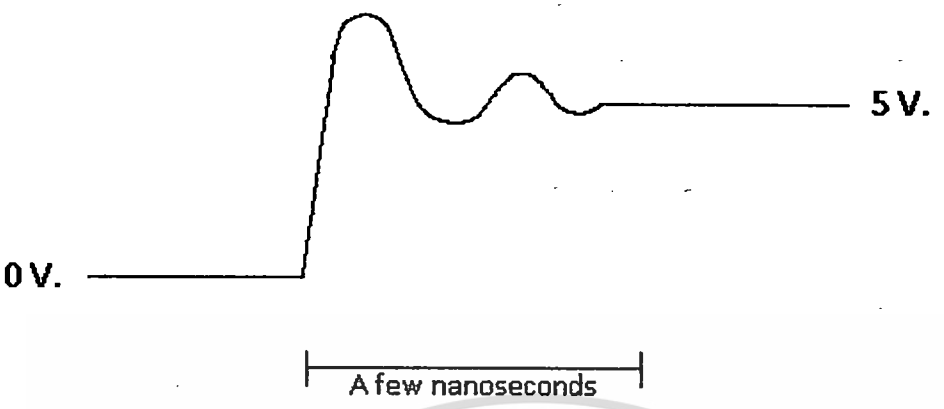


รูปที่ 2.1 ภาพรวมของการทำงานของ Flip-Flop

จากขั้นตอนการทำงานคร่าวๆของ Flip-Flop ที่อธิบายข้างต้นมานั้นในข้อที่สอง สาเหตุที่ต้องมีการรอเวลาช่วงเวลาหนึ่งนั้นก็เพราะว่า เมื่อสัญญาณมีการเปลี่ยนแปลง จาก Logic ต่ำ ไปเป็นอีก Logic หนึ่ง ซึ่งคือ Logic สูง จะมีช่วงเวลาหนึ่งที่ระดับของสัญญาณยังไม่เสถียรหรือยังไม่นิ่งนั่นเอง ช่วงเวลานี้จะใช้เวลาในระดับนาโนวินาที



รูปที่ 2.2 ภาพแสดงลักษณะของการเปิดสวิทช์

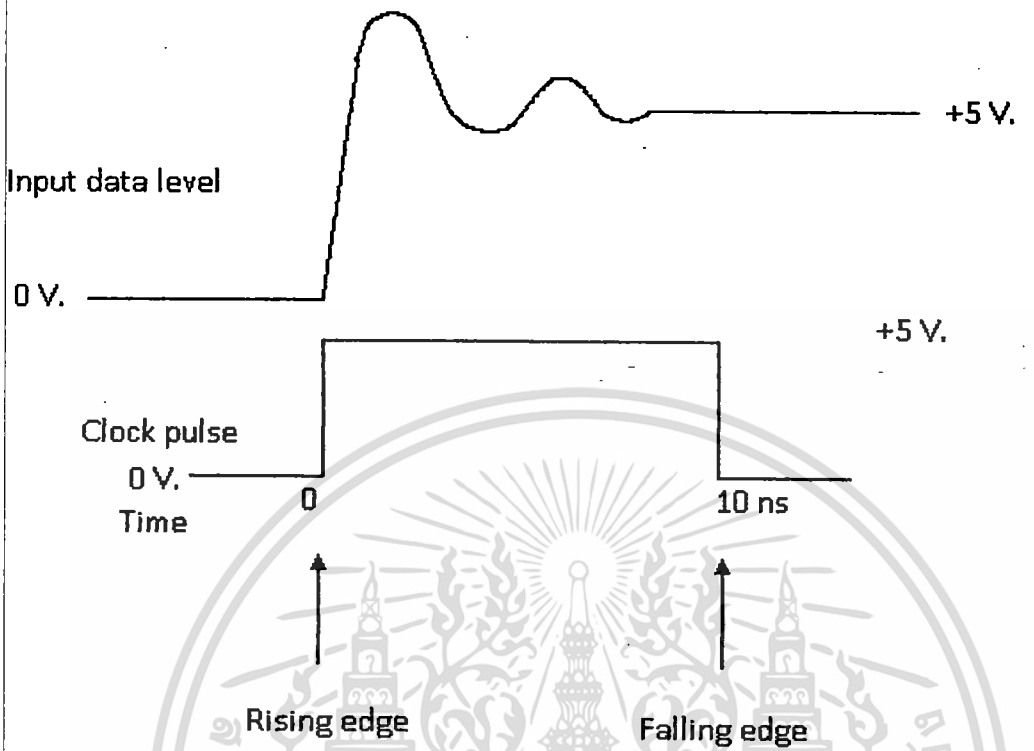


รูปที่ 2.3 ช่วงเวลาที่เกิดการเปลี่ยนแปลงของระดับสัญญาณ

Clock Signal

สัญญาณเวลาสำหรับ Flip-Flop นั้น ถือว่าเป็น Input ที่คอยบอกว่าจะต้องเริ่มการเก็บข้อมูลตอนไหน และ ถือว่าเป็นส่วนสำคัญส่วนหนึ่งสำหรับ Microprocessor เพราะคอยควบคุมจังหวะและลำดับขั้นตอนในการทำงาน ให้เป็นไปอย่างถูกต้อง

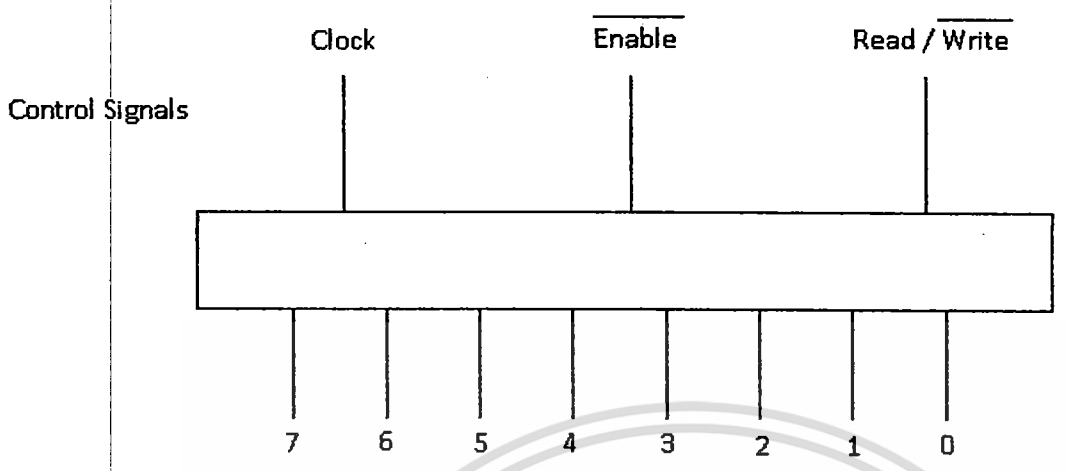
โดยส่วนมากแล้ว สัญญาณเวลาจะเป็นสัญญาณพัลส์ แบบ Positive Going Edge หรืออาจเรียกว่า Rising Edge



รูปที่ 2.4 การใช้สัญญาณเวลาควบคุมการทำงานของ Flip-Flop

A register

รีจิสเตอร์ (Register) นั้นเป็นการเอา ฟลิปฟล็อป หลายๆอันมารวมกัน ซึ่งฟลิปฟล็อปแต่ละอันสามมารถเก็บข้อมูลได้เพี่ยงบิทเดียวเท่านั้น ดังนั้นรีจิสเตอร์ 8 บิทหมายถึงการรวมเอา ฟลิปฟล็อป จำนวน 8 อันมาต่อกันทำให้สามารถเก็บข้อมูลขนาด 8 บิท ได้ รีจิสเตอร์นั้นแบ่งกลุ่มของการติดต่อได้ 2 ประเภท คือ สัญญาณควบคุม (Control Signals) ทำหน้าที่ควบคุมจังหวะการทำงานและทิศทางของการทำงาน ว่าจะอ่านข้อมูลหรือว่าเขียนข้อมูล และสัญญาณอีกประเภทคือสัญญาณข้อมูล (Data Signals)

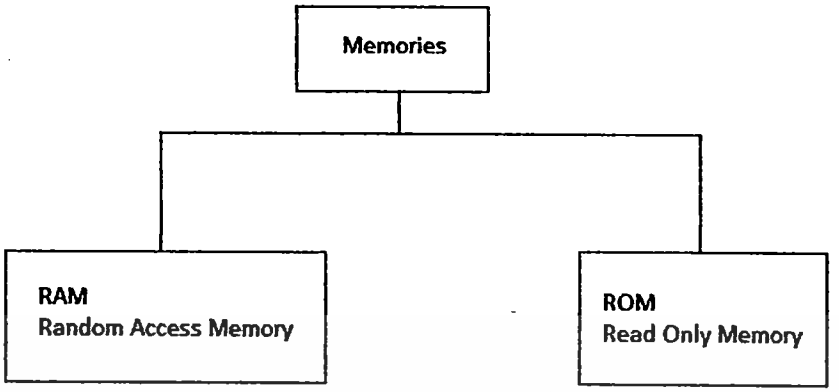


รูปที่ 2.5 รีจิสเตอร์ขนาด 8 บิต

ข้อมูลที่ถูกเก็บไว้ภายในรีจิสเตอร์จะยังคงอยู่ตราบเท่าที่ยังมีแหล่งจ่ายพลังงานไฟฟ้าอยู่ และจะหายไปเมื่อไม่มีแหล่งจ่ายพลังงานไฟฟ้าให้กับรีจิสเตอร์

หน่วยความจำ(Memories)

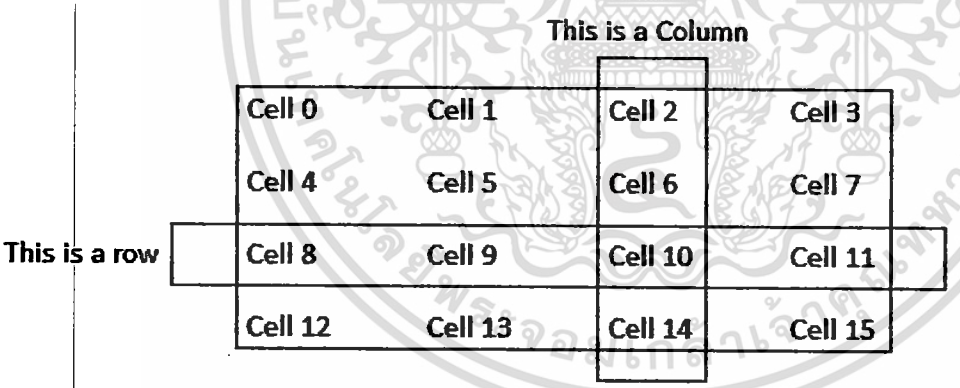
หน้าที่ของหน่วยความจำ คือการเก็บข้อมูลต่างๆ เช่นเดียวกับรีจิสเตอร์ ที่ทำหน้าที่เก็บข้อมูลเหมือนกัน หน่วยความจำกับรีจิสเตอร์มีความต่างกันตรงที่ รีจิสเตอร์ใช้สำหรับเก็บข้อมูลจำนวนน้อยแล้วนำไปใช้ได้ทันที แต่หน่วยความจำจะใช้เก็บข้อมูลจำนวนมากๆ หน่วยความจำบางประเภทสามารถรักษาข้อมูลไว้ได้แม้จะไม่มีแหล่งจ่ายพลังงานไฟฟ้าแล้วก็ตาม หน่วยความจำที่มีความสามารถนี้ เรียกว่าหน่วยความจำ รอม (ROM) หรือ หน่วยความจำลบเลือนไม่ได้ (Non-Volatile Memory) ส่วนอีกประเภทหนึ่งคือ เมื่อไม่มีแหล่งจ่างพลังงานไฟฟ้า ข้อมูลที่ถูกเก็บไว้ก็จะหายไป เรียกหน่วยความจำประเภทนี้ว่า หน่วยความจำ แรม (RAM) หรือ หน่วยความจำลบเลือนได้ (Volatile Memory)



รูปที่ 2.6 ประเภทของหน่วยความจำ

แรม(RAM)

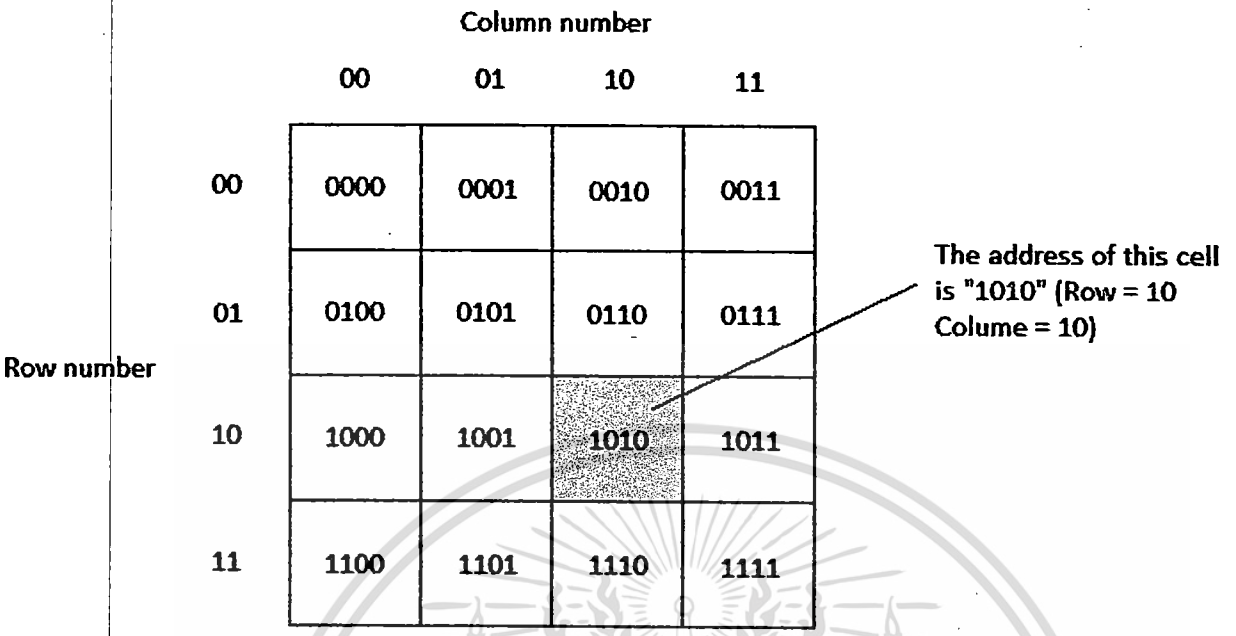
หน่วยความจำเข้าถึงแบบสุ่ม(Random access memories) เป็นชื่อเต็มๆของแรม หมายถึง หน่วยความจำชนิดนี้สามารถเข้าถึงได้โดยไม่ต้องอาศัยลำดับในการเข้าถึง ตำแหน่งที่ว่านี้คือตำแหน่งของรีจิสเตอร์ ที่อยู่ภายในนั่นเอง เพราะในแรมมีรีจิสเตอร์อยู่ภายในเป็นจำนวนมาก การที่จะระบุได้ว่าต้องการเข้าถึงรีจิสเตอร์ตัวไหนนั้น ต้องใช้การอ้างถึงตำแหน่งเลขที่อยู่ (Address)



รูปที่ 2.7 การวางผัง (Layout) ของเซลล์ (Cell) ที่อยู่ในหน่วยความจำ

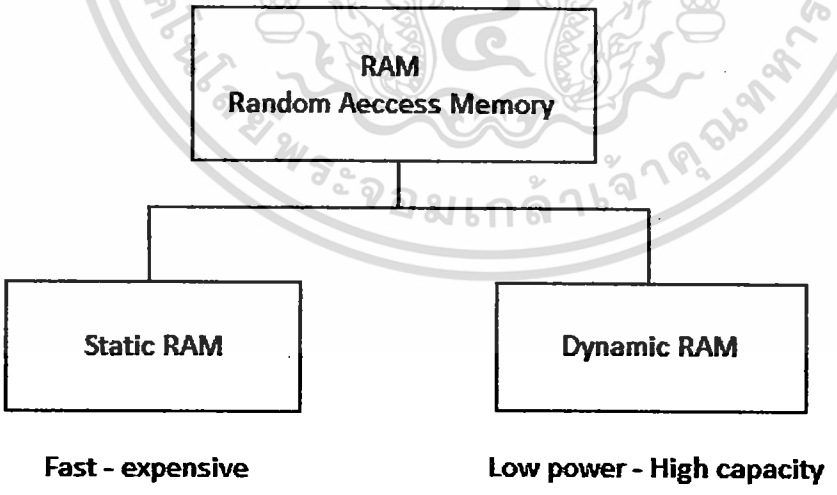
การเข้าถึงหน่วยความจำ

ในแต่ละตำแหน่งของหน่วยความจำจะมีเลขที่คอยบอกตำแหน่งนั้นๆอยู่ เรียกว่า แอดเดรส (Address)



รูปที่ 2.8 การเลือกใช้ตำแหน่งของหน่วยความจำ

หน่วยความจำแบบแรมนั้นแบ่งได้เป็นสองประเภทที่แตกต่างกัน คือ Static RAM (SRAM) และ Dynamic RAM (DRAM)



รูปที่ 2.9 แผนผังประเภทของแรมทั้ง 2 ประเภท

แรมพลวัต Dynamic RAM

ใช้วิธีการเก็บข้อมูลลงในตัวเก็บประจุ ที่เป็นส่วนประกอบเล็กๆในประจุไฟฟ้า ในรูปของไฟฟ้าสถิต ในความเป็นจริง ไฟฟ้าที่ถูกเก็บอยู่ในตัวเก็บประจุนั้น จะรั่วไหลออกมาตลอดเวลาเนื่องมาจากความไม่สมบรูณ์หรือบกพร่องของฉนวนนั้นเอง ดังนั้น หลังจากประจุที่ทำหน้าที่บันทึกข้อมูล กำลังจะรั่วไหลออกไปหมด ซึ่งส่งผลให้ DRAM นั้นจะว่างเปล่าหรือเสมือนว่าไม่ได้มีข้อมูลอยู่ภายในเลย และจะทำให้ข้อมูลที่เก็บอยู่ทั้งหมดหายไป. จึงต้องมีการเรียกข้อมูลซ้ำเข้ามาเก็บในตัวเก็บประจุใหม่ ซึ่งเรียกว่าเรียกว่า refreshing และจะเกิดขึ้นในช่วงเวลา 2 ms โดยวงจรควบคุมกระแสของ DRAM. เพื่อป้องกันไม่ให้ระบบถูกรบกวนจากการเรียกข้อมูลใหม่ในไมโครโปรเซสเซอร์ การเรียกข้อมูลซ้ำจะต้องกระทำหลังจากที่ DRAM หยุดการทำงานแล้ว หรือหมายความว่าการเรียกข้อมูลซ้ำไม่สามารถทำในขณะที่ DRAM กำลังเขียนหรืออ่านข้อมูลอยู่ได้

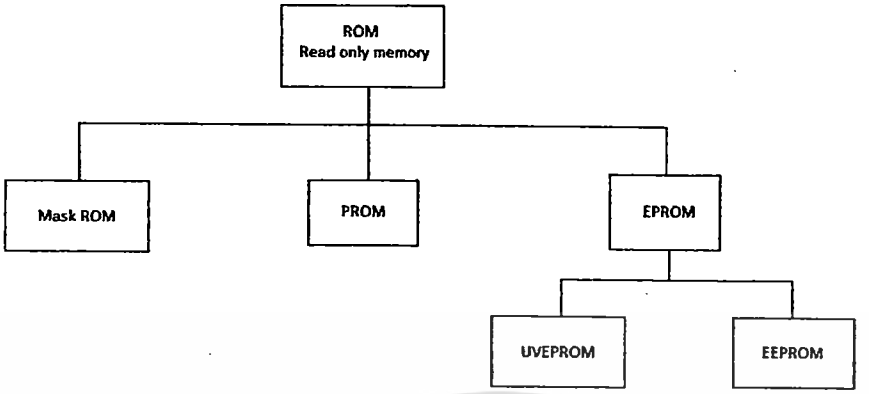
ในขณะที่ข้อมูลถูกเก็บอยู่ในตัวเก็บประจุ ข้อมูลจะยังคงอยู่ได้โดยไม่ต้องการกระแสไฟฟ้า ยกเว้นในการเรียกข้อมูลซ้ำเท่านั้น ทำให้เกิดความร้อนน้อยและสามารถจัดเก็บข้อมูลได้เป็นจำนวนมาก

แรมสถิต Static RAM

ภายในแรมชนิดนี้จะมีวงจรฟลิปฟล็อปอยู่ภายใน ทำให้สามารถเขียนและอ่านข้อมูลได้อย่างรวดเร็ว แต่เนื้อที่ในการเก็บข้อมูลจะน้อยกว่า DRAM ด้วยความเร็วของมันจึงนำมาใช้ในการเข้าถึงข้อมูลที่ต้องใช้ความเร็วเป็นสำคัญ

ประเภทของรอม (ROM)

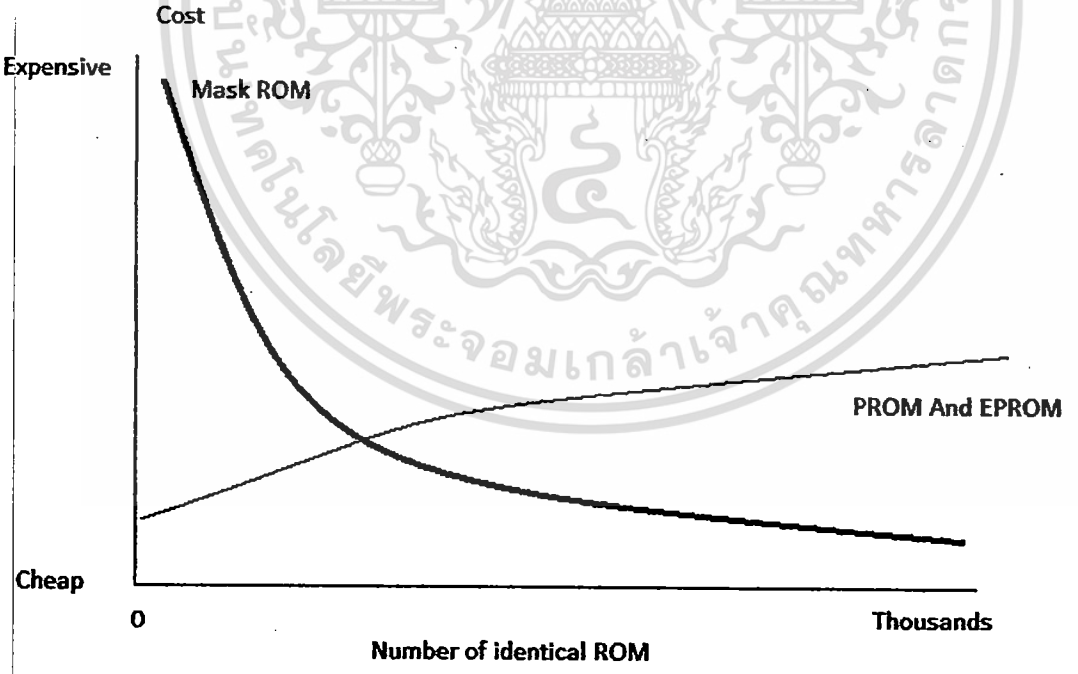
หลังจากที่ได้กล่าวถึงประเภทของแรมไปแล้วในข้างต้น ในหัวข้อนี้จะกล่าวถึงประเภทของรอม รอมทุกประเภทนั้น จะถูกใช้ในการเก็บข้อมูลต่างๆ แต่ลักษณะการเก็บจะเป็นการเก็บแบบ ถาวร คือ ถึงจะไม่ได้จ่ายกระแสไฟฟ้าให้กับรอมแล้ว ข้อมูลที่อยู่ภายในรอมจะยังคงอยู่ ไม่สูญหายไปเหมือนแรม การใช้งานรอมนั้น ผู้ใช้สามารถอ่านข้อมูลจากรอมได้เพียงอย่างเดียว แต่ไม่สามารถเขียนข้อมูลลงไปได้



รูปที่ 2.10 แผนผังประเภทของรอม

Mask ROM

หน่วยความจำประเภทนี้ ข้อมูลทั้งหมดที่อยู่ภายในจะถูกโปรแกรมมาจากโรงงานตั้งแต่ขั้นตอนการผลิตไอซี เราจะใช้ ROM ชนิดนี้ เมื่อข้อมูลนั้นไม่มีการเปลี่ยนแปลง และเหมาะสำหรับงานที่ผลิตครั้งละมากๆเพราะในการผลิตจะเสียค่าใช้จ่ายสูงถ้าผลิตออกมาจำนวนน้อยแต่จะมีราคาลดลงเมื่อจำนวนของ ROM ที่ผลิตมีมากขึ้น ผู้ใช้ ไม่สามารถ เปลี่ยนแปลงข้อมูลภายใน ROM ได้



รูปที่ 2.11 ค่าใช้จ่ายและจำนวนของ ROM ในการผลิต

PROM

PROM ย่อมาจาก Programmable Read Only Memory คือ หน่วยความจำชนิดหนึ่ง (ROM) ซึ่งสามารถโปรแกรมได้เพียงครั้งเดียว โดยใช้เครื่องมือที่เรียกว่า PROM Programmer ป้อนพัลส์แรงดันสูง (High Voltage Pulsed) ทำให้ Metal Strips , Polycrystaline ที่อยู่ข้างบนตัวของ IC ขาดออก จึงทำให้เกิดลอจิกเป็นลอจิก "1" และ ลอจิก "0" ขึ้น เรียกกระบวนการนี้ว่า " Burning PROM " กระบวนการนี้มีข้อจำกัด คือ จะต้องไม่เกิดข้อผิดพลาดใด ๆ ก็ตามในการ Burning PROM เลย เพราะสามารถโปรแกรมลงได้เพียงแค่ครั้งเดียวเท่านั้น

EPROM

เป็นหน่วยความจำที่มีลักษณะคล้าย ๆ PROM แต่มีคุณสมบัติที่เหนือกว่า PROM กล่าวคือ EPROM สามารถที่จะโปรแกรมข้อมูลลงไป เปลี่ยนแปลงข้อมูล หรือ ลบข้อมูลนั้นได้เรื่อย ๆ ซึ่งจะแตกต่างกับ PROM ซึ่งจะสามารถโปรแกรมข้อมูลได้เพียงครั้งเดียวเท่านั้น

EPROM แบ่งออกเป็น 2 ประเภทตามวิธีการลบข้อมูล

1.UVPROM เป็นประเภทที่ลบด้วยรังสีอัลตราไวโอเลต โดยการใช้แสงอัลตราไวโอเลตส่องลงบน IC โดยผ่านทางกระจกใส ๆ ที่ทำมาจากผลึกควอตซ์ เมื่อฉายแสงไปสักพัก ข้อมูลต่าง ๆ ที่ก็ถูกลบ จากนั้นก็สามารถนำไปทำการโปรแกรมใหม่ใหม่ได้อีก เมื่อเราโปรแกรมเสร็จเรียบร้อยแล้ว ควรหาวัสดุที่ทึบแสงมาปิดทับกระจกใสๆ เพื่อป้องกันไม่ให้แสงจากสิ่งแวดล้อมเข้ามาลบข้อมูลใน EPROM ให้สูญหายได้ เช่น แสงจากหลอดไฟ ฟลูออเรตเซนต์ เป็นต้น

2.EEPROM เป็นประเภทที่ลบด้วยกระแสไฟฟ้า ซึ่งจะรวดเร็ว และสะดวกกว่าแบบ UV PROM มาก แต่ข้อเสียของ EEPROM คือความเร็วในการอ่านข้อมูล และ เขียนข้อมูล ไม่ค่อยเร็วเท่าไหร่นัก

Reduced Instruction Set Computer (RISC)

เป็นไมโครโพรเซสเซอร์ชนิดหนึ่งที่มีชุดคำสั่งจำนวนไม่มากนัก คอมพิวเตอร์แบบ RISC สามารถกระทำการตามคำสั่งได้อย่างรวดเร็วมากเพราะคำสั่งจะสั้น ชิพแบบ RISC ยังผลิตได้ง่าย เพราะใช้จำนวนทรานซิสเตอร์ น้อยกว่า ตัวอย่างชิพประเภทนี้ได้แก่ ARM, DEC Alpha, PA-RISC, SPARC, MIPS, และ PowerPC RISC เป็นสถาปัตยกรรมคอมพิวเตอร์ที่นิยมใช้ในปัจจุบัน และยังมีแนวโน้มจะเพิ่มมากขึ้นเป็นลำดับในอนาคต ทั้งนี้เพราะการใช้สถาปัตยกรรมแบบนี้เป็นวิธีหนึ่งที่ทำให้คอมพิวเตอร์

ทำงานเร็วขึ้น ปกติ หากพิจารณาจำนวนบิตที่เท่ากันระหว่างสถาปัตยกรรมคอมพิวเตอร์แบบ CISC กับ RISC แล้ว จะพบว่าคอมพิวเตอร์แบบ RISC จะเร็วกว่าแบบ CISC ประมาณ 3 เท่า

ด้านการใช้หน่วยความจำ

ข้อเสีย สถาปัตยกรรมแบบ RISC นั้นเน้นหลักการของการนำเอาชุดของคำสั่งง่ายๆ เพียงไม่กี่คำสั่ง (โดยทั่วไปไม่เกิน 128 คำสั่ง เช่น บวก,ลบ,คูณ,หาร) มาประกอบรวมเข้าไว้ด้วยกัน 128 คำสั่ง มีค่าเท่ากับ 2^6 หรือกล่าวคือใช้งานแค่ 6 BIT ในการเก็บค่าของชุดคำสั่ง ในการเก็บชุดคำสั่งจึง FIX CODE ไว้แค่ 6 เท่านั้น ซึ่งเกิด • ข้อเสียคือถ้าหากคำสั่งที่ใช้งานใช้แค่ 1 BIT ก็ยังคงเก็บ 6 BIT เกิด Waste Space

• ข้อดี เนื่องจากการเก็บข้อมูลของ RISC นี้เป็นลักษณะ FIX CODE จึงส่งผลให้การถอดรหัสรวดเร็ว เพราะชุดคำสั่งเท่ากันทุก Record

ด้านการปฏิบัติ

1. การทำงานจะทำได้เร็วกว่า CISC
2. เนื่องจากการเข้ารหัสของชุดคำสั่งเป็นลักษณะ FIX-ENCODING จึงง่ายต่อการ DECODE หรือถอดรหัส
3. ในสถาปัตยกรรมแบบ RISC มี REGISTER จำนวนมากจึงทำให้การทำงานโดยรวมรวดเร็ว
4. การใช้งานคำสั่งง่ายๆ ของ RISC นี้ บางคำสั่งใช้เวลา (CLOCK CYCLE) ไม่ถึง 1 CLOCK จึงส่งผลให้ทำงานได้รวดเร็ว

ด้านการสนับสนุนคอมไพเลอร์

ใน RISC นั้นมีคำสั่งประมาณ 128 คำสั่ง แนบมากับ CPU และอนุญาตให้ใช้งานคำสั่งประเภท LOAD/STORE ที่นำข้อมูลจาก MEMORY ไปกระทำกับ REGISTER โดยตรงซึ่งทำให้การทำงานโดยรวมเร็วกว่า จากจุดนี้เองในการใช้งานในส่วนของคำสั่งที่ซับซ้อน อาจต้องใช้คำสั่งในตัว Compiler มาใช้งานมากกว่า RISC เพราะ RISC เน้นหลักการทำงานของชุดคำสั่งที่ง่ายๆ แต่เร็ว ดังนั้นคำสั่งยากๆ จึงโยนให้เป็นหน้าที่ของตัว Compiler แทน

Complex Instruction Set Computer (CISC)

เป็นสถาปัตยกรรมที่มีคำสั่งใช้งานมากเป็นร้อยคำสั่ง เป็นสถาปัตยกรรมในยุคแรกของคอมพิวเตอร์ ในแต่ละคำสั่งจะใช้วงรอบในการประมวลผลไม่เท่ากัน ตามความซับซ้อนของการประมวลผล ข้อดีคือมีคำสั่งให้เลือกใช้มาก การพัฒนาด้วยภาษาแอสเซมบลี จะทำได้ง่ายกว่าสถาปัตยกรรมแบบ RISC

ด้านการใช้หน่วยความจำ

ข้อดี ในการใช้ของสถาปัตยกรรมแบบ CISC นั้น ชุดของคำสั่งจะมีมากมายหลายคำสั่งที่ ซับซ้อนและยุ่งยาก แต่นั่นไม่ได้หมายความว่า ทุกชุดคำสั่งจะมีการ FIX CODE กล่าว คือ ถ้ามีการใช้ชุดคำสั่งที่มีความซับซ้อนมากก็จะใช้จำนวน BIT มาก แต่ถ้าใช้งานชุดคำสั่งที่มีความซับซ้อนน้อยก็จะใช้งานจำนวน BIT น้อยเช่นกัน ในการเก็บชุดคำสั่งของ CISC นั้นจะเก็บเท่ากับจำนวนจริงของการใช้งาน จึงประหยัดเนื้อที่ใน Memory

ข้อเสีย เนื่องจากการเก็บชุดของคำสั่งนั้น เก็บเฉพาะการใช้งานจริง ซึ่งจะใช้งาน Memory น้อย แต่นั่นไม่ได้หมายความว่าจะทำให้เกิดประสิทธิภาพในการทำงานแต่ จะทำให้ประสิทธิภาพการทำงานของเครื่องคอมพิวเตอร์ ช้าลง เพราะต้องเสียเวลาในการถอดรหัสที่ซับซ้อนของการเข้ารหัสที่มีขนาดไม่เท่ากัน

ด้านการปฏิบัติ

1. เนื่องจาก CISC มีชุดของ Complex Instruction มากกว่า RISC และในคำสั่งพิเศษที่มีอยู่ใน CISC นั้น (หรือคำสั่งยากๆ) เช่น การทำ Polynomial ในการทำงานหนึ่งคำสั่งของ CISC อาจใช้เวลา (CLOCK) มากกว่าการนำเอาคำสั่งที่มีอยู่ใน RISC หลายๆ คำสั่งมารวมกันเสียอีก
2. ประสิทธิภาพอาจลดลงเนื่องจากเสียเวลาในการ DECODE เพราะชุดคำสั่งของ CISC ไม่แน่นอน มีทั้งสั้นและยาว อีกทั้งวงจรมีความสลับซับซ้อนมาก และใช้ CLOCK CYCLE นาน จึงทำให้เสียค่าใช้จ่ายสูง และใช้เวลานานกว่าในการประมวลผล

ด้านการสนับสนุนคอมไพเลอร์

ใน CISC มีชุดคำสั่งที่ซับซ้อนซึ่งติดมากับตัว CPU อยู่แล้ว แต่เมื่อมาทำการ CODE หรือเขียนโปรแกรมแล้วผ่านตัว Compiler หรือตัวแปลงจากโปรแกรมเป็นภาษาเครื่อง จะพบว่าคำสั่งยากๆ ที่มีอยู่ใน CPU นั้น ตัว Compiler กลับแปลงให้อยู่ในรูปของคำสั่งง่ายๆ หรือ กล่าวคือ Software ไม่ Support กับ Hardware ซึ่งใน CPU หรือ Hardware นั้นมีการรองรับการทำงานของชุดคำสั่งนี้ แต่ตัว Software ไม่ได้มีการใช้คุณสมบัติจากชุดของคำสั่งที่ติดมากับตัว CPU แต่อย่างใด ดังนั้นชุดคำสั่งที่บรรจุเอา Complex Instruction ไว้ใน CISC นั้น จะไม่ค่อย

มีประโยชน์มากนัก ถ้าหากว่าตัว Compiler นั้นไม่รองรับ และยิ่งไปกว่านั้นตัว Compiler บางตัวยังมีชุดคำสั่งยากๆ อยู่ในตัวมันแล้ว กลับยิ่งหมายความว่าจะไม่ได้ใช้งานจากสถาปัตยกรรมที่สร้างขึ้นมาใน CISC นี้เลย

ไมโครคอนโทรลเลอร์ AVR

คุณลักษณะโดยทั่วไป

มีประสิทธิภาพสูง แต่ใช้พลังงานน้อย

มีสถาปัตยกรรมภายในเป็นแบบ RISC

- มีคำสั่งที่มีประสิทธิภาพสูง 130 คำสั่ง โดยส่วนมาก คำสั่งนี้สามารถทำงานได้ใน 1 รอบสัญญาณเวลา
- มีรีจิสเตอร์ที่ใช้ในการสั่งงานทั่วๆไปแบบ 8 บิตจำนวน 32 ตัว
- มีความเร็วถึง 16 MIPS เมื่อใช้ความถี่ที่ 16 MHz
- มีความทนทานสูง และในส่วนของหน่วยความจำ ไม่มีการสูญหายของข้อมูล
- มีหน่วยความจำแบบ FLASH ขนาด 128K Bytes สามารถเขียนและลบได้ 10,000 ครั้ง
- มีหน่วยความจำแบบ EEPROM ขนาด 4K Bytes สามารถเขียนและลบได้ 100,000 ครั้ง
- มีหน่วยความจำแบบ SRAM ขนาด 4K Bytes
- สามารถเก็บรักษาข้อมูล ได้นาน 20 ปี ที่อุณหภูมิ 85 องศา หรือ 100 ปี ที่อุณหภูมิ 25 องศา
- มีโหมดป้องกันหน่อยความจำ

ความสามารถ

- มีไทเมอร์/เคาน์เตอร์ขนาด 8 บิตพร้อมปรีสเกลเลอร์ จำนวน 2 ตัว
- มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิตพร้อมปรีสเกลเลอร์จำนวน 2 ตัว
- มีเคาน์เตอร์ แบบเรียบไทม์
- มีPWM(Pulse Width Modulation) จำนวน 3 ตัว
- มีโมดูลการแปลงสัญญาณ อนาล็อก เป็นสัญญาณดิจิตอล ขนาด 10 บิต จำนวน 6 ตัว
- โมดูลการสื่อสารข้อมูลดิจิตอลแบบอะซิงโครนัส (UART)
- โมดูลการสื่อสารข้อมูลดิจิตอลแบบซิงโครนัส (SPI)
- โมดูลตรวจจับการทำงานที่ผิดพลาดของ CPU (WATCHDOG TIMER)
- ระบบการรีเซ็ตแบบอัตโนมัติเมื่อเริ่มจ่ายกระแสไฟฟ้าเข้าไมโครคอนโทรลเลอร์ (Power on reset)
- ระบบการอินเตอร์รัพท์จากภายนอก (EXTERNAL INTERRUPT)

- ระบบการตรวจจับระดับสัญญาณอนาล็อก(Analog Comparator)

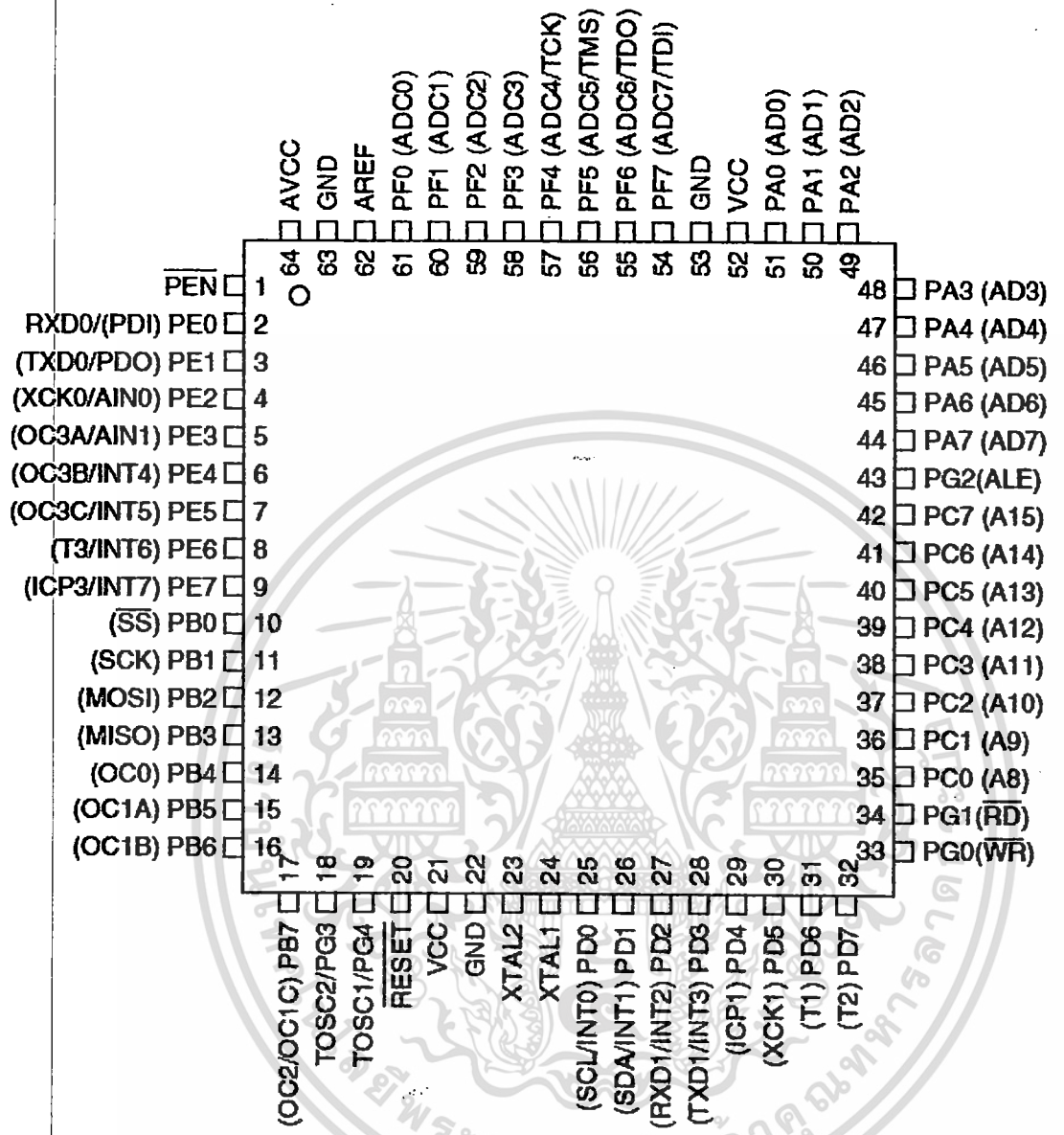
ความเร็ว

- ความถี่สัญญาณนาฬิกา 0 - 8 MHz (ATmega128L)
- ความถี่สัญญาณนาฬิกา 0 - 16MHz (ATmega8)

แรงดันในการทำงาน

- Vcc: 2.7 – 5.5 (ATmega8L)
- Vcc: 4.5 – 5.5 (ATmega8)

<u>องค์ประกอบของขาต่างๆ</u>



รูปที่ 2.12 สัญลักษณ์ของขาไมโครคอนโทรลเลอร์

การเขียนโปรแกรมภาษาซี

ในการใช้งานไมโครคอนโทรลเลอร์ตระกูล AVR หรือตระกูลใดก็ตาม จำเป็นต้องมีความรู้พื้นฐานเกี่ยวกับภาษาซี เพื่อจะทำให้การพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์เป็นไปโดยถูกต้องและไม่มีข้อผิดพลาด ในส่วนนี้จึงเป็นการทบทวนการเขียนโปรแกรมด้วยภาษาซีอย่างคร่าวๆ หากต้องการศึกษาเพิ่มเติม สามารถหาความรู้ได้จากหนังสือ บทความ เอกสารต่างๆได้

โครงสร้างของภาษาซี

โดยหลักๆแล้วโปรแกรมที่ถูกเขียนด้วยภาษาซีนั้นจะประกอบไปด้วยโครงสร้างตามนี้

- พรีโปรเซสเซอร์ ไดเร็กตีฟ (Preprocessor directives)
- การประกาศ (Declarations)
- การกำหนดค่า(Definitions)
- นิพจน์(Expressions)
- ข้อความคำสั่ง(Statements)
- ฟังก์ชัน(Functions)

พรีโปรเซสเซอร์ ไดเร็กตีฟ (Preprocessor directives)

คือชุดของคำสั่งที่เตรียมไว้เพื่อใช้งาน โดยเมื่อมีการคอมไพล์เกิดขึ้น โปรแกรมจะทำการคอมไพล์ส่วนนี้ก่อน

ยกตัวอย่างพรีโปรเซสเซอร์ ไดเร็กตีฟ (Preprocessor directives) ที่ใช้ในการเขียน Code นี้คือ

```
#include<avr/io.h>
#include<stdio.h>
#include<math.h>
```

การประกาศ (Declarations)

ในการเขียนโปรแกรมจำเป็นจะต้องมีการใช้งานตัวแปรต่าง การที่จะใช้งานตัวแปรได้นั้นจำเป็นต้องมีการประกาศตัวแปร ในการประกาศตัวแปรนั้นต้องมีการระบุชนิดของตัวแปรด้วย เช่น

```
unsigned int i;
char j;
```

การกำหนดค่า(Definitions)

เป็นการกำหนดค่าเริ่มต้นให้กับตัวแปร ในขณะที่มีการประกาศตัวแปร เช่น

```
Int I = 5;
```

นิพจน์(Expressions)

คือการคำเนินการกันระหว่างตัวแปร แล้วเกิดผลลัพธ์เป็นค่าใดค่าหนึ่ง เช่น

```
B = B + 10;                    //คือการเพิ่มค่าของตัวแปร B ขึ้น 10
```

ข้อความคำสั่ง(Statements)

คือคำสั่งต่างๆที่ใช้เพื่อให้โปรแกรมที่เขียนเป็นไปตามจุดประสงค์ตามต้องการ เช่น

```
for(i=0;i<=10;i++)
{
        j = j + I;
}
```

ฟังก์ชัน(Functions)

คือส่วนประกอบหนึ่งของโปรแกรมที่รวมเอาส่วนประกอบต่างที่กล่าวมาแล้วข้างต้นเข้าด้วยกัน จนสามารถทำงานได้เสร็จสิ้นด้วยตัวมันเอง เช่น

```
void delay(unsigned int i)
{
        While(i--);
}
```

การสร้างประโยคคำสั่งในภาษา C

ในภาษา C โดยรายละเอียดพื้นฐานดังกล่าว จะประกอบด้วย เซตของอักขระ (Character Set), ชื่อ (Identifiers), คำหลัก (Keyword), ชนิดข้อมูล (Data Types), ค่าคงตัว (Constants), ตัวแปร (Variable)

เซตของอักขระในภาษา C (The C Character set)

จะประกอบด้วย

1. ตัวอักษร (Letter) ได้แก่ A ถึง Z, a ถึง z
2. ตัวเลข (Digital) ได้แก่ 0 ถึง 9
3. อักขระพิเศษ (Special Character) ต่าง ๆ มีดังรายการต่อไปนี้

!	*	+	\	”	<
#	(	=	□	{	>
%	)	~	;	}	/
^	_	[	:	,	?
&	_	]	’	.	(blank)

ชื่อ (Identifiers)

เป็นเซตของอักขระที่ประกอบด้วยตัวอักษร, ตัวเลข และอักขระขีดล่าง (_) และห้ามขึ้นต้นด้วยตัวเลข

คำหลัก (Keywords) หรือ คำสงวน (Reserved words)

เป็นคำที่มีความหมายพิเศษให้กับคอมไพเลอร์ของ C ดังนั้น นักเขียนโปรแกรม (Programmer) จะต้องระมัดระวัง ไม่ควรใช้คำเหล่านี้มาตั้งชื่อเป็นตัวแปร โดยเด็ดขาด คำหลัก หรือ คำสงวนในภาษา C มีดังรายการต่อไปนี้

auto	extern	sizeof
break	float	static
case	for	struct
char	goto	switch
const	if	typedef
continue	int	union
default	long	unsigned
do	register	void
double	return	volatile
else	short	while
enum	signed	

ชื่อ (Identifier) เราจะแบ่งออกเป็น 2 แบบ คือ

1. ชื่อมาตรฐาน (Standard identifiers)
2. ชื่อที่ผู้ใช้นิยามขึ้นมาเอง (User–defined identifiers)

ชื่อมาตรฐาน เป็นคำที่มีความหมายพิเศษที่ภาษา C ได้กำหนดขึ้นมา เช่น printf และ scanf เป็นชื่อที่ถูกนิยามในไลบรารีแสดงและรับข้อมูลส่วน ชื่อที่ผู้ใช้นิยามขึ้นมาเองเป็นชื่อที่ผู้ใช้กำหนดขึ้นมาเอง เพื่อนำมาใช้ในโปรแกรม

กฎเกณฑ์ในการตั้งชื่อ (Rules for Constructing Identifiers) มีดังนี้

1. ชื่อจะต้องประกอบด้วยอักษร A ถึง Z, a ถึง z, ตัวเลข 0 ถึง 9 และอักขระขีดล่าง (_)
2. อักขระตัวแรก จะต้องเป็นตัวอักษรหรืออักขระขีดล่าง

3. ความยาวของชื่อจะเป็นเท่าไรก็ได้ แต่คอมไพเลอร์ของภาษา C จะแยกความแตกต่างของอักขระ 31 ตัวแรกเท่านั้น ตามมาตรฐานของ ANSI C (คอมไพเลอร์ของ C บางตัว จะแยกความแตกต่างเพียง 8 ตัวอักขระแรกเท่านั้น)

หมายเหตุ: ANSI ย่อมาจาก American National Standards Institute

4. ห้ามมีช่องว่าง

5. ห้ามนำคำหลักหรือคำสงวนมาใช้ในการตั้งชื่อ

6. ชื่อที่ตั้งในภาษา C จะมีลักษณะเป็น Case Sensitive หมายความว่า อักษรตัวเล็กกับอักษรตัวใหญ่ จะถือว่ามีความแตกต่างกัน เช่น Tax และ tax ซึ่งชื่อทั้ง 2 นี้ ตั้งได้ถูกต้อง แต่เราจะถือว่ามีความแตกต่างกัน

การตั้งชื่อ ควรจะตั้งชื่อที่ง่ายต่อความเข้าใจ (Easy to Understand) และมีความหมายที่ชัดเจน (Meaning is clear) เช่น ถ้าเราต้องการตั้งชื่อเก็บข้อมูลเกี่ยวกับเงินเดือน เราควรจะตั้งชื่อว่า Salary แทนที่จะใช้ S หรือ Bagel ซึ่งดูแล้วไม่สื่อความหมายเลย ถ้าชื่อประกอบด้วย 2 คำ หรือมากกว่า เราควรจะใช้อักษรขีดล่าง (_) ระหว่างคำ เพื่อปรับปรุงทำให้ชื่อนั้นอ่านง่ายและดูดี เช่น dollars_per_hour แทนที่จะใช้ dollarsperhour แต่พยายามหลีกเลี่ยงการตั้งชื่อที่ยาว เนื่องจากอาจจะมีผลต่อการพิมพ์ชื่อผิดได้

ค่าคงตัว (Constants)

ในภาษา C จะมีค่าคงตัวพื้นฐานอยู่ 4 แบบ คือ

ค่าคงตัวจำนวนเต็ม (Integer Constants)

ค่าคงตัวจุดลอยตัว (Floating–point Constants)

ค่าคงตัวอักขระ (Character Constants)

ค่าคงตัวสายอักขระ (String Constants)

<u>ค่าคงตัวจำนวนเต็ม</u>

ซึ่งในภาษา C จะมีตัวเลขอยู่ 3 แบบ คือ ตัวเลขจำนวนเต็มฐานสิบ, ตัวเลขจำนวน เต็มฐานแปด และตัวเลขจำนวนเต็มฐานสิบหก

กฎเกณฑ์ในการพิจารณาค่าคงตัวจำนวนเต็ม มีดังนี้

1) ห้ามมีเครื่องหมายต่าง ๆ และช่องว่างระหว่างตัวเลข
2) จำนวนเต็มใด ๆ ถ้าไม่มีเครื่องหมายนำหน้า จะถือว่ามีเครื่องหมายบวก
3) ถ้ามีตัวเลขมากกว่า 1 หลัก ห้ามใส่เลขศูนย์นำหน้า เนื่องจากภาษา C จะตีความหมายว่าเป็นตัวเลขจำนวนเต็มฐานแปด

ค่าคงตัวจุดลอยตัว

จะเป็นตัวเลขฐานสิบที่มีจุดทศนิยม หรือเขียนให้อยู่ในรูปของเลขชี้กำลัง ที่เป็นสัญลักษณ์ทางวิทยาศาสตร์ กล่าวคือ จะเขียนจำนวนนั้นให้อยู่ในรูป $A \times 10n$ เมื่อ $1 \leq A < 10$, n เป็นจำนวนเต็มใด ๆ เช่น $63200 = 6.3200 \times 104$

ค่าคงตัวอักขระ

จะเป็นอักขระเพียง 1 ตัว ที่ปิดล้อมโดยเครื่องหมาย apostrophes (single quotation marks) (‘ ’) ค่าคงตัวอักขระจะเป็นรหัสที่ใช้ในคอมพิวเตอร์ที่ถูกเรียกว่า รหัสแอสกี (ASCII) ซึ่งจะมีการกำหนดค่าให้กับรหัสแต่ละตัว

เซตของอักขระ ASCII (The ASCII Character Set)

ASCII Value	Character	ASCII Value	Character	ASCII Value	Character	ASCII Value	Character
000	NUL	032	Blank	064	@	096	`
001	SOH	033	!	065	A	097	a
002	STX	034	"	066	B	098	b
003	ETX	035	#	067	C	099	c
004	EOT	036	$	068	D	100	d
005	ENQ	037	%	069	E	101	e
006	ACK	038	&	070	F	102	f
007	BEL	039	'	071	G	103	g
008	BS	040	(	072	H	104	h
009	HT	041	)	073	I	105	i
010	LF	042	*	074	J	106	j
011	VT	043	+	075	K	107	k
012	FF	044	,	076	L	108	l
013	CR	045	-	077	M	109	m
014	SO	046	.	078	N	100	n
015	SI	047	/	079	O	111	o
016	DLE	048	0	080	P	112	p
017	DC1	049	1	081	Q	113	q
018	DC2	050	2	082	R	114	r
019	DC3	051	3	083	S	115	s
020	DC4	052	4	084	T	116	t
021	NAK	053	5	085	U	117	u
022	SYN	054	6	086	V	118	v
023	ETB	055	7	087	W	119	w
024	CAN	056	8	088	X	120	x
025	EM	057	9	089	Y	121	y
026	SUB	058	:	090	Z	122	z
027	ESC	059	;	091	[	123	{
028	FS	060	<	092	\	124	□
029	GS	061	=	093	]	125	}
030	RS	062	>	094	↑	126	~
031	US	063	?	095	_	127	DEL

Note : The first 32 characters and the last character are control character; they cannot be printed.

(ข้อสังเกต : อักขระ 32 ตัวแรก และอักขระตัวสุดท้าย จะเป็นตัวอักขระที่ใช้ในการควบคุม ไม่สามารถนำมาพิมพ์ได้)

ค่าคงตัวสายอักขระ

ได้แก่ ชุดของอักขระที่นำมาเขียนให้ติดต่อกัน และปิดล้อมด้วยเครื่องหมาย double quotation markes (“ ”)

ค่าคงตัวแบบสายอักขระที่ถูกต้อง ได้แก่

“125” , “27–10–99” , “” , “what is it?”

หมายเหตุ

1) ถ้าต้องการพิมพ์ผลลัพธ์ดังนี้ Jim “Mac” MacDonald.
ให้ใส่ลำดับหลีกดังนี้“Jim \“Mac\“MacDonald.”

2) “ ” เป็นสายอักขระที่ไม่มีตัวอักขระ เราจะเรียกว่า null (empty) string

3) “Line1\n Line2\n line3” จะแสดงผลบนจอภาพดังนี้

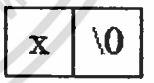
Line1
Line2
Line 3

เนื่องจากมีการใช้ลำดับหลีก \n หมายถึงขึ้นบรรทัดใหม่

4) ค่าคงตัวอักขระ ‘x’ กับค่าคงตัวสายอักขระ “x” จะมีความแตกต่างกัน กล่าวคือ

ค่าคงตัวอักขระ ‘x’ จะเก็บข้อมูล ดังรูป

ค่าคงตัวสายอักขระ “x” จะเก็บข้อมูล ดังรูป



โดย \0 เป็นตัวอักขระที่เรียกว่า อักขระว่าง (null character) ที่ใช้เติมท้ายเพื่อบอกจุดสิ้นสุดของสายอักขระ ภาษา C จะเติมให้โดยอัตโนมัติ

ลำดับหลีก (Escape Sequences)

เครื่องหมาย backslash (\ ) จะถูกเรียกว่า อักขระหลีก (Escape Character) ซึ่งจะเป็นตัวที่บอกให้คำสั่ง printf ทำบางสิ่งบางอย่างนอกเหนือจากงานปกติ และเมื่อนำมารวมกับอักขระจะถูกเรียกว่าลำดับหลีก (Escape Sequences) ซึ่งจะมีความหมายพิเศษตามอักขระที่ตามหลังเครื่องหมาย \ โดยเมื่อคอมไพเลอร์พบเครื่องหมาย \ ในสายอักขระ (string) มันก็จะมองหาอักขระตัวถัดไป เมื่อพบก็จะทำตามคำสั่งที่

ได้กำหนดไว้ เช่น \n เป็นลำดับหลีก ซึ่งมีความหมายว่าให้ขึ้นบรรทัดใหม่ (Newline) นั่นคือ เมื่อมีการใช้ลำดับหลีก \n ในคำสั่ง printf ก็จะทำให้มีการขึ้นบรรทัดใหม่ โดยเคอร์เซอร์จะไปอยู่ที่ตำแหน่งแรกของบรรทัดใหม่ หมายเหตุ อักขระที่ตามหลังเครื่องหมาย \ อาจจะมี 1 ตัวหรือมากกว่า 1 ตัว ก็ได้

ตารางแสดงลำดับหลีก

ตัวอักขระ (Character)	ลำดับหลีก (Escape Sequence)	ค่ารหัสแอสกี (ASCII Value)
bell (alert)	\a	007
backspace	\b	008
horizontal tab	\t	009
vertical tab	\v	011
newline (line feed)	\n	010
form feed	\f	012
carriage return	\r	013
quotation mark (")	\"	034
apostrophe (')	\'	039
question mark (?)	\?	063
backslash (\)	\\	092
null	\0	000

ชนิดของข้อมูล (Types of data)

ในภาษา C จะแบ่งประเภทข้อมูลออกเป็น 2 แบบ คือ

1. จำนวน (Numbers) ยังแบ่งออกเป็น 2 แบบ คือ
   - จำนวนเต็ม (Integer)
   - จำนวนที่มีจุดทศนิยม (Float)
2. ตัวอักขระ (Characters)
3. สายอักขระ (String)

โดยในภาษา C ได้กำหนดชื่อที่เป็นคำเฉพาะให้กับข้อมูลแต่ละประเภท ดังนี้

ข้อมูลประเภทชนิดจำนวนเต็ม จะเขียนแทนด้วย int

ข้อมูลประเภทชนิดจำนวนที่มีจุดทศนิยม จะเขียนแทนด้วย float

ข้อมูลประเภทชนิดตัวอักขระ จะเขียนแทนด้วย char

แสดงชนิดข้อมูล และพิสัยของค่าที่ใช้ใน IBM PC.

ชนิด	ความหมาย	พิสัยของค่า
void	ว่าง (null)	0
char	ตัวอักขระ	–128 ถึง 127
int	จำนวนเต็ม	–32,768 ถึง 32,767
short	จำนวนเต็มแบบ short	–32,768 ถึง 32,767
long	จำนวนเต็มแบบ long	–2,147,483,648 ถึง 2,147,483,647
unsigned char	ตัวอักขระไม่มีเครื่องหมาย	0 ถึง 255
unsigned	จำนวนเต็มไม่มีเครื่องหมาย	0 ถึง 65,535
unsigned short	จำนวนเต็มไม่มีเครื่องหมายแบบ short	0 ถึง 65,535
unsigned long	จำนวนเต็มไม่มีเครื่องหมายแบบ long	0 ถึง 4,294,967,295
float	จุดลอยตัว	3.4E +/– 38 (7 digits)
double	จุดลอยตัวแบบ double	1.7E +/– 308 (15 digits)
long double	จุดลอยตัวแบบ long double	1.7E +/– 4932 (15 digits)

แสดงขนาดชนิดข้อมูล

ชนิดข้อมูล	ขนาด (ไบต์)
จำนวนเต็ม (Integer)	
- มีเครื่องหมาย (Signed)	
จำนวนเต็ม แบบ short (short, short int, signed short, signed short int)	2
จำนวนเต็ม (int, signed, signed int)	2
จำนวนเต็ม แบบ long (long, long int, signed long, signed long int)	4
- ไม่มีเครื่องหมาย (Unsigned)	
จำนวนเต็ม แบบ short (unsigned short, unsigned short int)	2
จำนวนเต็ม (unsigned, unsigned int)	2
จำนวนเต็ม แบบ long (unsigned long, unsigned long int)	4
จุดลอยตัว (Floating–point)	

จุดลอยตัว (float)	4
จุดลอยตัว แบบ double (double)	8
จุดลอยตัว แบบ long double (long double)	16

ตัวอักขระ (Character)

ตัวอักขระ (char)	1

การประกาศตัวแปร (Variable Declarations)

ก่อนที่จะมีการประมวลผล เราจะต้องมีการประกาศตัวแปรทุกตัวให้คอมไพเลอร์ทราบว่า ตัวแปรแต่ละตัวมีข้อมูลเป็นแบบใด เพื่อที่จะได้จองเนื้อที่ให้กับตัวแปรแต่ละตัวได้ถูกต้องการเขียนการประกาศตัวแปรที่เหมาะสมให้กับตัวแปรแต่ละตัว ดังต่อไปนี้

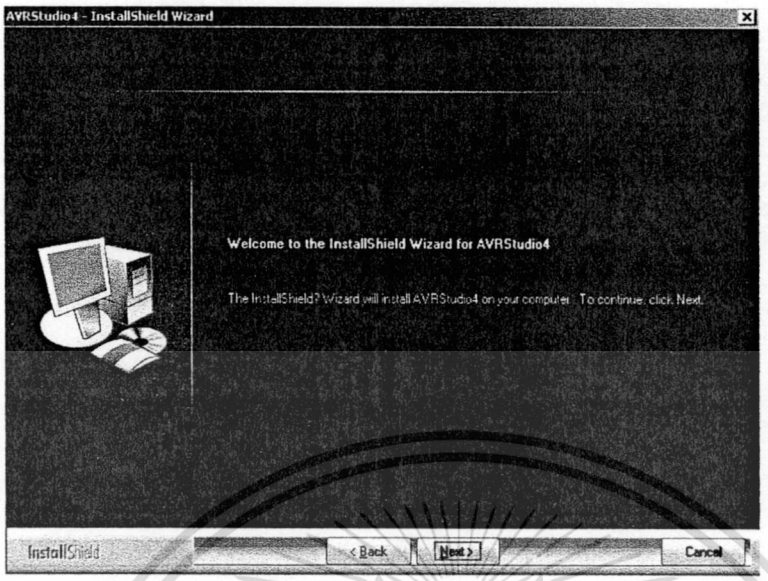
ตัวแปรจำนวนเต็ม : p , q	จะเขียนแทนด้วย int p, q ;
ตัวแปรจำนวนจุดลอยตัว : x, y, z	จะเขียนแทนด้วย float x, y, z ;
ตัวแปรอักขระ : a, b, c	จะเขียนแทนด้วย char a, b, c ;
ตัวแปรจำนวนเต็มแบบ long : l	จะเขียนแทนด้วย long l ;
ตัวแปรจำนวนเต็มแบบ short : t	จะเขียนแทนด้วย short t ;
ตัวแปรจำนวนเต็มแบบ unsigned : n	จะเขียนแทนด้วย unsigned n ;
ตัวแปรแถวลำดับของอักขระที่มีจำนวน 80 ตัว : message	จะเขียนแทนด้วย char message [80] ;

การติดตั้ง AVR Studio 4 และ WinAVR เพื่อใช้ในการทดลองเขียนโปรแกรม ไมโครคอนโทรลเลอร์

ขั้นตอนแรกในการติดตั้งคือ ติดตั้งตัวโปรแกรม AVR Studio 4 ก่อน ซึ่งสามารถดาวโหลดตัวโปรแกรมได้จากเวป http://www.atmel.com

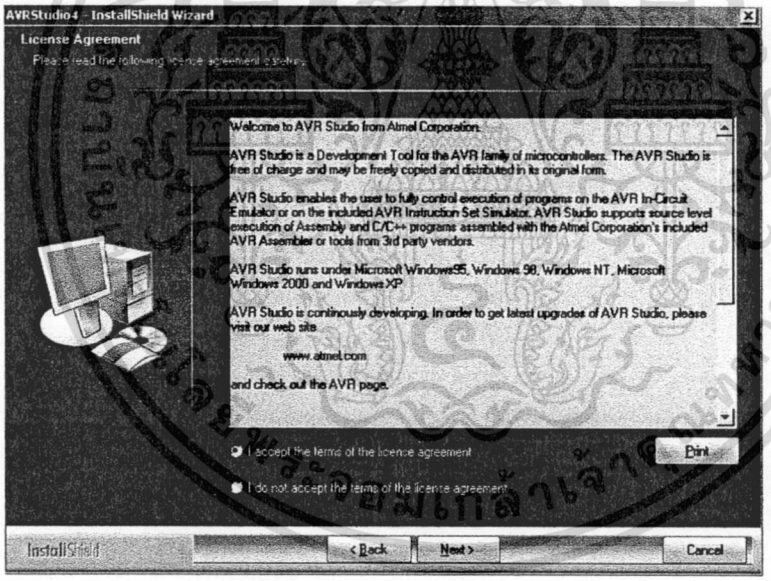
รูปที่ 2.13 ไอคอนของโปรแกรมติดตั้ง AVR Studio 4

เมื่อดับเบิ้ลคลิก เปิดโปรแกรมสำหรับติดตั้งตัวนี้ขึ้นมาแล้ว จะมีหน้าต่างขึ้นมาดังรูป



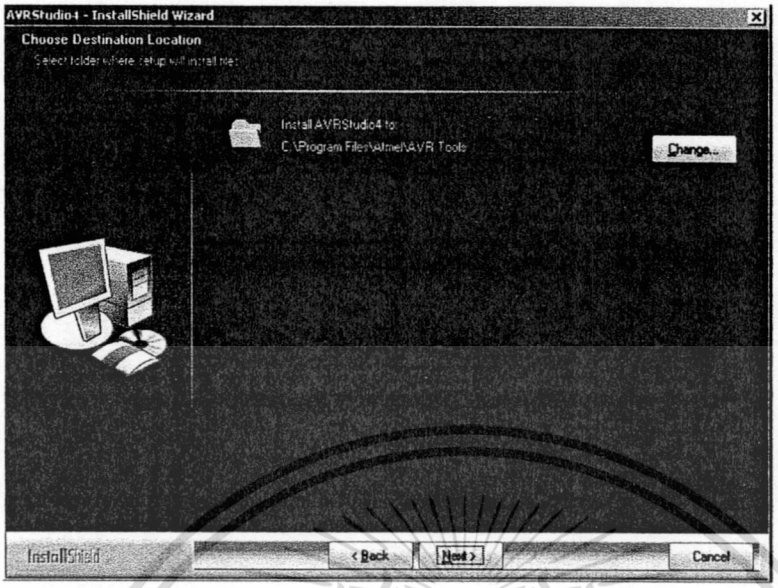
รูปที่ 2.14 ภาพหน้าจอของขั้นตอนการปฏิบัติการ

ให้คลิกที่ Next เพื่อเข้าสู่ขั้นตอนการติดตั้งขั้นตอนต่อไป



รูปที่ 2.15 ภาพหน้าจอของขั้นตอนการปฏิบัติการสู่ขั้นตอนการติดตั้งและยอมรับกฎ

ให้เลือกที่ I accept the terms of the license agreement หลังจากนั้น ให้คลิกที่ Next เพื่อเข้าสู่ขั้นตอน

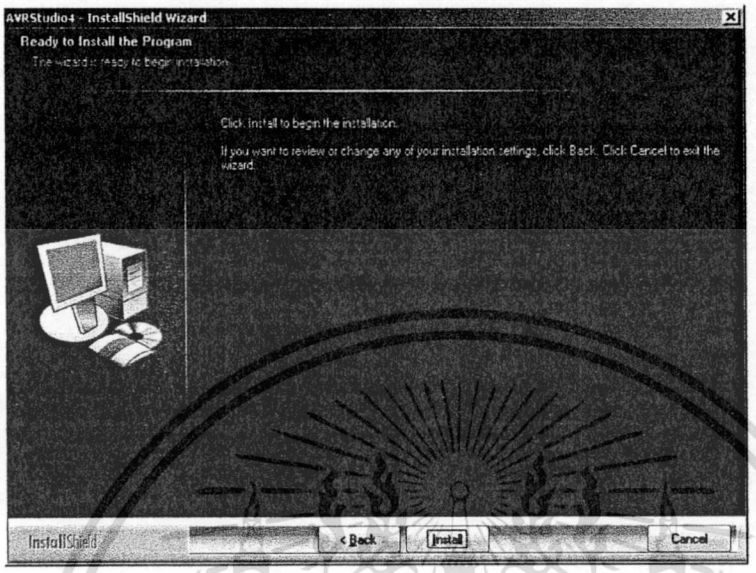


รูปที่ 2.16 ภาพหน้าจอของขั้นตอนการปฏิบัติการสู่ขั้นตอนการติดตั้งขั้นตอนต่อไป

หน้าต่างนี้ จะเป็นการเลือกโฟลเดอร์ที่ใช้ในการติดตั้ง โปรแกรม หากไม่มีการเปลี่ยนแปลงใดๆ โปรแกรมจะตั้งค่ามาตรฐานให้ติดตั้งไว้ที่ C:\Program Files\Atmel\AVR Tools หากต้องการเปลี่ยนแปลงโฟลเดอร์ที่ใช้ในการติดตั้ง ให้เลือกที่ Change...เพื่อเปลี่ยนแปลงสถานที่ติดตั้งโปรแกรม หลังจากนั้น ให้คลิกที่ Next เพื่อเข้าสู่ขั้นตอนต่อไป

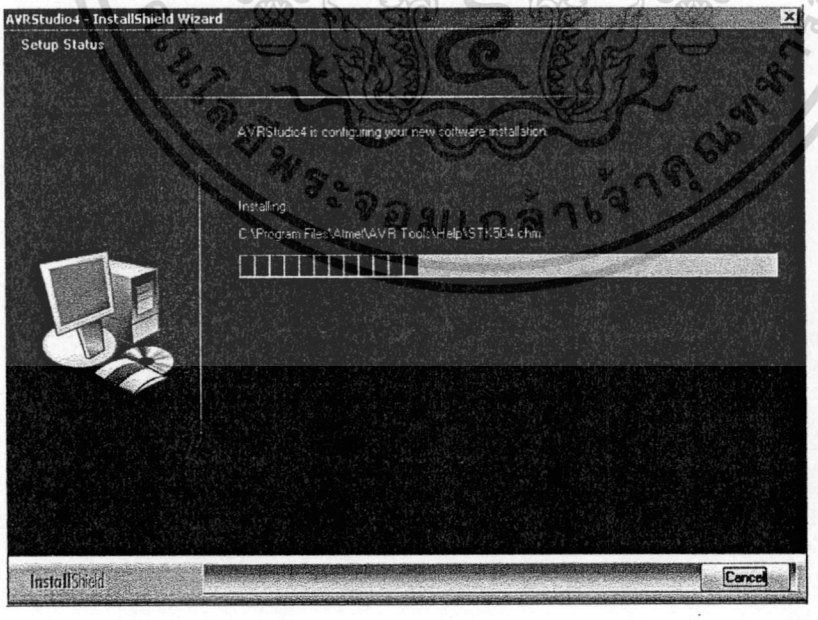
รูปที่ 2.17 ภาพหน้าจอของขั้นตอนการปฏิบัติการการเลือกโฟลเดอร์ที่ใช้ในการติดตั้ง

หน้าต่างต่อมา โปรแกรมจะถามว่า ให้ติดตั้ง USB Driver เพิ่มเติมหรือไม่ ในที่นี้ จะไม่เลือกให้ติดตั้งเพิ่มเพราะ สามารถติดตั้งภายหลังได้ถ้าต้องการ หลังจากนั้น ให้คลิกที่ Next เพื่อเข้าสู่ขั้นตอนต่อไป



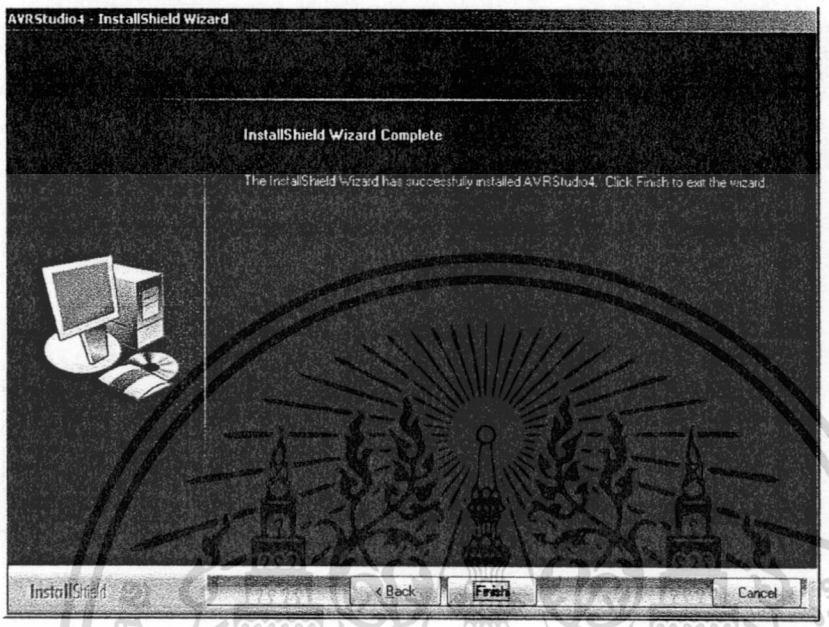
รูปที่ 2.18 ภาพหน้าจอของขั้นตอนการปฏิบัติการให้ติดตั้ง USB Driver เพิ่มเติม

หน้าต่างนี้จะเป็นการยืนยันครั้งสุดท้ายก่อนทำการติดตั้งโปรแกรม AVR Studio 4 หากต้องการเปลี่ยนแปลงกาติดตั้งสามารถ คลิก Back เพื่อกลับไปแก้ไขได้ หลังจากนั้น ให้คลิกที่ Install เพื่อเข้าสู่ขั้นตอนการติดตั้ง



รูปที่ 2.19 ภาพหน้าจอของขั้นตอนการปฏิบัติการ Install เพื่อเข้าสู่ขั้นตอนการติดตั้ง

หลังจากนั้นโปรแกรมจะทำการติดตั้งโปรแกรมให้ที่โฟลเดอร์ที่ได้กำหนดไว้ก่อนการติดตั้ง รอให้โปรแกรมติดตั้งจนเสร็จสมบูรณ์

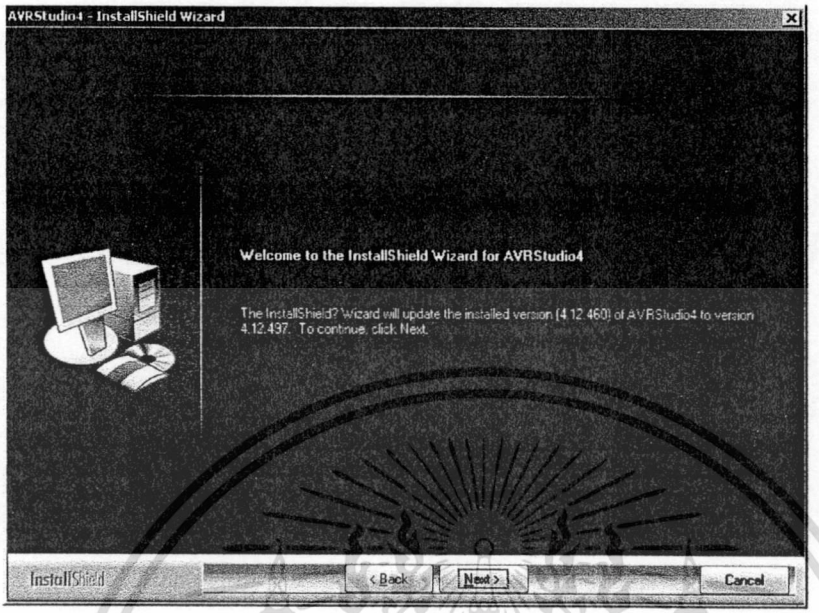


รูปที่ 2.20 ภาพหน้าจอของขั้นตอนการปฏิบัติการทำการติดตั้งโปรแกรมให้ที่โฟลเดอร์

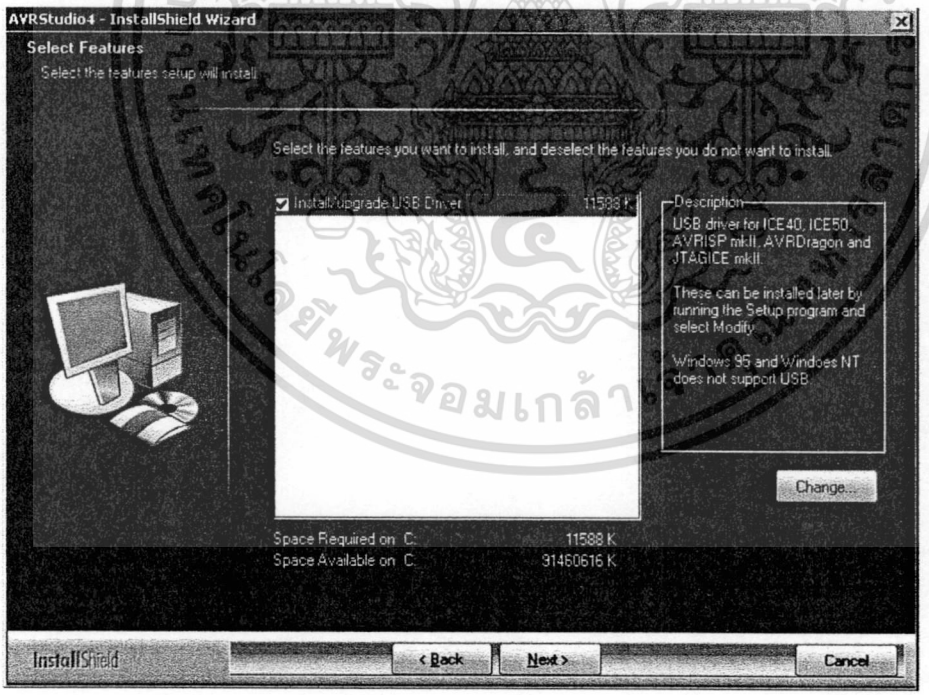
คลิกที่ Finish เพื่อเสร็จสิ้นการติดตั้ง AVR Studio 4 หน้าต่างจะถูกปิดไปเองโดยอัตโนมัติ ถือเป็นการเสร็จสิ้นขั้นตอนการติดตั้ง ขั้นตอนต่อมาจะเป็นการติดตั้ง aStudio412SP4b497.exe ซึ่งเป็นส่วนเสริมของ AVR Studio 4 ที่ได้ติดตั้งมาแล้ว ทำให้ โปรแกรมมีความสามารถมากขึ้น

รูปที่ 2.21 ไอคอน ส่วนเสริมของ AVR Studio 4

เมื่อดับเบิ้ลคลิก เปิดโปรแกรมสำหรับติดตั้งส่วนเสริมขึ้นมาแล้ว จะมีหน้าต่างขึ้นมาดังรูป

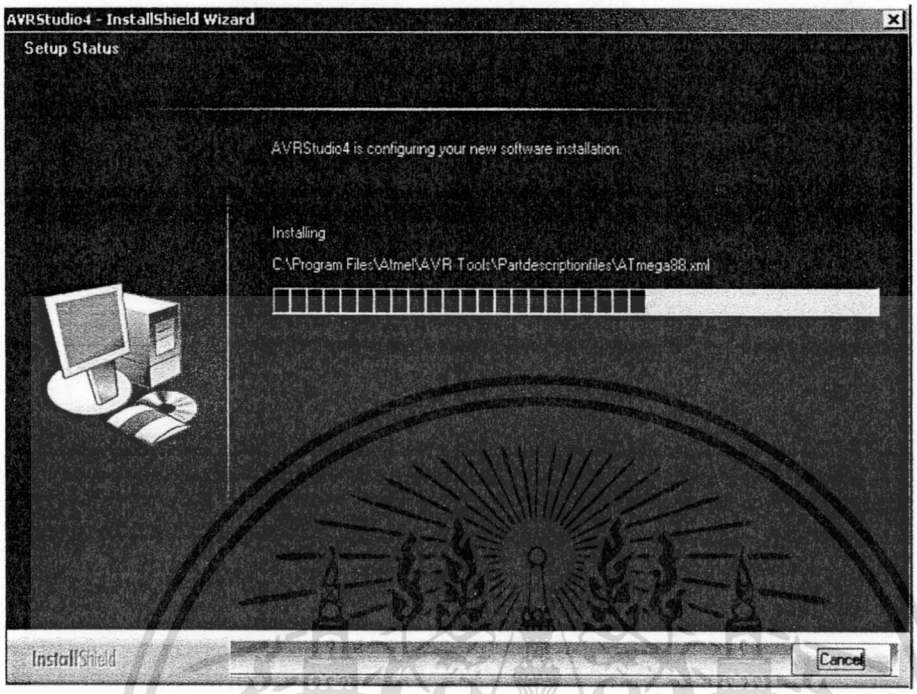


รูปที่ 2.22 ภาพหน้าจอของขั้นตอนการปฏิบัติการดับเบิ้ลคลิก เปิดโปรแกรมสำหรับติดตั้ง ให้คลิกที่ Next เพื่อเข้าสู่ขั้นตอนการติดตั้งขั้นตอนต่อไป



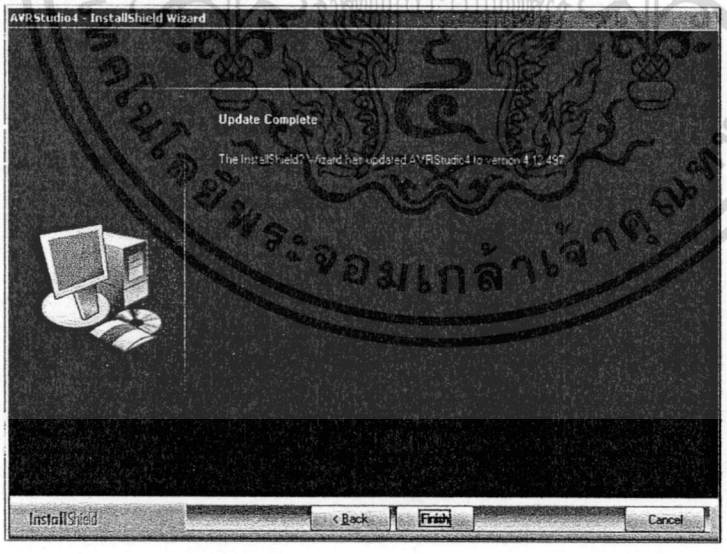
รูปที่ 2.23 ภาพหน้าจอของขั้นตอนการปฏิบัติการให้ติดตั้ง USB Driver เพิ่มเติมหรือไม่

หน้าต่างต่อมา โปรแกรมจะถามว่า ให้ติดตั้ง USB Driver เพิ่มเติมหรือไม่ ในที่นี้ให้เลือกที่ Install/upgrade USB Driver หลังจากนั้น ให้คลิกที่ Next เพื่อเข้าสู่ขั้นตอนต่อไป



รูปที่ 2.24 ภาพหน้าจอของขั้นตอนการปฏิบัติการการติดตั้งโปรแกรมให้ที่โฟลเดอร์

หลังจากนั้นโปรแกรมจะทำการติดตั้งโปรแกรมให้ที่โฟลเดอร์ที่ติดตั้ง AVR Studio 4 ไว้ หลังจากนั้นรอให้โปรแกรมติดตั้งจนเสร็จสมบูรณ์



รูปที่ 2.25 ภาพหน้าจอของขั้นตอนการปฏิบัติการ Finish เพื่อเสร็จสิ้นการติดตั้ง AVR Studio 4

คลิกที่ Finish เพื่อเสร็จสิ้นการติดตั้ง AVR Studio 4 หน้าต่างจะถูกปิดไปเองโดยอัตโนมัติ ถือเป็นการเสร็จสิ้นขั้นตอนการติดตั้ง ขั้นตอนต่อไปเป็นการติดตั้งโปรแกรม WinAVR ซึ่งเป็นโปรแกรมที่ใช้งานร่วมกับ

AVR Studio 4 ที่ได้ติดตั้งไว้แล้วเป็น WinAVR เป็นคอมไพล์เลอร์ที่ใช้สำหรับ คอมไพล์โค้ดโปรแกรมภาษา C ให้เป็น ไฟล์นามสกุล Hex

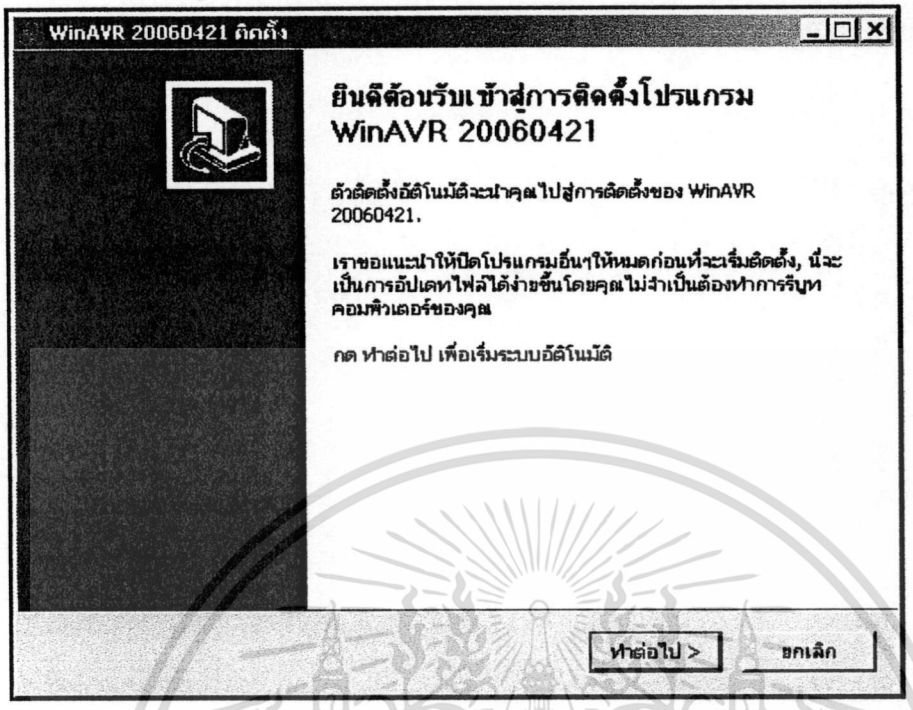
WinAVR-20060421-install

รูปที่ 2.26 ไอคอนของตัวติดตั้ง WinAVR

ในการติดตั้งนั้นให้ดับเบิ้ลคลิกไอคอนของ WinAVR ขึ้นมาก่อน

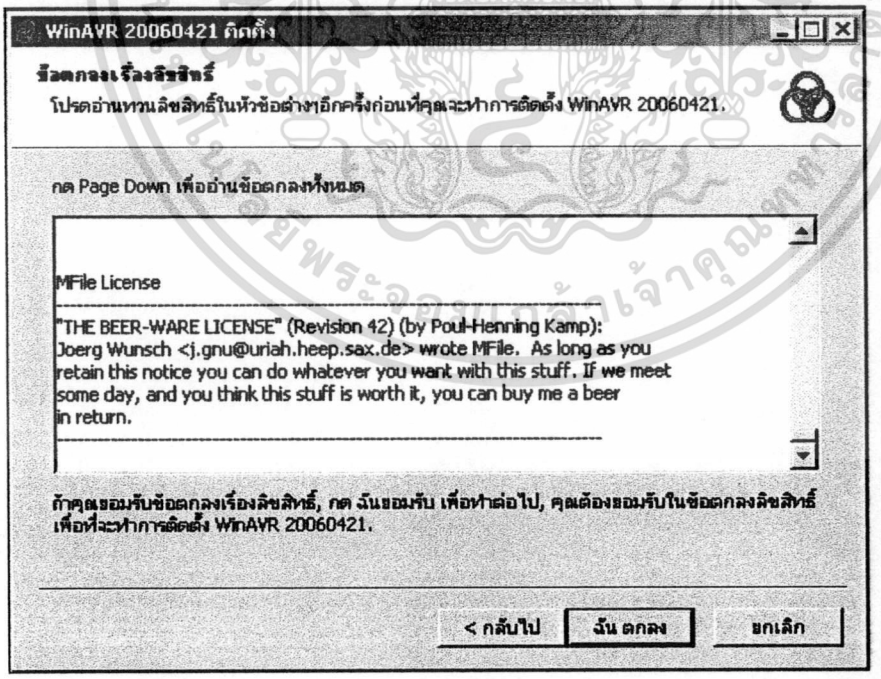
รูปที่ 2.27 ภาพหน้าจอของขั้นตอนการปฏิบัติการเลือกภาษาที่ใช้

จะมีหน้าต่างให้เลือกภาษาที่ใช้ในการติดตั้ง ปรากฏขึ้นมา สามารถเลือกภาษาตามที่ต้องการได้ ในที่นี้จะเลือกภาษาที่ใช้ติดตั้งเป็น ภาษาไทย เมื่อเลือกภาษาแล้วต่อมาให้คลิกที่ OK เพื่อเข้าสู่ขั้นตอนการติดตั้ง



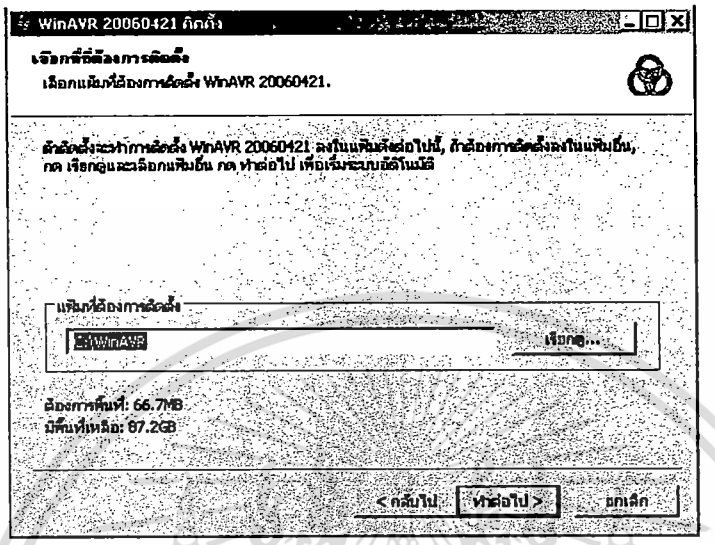
รูปที่ 2.28 ภาพหน้าจอของขั้นตอนการปฏิบัติการเข้าสู่โปรแกรมการติดตั้ง

คำแนะนำในการติดตั้งบอกว่าให้ปิดโปรแกรมอื่นๆ ให้หมดเสียก่อนเพื่อให้การติดตั้งเป็นไปได้ด้วยความรวดเร็ว การติดตั้ง WinAVR นั้น ไม่จำเป็นต้อง รีสตาร์ทเครื่อง คอมพิวเตอร์ใหม่ คลิกที่ “ทำต่อไป”



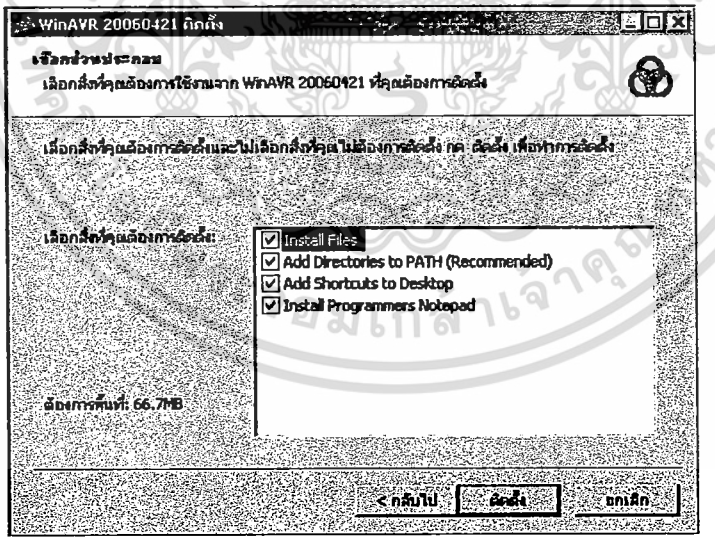
รูปที่ 2.29 ภาพหน้าจอของขั้นตอนการปฏิบัติการคำแนะนำในการติดตั้ง

หน้าต่างต่อมาจะเป็นการตกลงเกี่ยวกับลิขสิทธิ์ ให้คลิกที่ “ฉัน ตกลง” เพื่อยอมรับเงื่อนไขเกี่ยวกับลิขสิทธิ์ และเข้าสู่ขั้นตอนติดตั้งขั้นตอนอื่นต่อไป



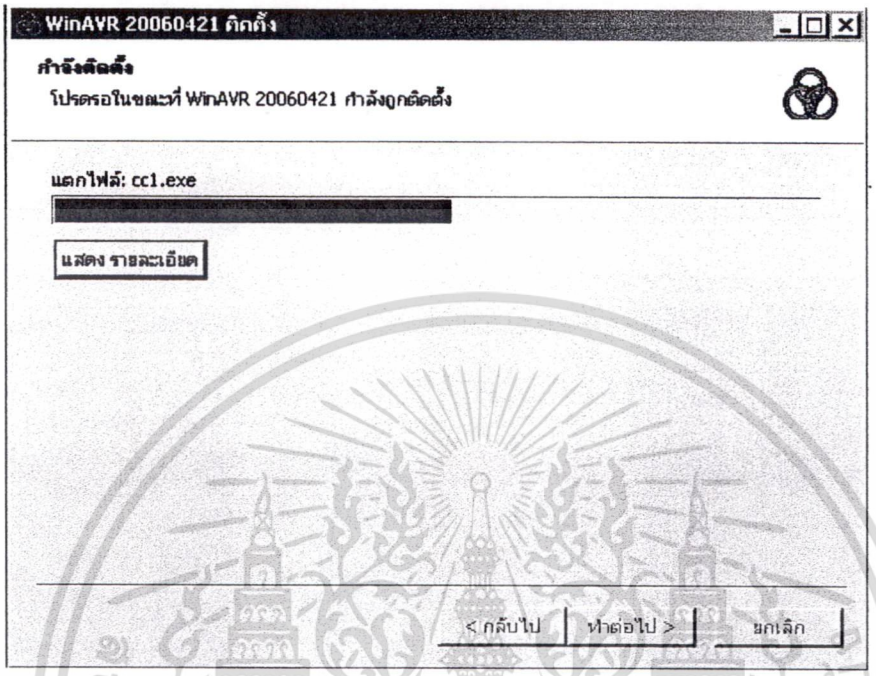
รูปที่ 2.30 ภาพหน้าจอของขั้นตอนการปฏิบัติการยอมรับเงื่อนไข

ต่อไปเป็นการเลือกโฟลเดอร์ที่ใช้ให้การติดตั้ง WinAVR โดยโปรแกรมจะตั้งค่ามาตรฐานไว้ที่ C:\WinAVR หากต้องการเปลี่ยนแปลงโฟลเดอร์ที่ใช้ติดตั้ง WinAVR ให้คลิกที่ “เรียกดู..” เพื่อเปลี่ยนแปลงโฟล์เดอร์ที่ใช้ติดตั้ง WinAVR เมื่อเลือกโฟลเดอร์ที่ใช้ติดตั้งเรียบร้อยแล้วให้คลิก “ทำต่อไป”



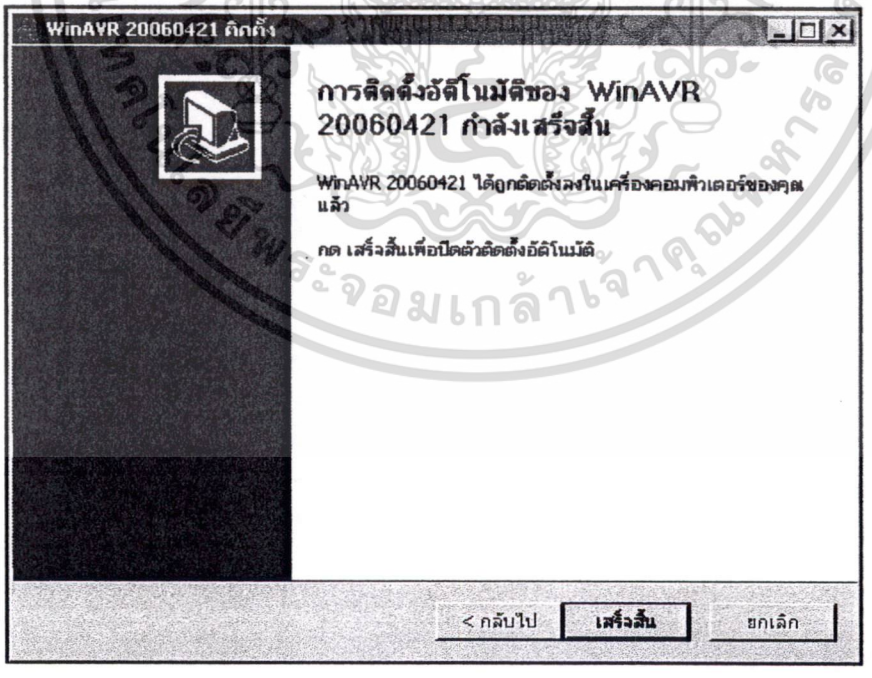
รูปที่ 2.31 ภาพหน้าจอของขั้นตอนการปฏิบัติการยอมรับเงื่อนไขและทำต่อไป

หน้าต่างต่อมาจะเป็นการเลือกสิ่งที่จะติดตั้ง ให้เลือกติดตั้งทั้งหมดโดยการทำเครื่องหมาย หน้าสิ้งที่ต้องการติดตั้งทั้งหมด หลังจากนั้นคลิกที่ “ติดตั้ง” เพื่อเริ่มการติดตั้ง



รูปที่ 2.32 ภาพหน้าจอของขั้นตอนการปฏิบัติการการติดตั้ง

โปรแกรมจะทำการติดตั้ง WinAVR ให้รอจนกว่าการติดตั้งจะเสร็จสมบูรณ์คลิกที่ “เสร็จสิ้น”



รูปที่ 2.33 ภาพหน้าจอของขั้นตอนการปฏิบัติการเสร็จสิ้นหน้าต่างจะถูกปิดไปเองโดยอัตโนมัติ

ต่อไปเป็นติดตั้งโปรแกรม Khazama AVR Programmer โปรแกรมนี้เป็นโปรแกรมสำหรับนำ ไฟล์ ที่ได้จากการเขียนโปรแกรม ภาษาC จากโปรแกรม AVRStudio4แล้วคอมไฟล์ด้วย WinAVR ได้เป็นไฟล์นามสกุล Hex ไปโปรแกรมใส่ ATmega8 ที่อยู่บนบอร์ดทดลอง

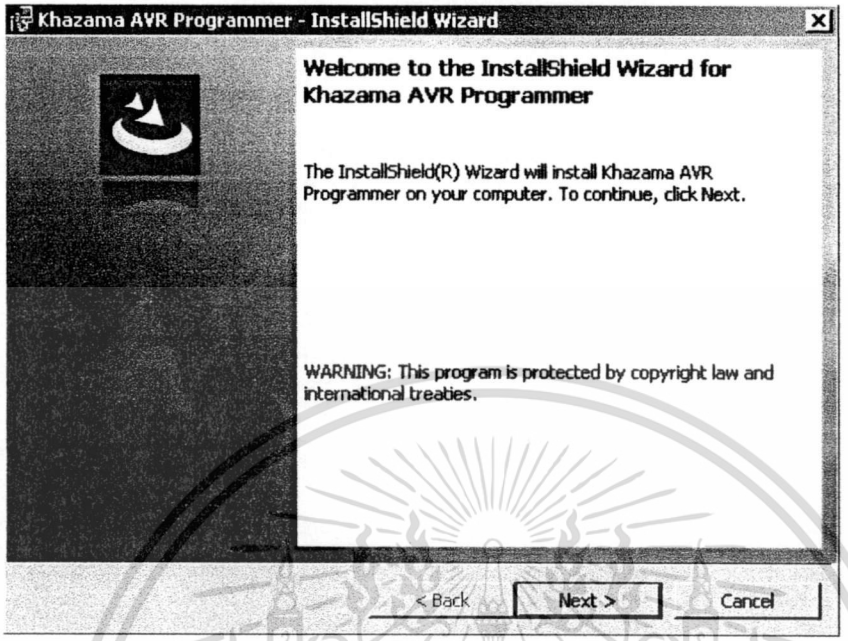
รูปที่ 2.34 ภาพรวมในการเขียนโปรแกรมลงบน ATmega8

จากไดอะแกรมด้านบนจะแสดงภาพรวมในการเขียนโปรแกรมลงบน ATmega8 โดยใช้โปรแกรมต่างๆ ที่ได้ติดตั้งไว้

ขั้นตอนต่อมาเป็นการติดตั้ง โปรแกรม Khazama

รูปที่ 2.35 ไอคอนของตัวติดตั้งโปรแกรม Khazama

เมื่อดับเบิลคลิกเปิดตัวติดตั้งขึ้นมาแล้วจะปรากฏหน้าดังดังรูป

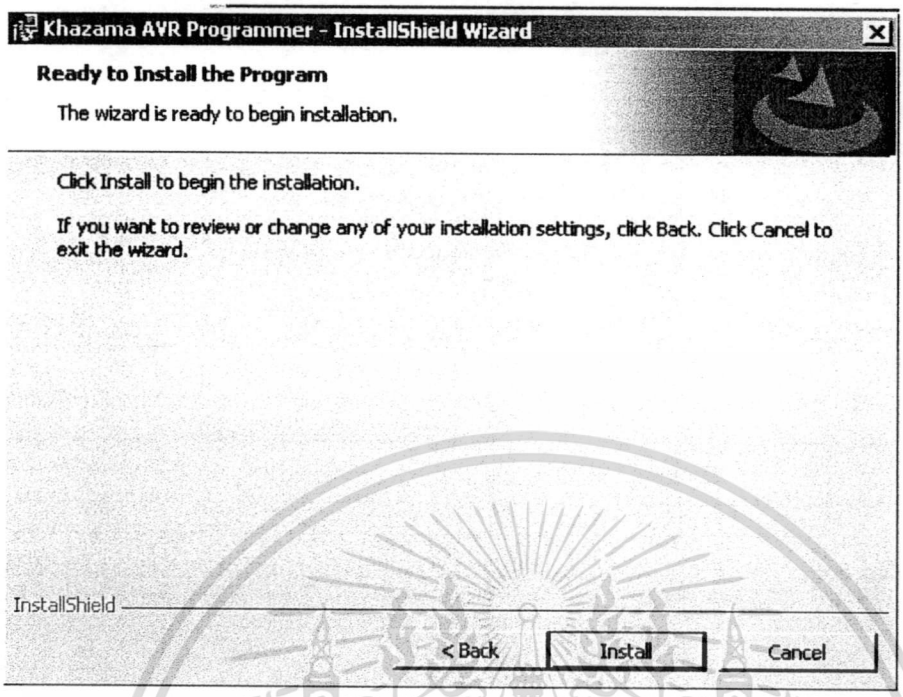


รูปที่ 2.36 ภาพหน้าจอของขั้นตอนการปฏิบัติการเมื่อดับเบิลคลิกเปิดตัวติดตั้ง

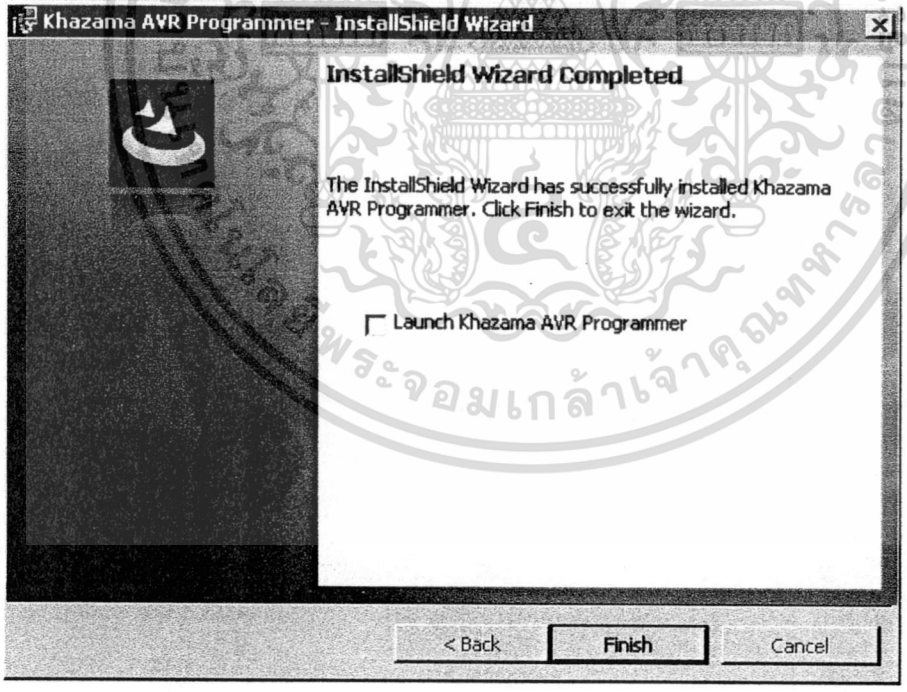
ให้คลิก Next เพื่อเข้าสู่ขั้นตอนการติดตั้งต่อไป

คลิกที่ Install เพื่อเริ่มทำการติดตั้ง ในการติดตั้งโปรแกรมKhazama จะไม่มีหน้าต่างในการเลือกโฟลเดอร์ที่ใช้ติดตั้ง จึงไม่สามารถเปลี่ยนแปลงโฟลเดอร์ที่ใช้ติดตั้งได้ แต่ค่ามาตรฐานของโปรแกรมโฟลเดอร์ที่ใช้ติดตั้งจะอยู่ที่

C:\Program Files\khazama.com\Khazama AVR Programmer



รูปที่ 2.37 ภาพหน้าจอของขั้นตอนการปฏิบัติการเริ่มเปิดตัวติดตั้ง

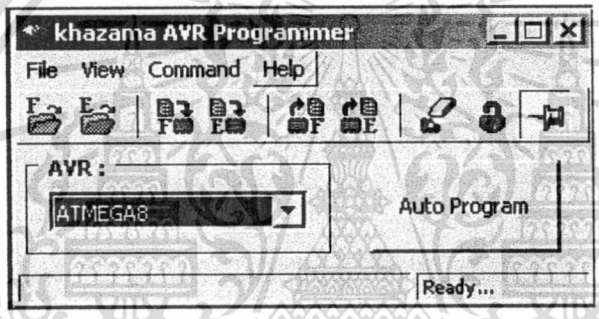


รูปที่ 2.38 ภาพหน้าจอของขั้นตอนการปฏิบัติการเมื่อสำเร็จ

รอจนโปรแกรมติดตั้ง Khazama เสร็จสมบูรณ์ แล้วคลิกที่ Finish เป็นอันเสร็จสิ้นการติดตั้ง หน้าต่างจะถูกปิดโดยอัตโนมัติ

รูปที่ 2.39 ไอคอนโปรแกรม Khazama สำหรับการใช้งาน

เมื่อดับเบิ้ลคลิก เปิดโปรแกรม Khazama ขึ้นมาจะพบหน้าต่างโปรแกรมดังรูป



รูปที่ 2.40 ภาพหน้าจอของขั้นตอนการปฏิบัติการ

Unicode

Unicode กำหนดหมายเลขเฉพาะสำหรับทุกอักขระโดยไม่สนใจว่าเป็นแพล็ตฟอร์มใดไม่ขึ้นกับว่าจะเป็นโปรแกรมใดและไม่ว่าจะเป็นภาษาใด โดยพื้นฐานแล้ว คอมพิวเตอร์จะเกี่ยวข้องกับเรื่องของตัวเลข คอมพิวเตอร์จัดเก็บตัวอักษรและอักขระอื่นๆ โดยการกำหนดหมายเลขให้สำหรับแต่ละตัว ก่อนหน้าที่ Unicode จะถูกสร้างขึ้น ได้มีระบบ encoding อยู่หลายร้อยระบบสำหรับการกำหนดหมายเลขเหล่านี้ ไม่มี encoding ใดที่มีจำนวนตัวอักขระมากเพียงพอยกตัวอย่างเช่น เฉพาะในกลุ่มสหภาพยุโรปเพียงแห่งเดียว ก็ต้องการหลาย encoding ในการครอบคลุมทุกภาษาในกลุ่ม หรือแม้แต่ในภาษาเดี่ยว เช่น ภาษาอังกฤษ ก็ไม่มี encoding ใดที่เพียงพอสำหรับทุกตัวอักษร เครื่องหมายวรรคตอน และสัญลักษณ์ทางเทคนิคที่ใช้กันอยู่ทั่วไป
ระบบ Encoding เหล่านี้ยังขัดแย้งซึ่งกันและกัน นั่นก็คือ ในสอง encoding สามารถใช้หมายเลขเดียวกันสำหรับตัวอักขระสองตัวที่แตกต่างกันหรือใช้หมายเลขต่างกันสำหรับอักขระตัวเดียวกัน ในระบบคอมพิวเตอร์ (โดยเฉพาะเซิร์ฟเวอร์) ต้องมีการสนับสนุนหลาย encoding และเมื่อข้อมูลที่ผ่านไปมาระหว่างการเข้ารหัสหรือแพล็ตฟอร์มที่ต่างกัน ข้อมูลนั้นจะเสี่ยงต่อการผิดพลาดเสียหาย

Unicode กำหนดหมายเลขเฉพาะสำหรับแต่ละอักขระ โดยไม่สนใจว่าเป็นแพล็ตฟอร์มใด ไม่ขึ้นกับว่าจะเป็นโปรแกรมใดและไม่ว่าจะเป็นภาษาใด มาตรฐาน Unicode ได้ถูกนำไปใช้โดยผู้นำในอุตสาหกรรม เช่น Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys และอื่นๆ อีกมาก Unicode เป็นสิ่งที่จำเป็นสำหรับมาตรฐานใหม่ๆ เช่น XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML ฯลฯ. และเป็นแนวทางอย่างเป็นทางการในการทำ ISO/IEC 10646 Unicode ได้รับการสนับสนุนในระบบปฏิบัติการจำนวนมาก บราวเซอร์ใหม่ๆ ทุกตัว และผลิตภัณฑ์อื่นๆ อีกมาก การเกิดขึ้นของ Unicode Standard และทูลส์ต่างๆ ที่มีในการสนับสนุน Unicode เป็นหนึ่งในแนวโน้มทางเทคโนโลยีซอฟต์แวร์ระดับโลกที่มีความสำคัญที่สุด

การรวม Unicode เข้าไปในระบบไคลเอ็นต์-เซิร์ฟเวอร์ หรือ Applications แบบ multi-tiered และเว็บไซต์ จะทำให้เกิดการประหยัดค่าใช้จ่ายมากกว่าการใช้ชุดอักขระแบบเดิม Unicode ทำให้ผลิตภัณฑ์ซอฟต์แวร์หนึ่งเดียว หรือเว็บไซต์แห่งเดียว รองรับได้หลายแพล็ตฟอร์ม หลายภาษาและหลายประเทศโดยไม่ต้องทำการรื้อปรับระบบ Unicode ยังทำให้ข้อมูลสามารถเคลื่อนย้ายไปมาในหลายๆ ระบบโดยไม่เกิดความผิดพลาดเสียหาย
ความสำเร็จของ Unicode ในเรื่องของการรวมการเข้ารหัสอักขระให้เป็นหนึ่ง เป็นที่รู้จักกว้างขวางและมีอิทธิพลต่อการทำซอฟต์แวร์ให้เป็นสากล กล่าวคือสามารถใช้ได้หลายภาษา มาตรฐานนี้มีการนำไปใช้เป็นเทคโนโลยีหลักหลายอย่าง อาทิ XML JAVA และระบบปฏิบัติการสมัยใหม่

ในปัจจุบันมีตัวอักษรให้เลือกใช้อยู่หลายแบบ มี Character set อยู่มากมาย เช่น ตัวอักษรภาษาไทย, ตัวอักษรภาษาญี่ปุ่น ฯลฯ ในอดีตเวลาที่เรา Encode ภาษา ถ้าเป็น ASCII ขนาด 1 byte จะมี 8 bit ซึ่ง ASCII code ก็คือพวกที่เก็บตัวอักษรภาษาอังกฤษตัวเล็ก, ตัวใหญ่, ตัวเลข, เครื่องหมายมากกว่า, น้อยกว่า, ไม่เท่ากับ, full stop, # เป็นต้น ใช้พื้นที่แค่ 7 bit ก็สามารถ Encode ข้อมูลของได้ครบแล้ว

ส่วน bit ที่ 8 นี่ก็ใส่ 0 ลงไป แต่ในประเทศที่ไม่ได้ใช้ตัวอักษรภาษาอังกฤษในการเขียน ก็อยากจะมีรหัสในการเขียนตัวอักษรเป็นของตัวเองเหมือนกัน ดังนั้นแล้วในการแก้ปัญหา ก็คือ จะเอา bit ที่ 8 มาใช้ เช่น ตั้ง bit ที่ 8 เป็น 1 แล้วก็ Encode ข้างในด้วยรหัสของตัวเอง แต่ต้องเป็นรหัสที่ ASCII code ยังไม่ได้ใช้ เช่น 1000011 เป็น 'ก', 1000012 เป็น 'ข', 1000013 เป็น 'ค' เป็นต้น

วิธีการเช่นนี้ทำให้เราสามารถเก็บได้ 2 ภาษา แต่ว่าการทำแบบนี้ไม่ international เพราะว่า ภาษาจีนหรือภาษาญี่ปุ่นก็ใช้แบบเดียวกัน ก็เอา bit ที่ 8 มาใช้ เพราะฉะนั้นแล้ว ตัว 'ก' ของภาษาไทยก็จะไปมีรหัสเหมือนกับตัวอะไรซักอย่างในภาษาจีน ดังนั้นก็เลยมีปัญหาว่าถ้าเราต้องการทำ international business จริงๆ ควรจะมีรหัสเดียวสำหรับอักษรตัวเดียว และในแต่ละภาษาก็ไม่ควรมีรหัสซ้ำกัน

ดังนั้นเราก็เลยมี Unicode ขึ้นมา Unicode ต่างจาก ASCII คือ ASCII เก็บ byte เดียว แต่ Unicode เก็บ 2 byte ซึ่ง ข้อมูล 2 byte เก็บข้อมูลได้มากมายมหาศาล สามารถเก็บข้อมูลได้มากมายหลายภาษาในโลก อย่างภาษาไทยก็อยู่ใน Unicode นี้ด้วยเหมือนกัน ดังนั้นรหัสภาษาไทยเอาไปเปิดในภาษาจีน ก็ยังเป็นภาษาไทยอยู่ ไม่ออกมาเป็นภาษาจีน เพราะว่ามี code ตายตัวอยู่ว่า code นี้จองไว้สำหรับภาษาไทย แล้ว code ตรงช่วงนี้เป็นภาษาจีน ตรงนี้เป็นภาษาญี่ปุ่น จะไม่ใช้ที่ซ้ำกัน เป็นต้น

การส่งข้อความ SMS ผ่านโมดูลโทรศัพท์ (ET-GSM SIM300CZ V1.0)

ในการส่งข้อความ SMS นั้นจะใช้คำสั่ง "AT+CMGS" ในการสั่งงาน โดยในกรณีที่ใช้ Text Mode นั้นให้ใช้รูปแบบคำสั่งเป็น "AT+CMGS="+เบอร์ผู้รับ" โดยเบอร์ของผู้รับต้องใส่รหัสประเทศนำหน้าแทนศูนย์ด้วยเสมอ ซึ่งในกรณีที่เป็นประเทศไทยจะใช้รหัสประเทศเป็น "66" ดังนั้นถ้าต้องการส่งข้อความ SMS ให้กับเบอร์ที่ใช้งานอยู่ในประเทศไทย เช่น 081-1234567 ก็จะต้องกำหนดหมายเลขของเบอร์ผู้รับปลายทางเป็น 6681-1234567 แทน ซึ่งในกรณีนี้จะได้รหัสเบอร์ผู้รับข้อความเป็น "+66811234567" ซึ่งเมื่อโมดูล SIM300CZ ได้รับคำสั่ง "AT+CMGS" เรียบร้อยแล้วมันจะตอบรับด้วยการส่งเครื่องหมาย ">"กลับมาบอก ซึ่งหลังจากนี้เป็นต้นไปผู้ใช้ก็สามารถจะทำการพิมพ์ข้อความต่างๆที่ต้องการจะส่งให้กับโมดูลได้ทันที โดยให้ปิดท้ายข้อความด้วยรหัส "Ctrl+Z" ตามด้วย "Enter" เช่นถ้าต้องการส่งข้อความ SMS ให้กับหมายเลข 0811234567 ด้วยข้อความ "Hello Test SMS" เป็นดังนี้

AT+CMGS="+66811234567"<Ent>

> Hello Test SMS<Ctrl+Z><Ent>

+CMGS: 6

OK

รหัสข้อความ SMS ภาษาไทย

สำหรับข้อความ SMS ที่เป็นภาษาไทยนั้น จะไม่สามารถแสดงผลด้วยโปรแกรม Terminal ปกติได้ ทั้งนี้ก็เนื่องมาจากว่าระบบตัวอักษรที่ใช้ในโปรแกรม Terminal นั้นจะใช้รหัส ASCII ปรกติที่มีขนาดเพียง 1 ไบท์ แต่สำหรับรหัสภาษาไทยที่ใช้ในระบบสื่อสารของโทรศัพท์มือถือต่างๆนั้น จะใช้รหัสพิเศษเฉพาะที่เรียกว่า "Unicode" ซึ่งตัวอักษร 1 ตัวจะประกอบไปด้วยข้อมูลจำนวน 2 ไบท์ โดยรหัส Unicode ของภาษาไทยนั้นจะมีค่าอยู่ระหว่าง 0E00H...0E7FH สำหรับภาษาอังกฤษนั้นถ้าเป็น Unicode จะใช้รหัสตัวอักษรขนาด 2 Byte เช่นเดียวกันกับภาษาไทย โดยจะมีค่าอยู่ระหว่าง 0000H..007FH โดยตามปรกติแล้วถ้าข้อความเป็นภาษาอังกฤษอย่างเดียวรหัสของตัวอักษรที่ใช้ใน SMS จะเป็นแบบ ASCII คือ ใช้รหัส ขนาด 1 ไบท์ โดยตัดรหัส 00H ไบท์แรกใน Unicode ทิ้งไป เช่น A แทนที่จะเป็นรหัส 0041H ก็จะเหลือเพียง 41H เป็นต้น

	┌	┐	└	┘	│	─	●	◘						♫	☼
0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
┼	◀	↕	‼	¶	┴	┬	┤	↑	├	→	←				
0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	002A	002B	002C	002D	002E	002F
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0030	0031	0032	0033	0034	0035	0036	0037	0038	0039	003A	003B	003C	003D	003E	003F
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0040	0041	0042	0043	0044	0045	0046	0047	0048	0049	004A	004B	004C	004D	004E	004F
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	005A	005B	005C	005D	005E	005F
`	a	B	c	d	e	f	g	h	i	j	k	l	m	n	o
0060	0061	0062	0063	0064	0065	0066	0067	0068	0069	006A	006B	006C	006D	006E	006F
p	q	R	s	t	u	v	w	x	y	z	{	|	}	~	
0070	0071	0072	0073	0074	0075	0076	0077	0078	0079	007A	007B	007C	007D	007E	007F

รูปที่ 2.41 ตาราง แสดงรหัส Unicode ภาษาอังกฤษ

0E00	ก 0E01	ข 0E02	ฃ 0E03	ค 0E04	ฅ 0E05	ฆ 0E06	ง 0E07	จ 0E08	ฉ 0E09	ช 0E0A	ซ 0E0B	ฌ 0E0C	ญ 0E0D	ฎ 0E0E	ฏ 0E0F
ฐ 0E10	ฑ 0E11	ฒ 0E12	ณ 0E13	ด 0E14	ต 0E15	ถ 0E16	ท 0E17	ธ 0E18	น 0E19	บ 0E1A	ป 0E1B	ผ 0E1C	ฝ 0E1D	พ 0E1E	ฟ 0E1F
ภ 0E20	ม 0E21	ย 0E22	ร 0E23	ฤ 0E24	ล 0E25	ฦ 0E26	ว 0E27	ศ 0E28	ษ 0E29	ส 0E2A	ห 0E2B	ฬ 0E2C	อ 0E2D	ฮ 0E2E	ฯ 0E2F
ะ 0E30	ั 0E31	า 0E32	ำ 0E33	ิ 0E34	ี 0E35	ึ 0E36	ื 0E37	ุ 0E38	ู 0E39	ฺ 0E3A	0E3B	0E3C	0E3D	0E3E	฿ 0E3F
เ 0E40	แ 0E41	โ 0E42	ใ 0E43	ไ 0E44	ๅ 0E45	ๆ 0E46	็ 0E47	่ 0E48	้ 0E49	๊ 0E4A	๋ 0E4B	์ 0E4C	ํ 0E4D	๎ 0E4E	๏ 0E4F
๐ 0E50	๑ 0E51	๒ 0E52	๓ 0E53	๔ 0E54	๕ 0E55	๖ 0E56	๗ 0E57	๘ 0E58	๙ 0E59	๚ 0E5A	๛ 0E5B	0E5C	0E5D	0E5E	0E5F
0E60	0E61	0E62	0E63	0E64	0E65	0E66	0E67	0E68	0E69	0E6A	0E6B	0E6C	0E6D	0E6E	0E6F
0E70	0E71	0E72	0E73	0E74	0E75	0E76	0E77	0E78	0E79	0E7A	0E7B	0E7C	0E7D	0E7E	0E7F

รูปที่ 2.42 ตาราง แสดงรหัส Unicode ภาษาไทย

<u>หลักการถอดรหัสตัวอักษร Unicode</u>

สำหรับรหัสตัวอักษรที่เป็น Unicode นั้น จะเห็นได้ว่าแต่ละตัวอักษรจะประกอบไปด้วยรหัส Code จำนวน 2 ไบท์เสมอ โดยตัวแรกเป็นตัวบอกรหัส Table ว่าเป็น Unicode ของภาษาใด โดยถ้าเป็นรหัส Unicode ของภาษาอังกฤษ ไบท์แรกจะมีค่าเป็น 00H ส่วนไบท์ที่2 จะเป็นรหัสตัวอักษร ซึ่งมีค่าตรงกันกับรหัส ASCII ส่วนภาษาไทยนั้น ไบท์แรกจะมีค่ารหัสเป็น 0EH ส่วนไบท์ที่ 2 จะเป็นรหัสตัวอักษร ซึ่งจากการทดสอบรับข้อความรหัสตัวอักษรจาก SMS พบว่า ถ้าใช้ภาษาอังกฤษอย่างเดียว รหัสของตัวอักษรจะเป็นแบบรหัส ASCII คือ 1 ตัวอักษร จะมีรหัส 1 ไบท์ แต่เมื่อมีการใช้ข้อความที่มีทั้งภาษาไทยและภาษาอังกฤษรวมกันพบว่ามีการเข้ารหัสตัวอักษรภาษาอังกฤษเป็นแบบ Unicode ด้วย

ดังนั้นพอสรุปได้ว่า ถ้าใช้ข้อความที่เป็นภาษาไทย ในระบบ SMS จะใช้รหัสตัวอักษรที่เป็นแบบ Unicode เสมอ แต่สำหรับภาษาอังกฤษนั้น ในระบบโทรศัพท์จะสามารถเลือกใช้ได้ทั้งระบบ Unicode และ ASCII Code โดยถ้าเป็น Unicode จะใช้รหัสตัวอักษรขนาด 2 Byte เช่นเดียวกันกับภาษาไทย โดยจะมีค่าอยู่ระหว่าง 0000H..007FH โดยมีรหัส 00H เป็นข้อมูลไบท์แรก ซึ่งถ้าข้อความเป็นภาษาอังกฤษอย่างเดียวรหัสของ

ตัวอักษรที่ใช้ใน SMS จะเป็นแบบ ASCII คือ ใช้รหัส ขนาด 1 ไบท์ โดยตัดรหัส 00H ไบท์แรกใน Unicode ทิ้งไป เช่น A แทนที่จะเป็นรหัส 0041H ก็จะเหลือเพียง 41H เป็นต้น แต่สำหรับข้อความที่มีทั้งภาษาไทยและภาษาอังกฤษรวมกันพบว่ามีการเข้ารหัส Code ตัวอักษรเป็นแบบ Unicode ด้วยเช่นเดียวกันกับภาษาไทย

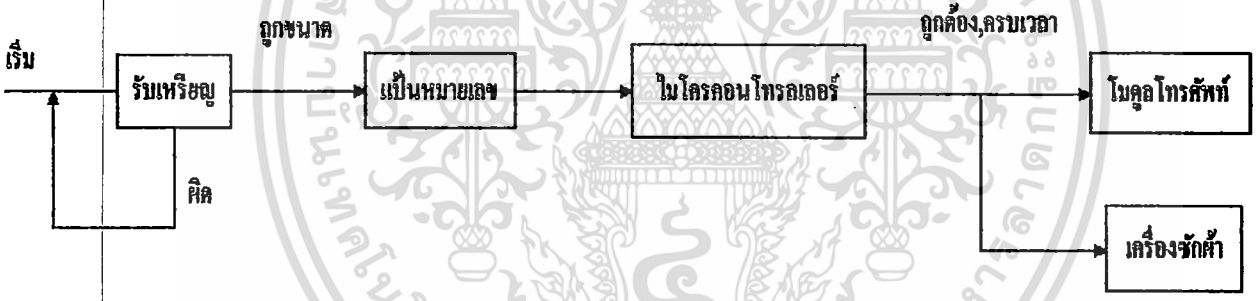
ดังนั้นในการถอดรหัสตัวอักษรต้องพิจารณาถึงจุดนี้ด้วย โดยมีข้อสังเกตว่า ถ้าพบรหัสตัวอักษรที่มีค่าระหว่าง 20H-7FH แสดงว่าเป็นรหัสแบบ ASCII สามารถนำไปแสดงผลได้เลย แต่ถ้าพบว่ารหัสเป็น 00H แสดงว่าเป็นรหัสแบบ Unicode ภาษาอังกฤษ ซึ่งรหัส Code ที่เป็นรหัสตัวอักษรจะอยู่ในรหัสข้อมูลไบท์ถัดไป และถ้าพบรหัสเป็น 0EH แสดงว่าเป็นรหัส Unicode ภาษาไทย ซึ่งรหัส Code ที่เป็นรหัสตัวอักษรจะอยู่ในข้อมูลไบท์ถัดไป เช่นเดียวกัน

บทที่ 3

หลักการและการออกแบบ

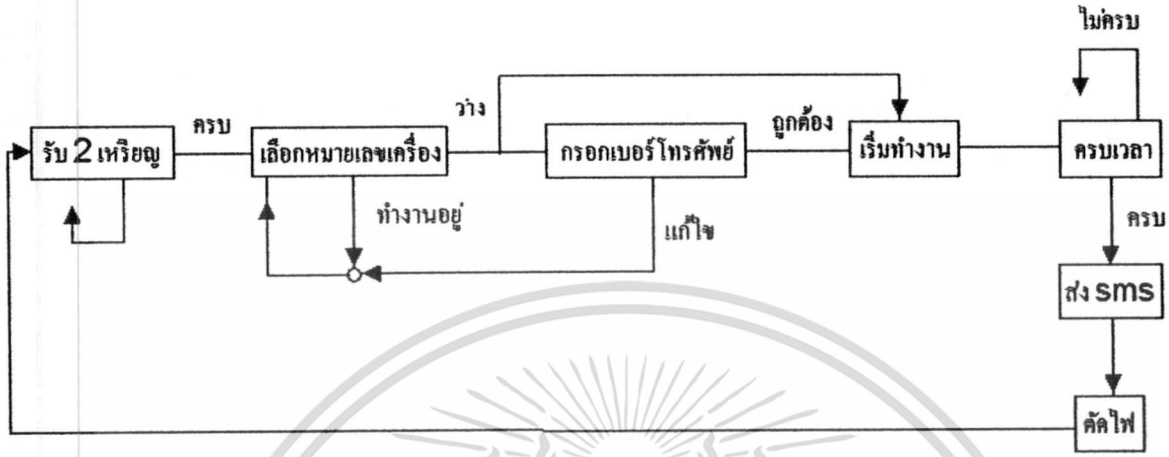
ในการออกแบบอุปกรณ์ตามแนวคิดที่จะแก้ไขปัญหาดังกล่าว จึงใช้โมดูลเครื่องหยอดเหรียญที่มีขายทั่วไปตามท้องตลาด มารวมกับความสามารถของ ไมโครคอนโทรลเลอร์ และโมดูลโทรศัพท์ มาสร้างเป็นเครื่องหยอดเหรียญอัจฉริยะ ที่สามารถควบคุมเครื่องใช้ไฟฟ้าและอุปกรณ์อำนวยความสะดวกอื่นๆที่ใช้ในเชิงธุรกิจการให้บริการ นอกจากความสามารถในการควบคุมเครื่องใช้ไฟฟ้าหลายๆเครื่องจากเครื่องหยอดเหรียญเพียงเครื่องเดียวแล้ว ยังมีความสามารถในการส่งข้อความสั้น (SMS) เข้าสู่โทรศัพท์มือถือของผู้ใช้เพื่อแจ้งเตือนข้อมูลต่างๆในการให้บริการอีกด้วย

สำหรับกรณีตัวอย่างที่ใช้ทดลองเครื่องหยอดเหรียญอัจฉริยะนี้ เป็นการทดลองนำมาควบคุมเครื่องซักผ้าหยอดเหรียญ โดยหลักการทำงานของเครื่องหยอดเหรียญอัจฉริยะเป็นไปตามไดอะแกรมการทำงานตามรูป



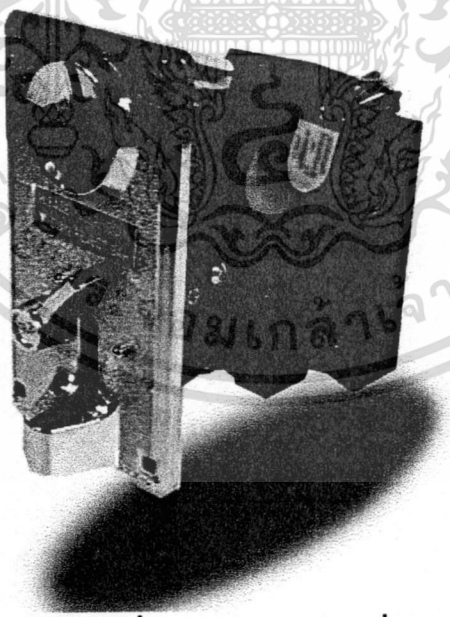
รูปที่ 3.1 ไดอะแกรมการทำงานของเครื่องหยอดเหรียญอัจฉริยะ

ในด้านการติดต่อกับผู้ใช้จะใช้การแสดงผลเป็น LCD สองบรรทัดเพื่อให้ผู้ใช้กรอกรายละเอียดต่างๆผ่านแป้นหมายเลข เช่น เบอร์โทรศัพท์ที่ผู้ใช้ต้องการให้เครื่องส่ง SMS ให้ และ หมายเลขของเครื่องใช้ไฟฟ้าที่ผู้ใช้ต้องการให้ทำงานเนื่องจากมีชุดควบคุมชุดเดียว จากเครื่องหยอดเหรียญอัจฉริยะ



รูปที่ 3.2 ไดอะแกรมการทำงานในส่วนของโปรแกรมและการติดต่อกับผู้ใช้

ในการพัฒนาเครื่องหยอดเหรียญอัจฉริยะนั้น อุปกรณ์หลักที่สำคัญที่สุดคือ โมดูลหยอดเหรียญที่ขายกันทั่วไปในท้องตลาด



รูปที่ 3.3 เครื่องหยอดเหรียญที่จะนำมาพัฒนาเป็นเครื่องหยอดเหรียญอัจฉริยะ

เครื่องหยอดเหรียญที่นำมาใช้พัฒนาเป็นเครื่องหยอดเหรียญชนิดที่หยอดเหรียญด้านหน้า รับเหรียญได้หลายขนาด หลายชนิด แต่ถ้าเลือกชนิดของเหรียญที่จะหยอดแล้ว จะหยอดได้ประเภทเดียวเท่านั้น ในการทดลองนี้จะใช้กับเหรียญชนิด 10 บาท

รูปที่ 3.4 เหรียญที่ใช้ในการทดลอง

โดยข้อมูลจำเพาะต่างๆของเหรียญ 10 บาท มีดังนี้ (ที่มา http://th.wikipedia.org)

น้ำหนัก	8.5	กรัม
เส้นผ่านศูนย์กลาง	26	มิลลิเมตร
ความหนา	2	มิลลิเมตร
ขอบ	เฟืองสลับแบบเรียบ	
ส่วนประกอบ		
วงแหวน:	คิวโปรนิกเกิล (75% Cu, 25% Ni)	
ตรงกลาง:	อะลูมิเนียมบรอนซ์ (92% Cu, 6% Al, 2% Ni)	

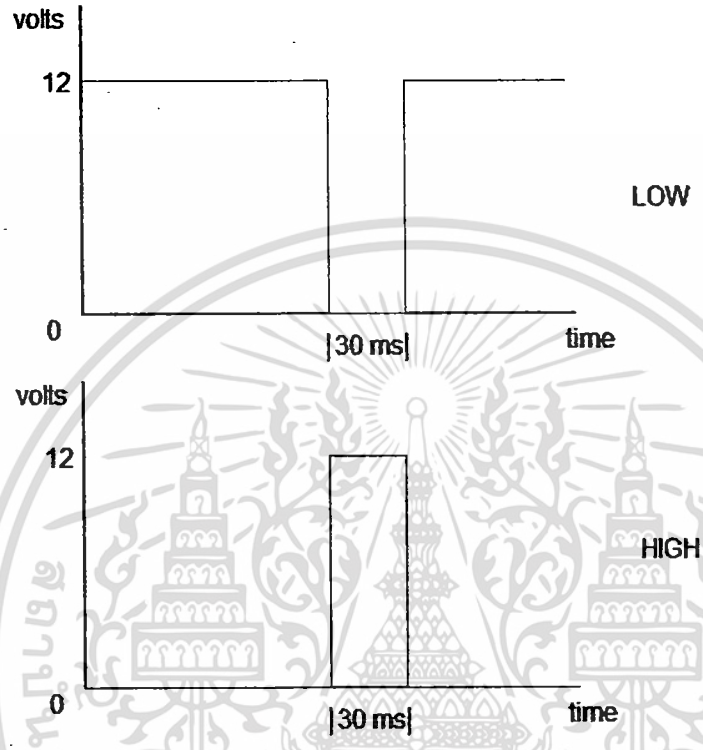
เครื่องหยอดเหรียญชนิดนี้ใช้เงื่อนไขในการตรวจสอบเหรียญปลอม 3 เงื่อนไข

1 ขนาดเส้นผ่าศูนย์กลางของเหรียญ
2 ขนาดความหนาของเหรียญ
3 วัสดุที่ใช้ทำเหรียญ(น้ำหนักของเหรียญ)

หากเหรียญที่หยอดลงไปในเครื่องหยอดเหรียญไม่ตรงตามเงื่อนไข เครื่องหยอดเหรียญจะถือว่าเหรียญนั้นเป็นเหรียญปลอมและทำการจ่ายออกจากเครื่องทันที

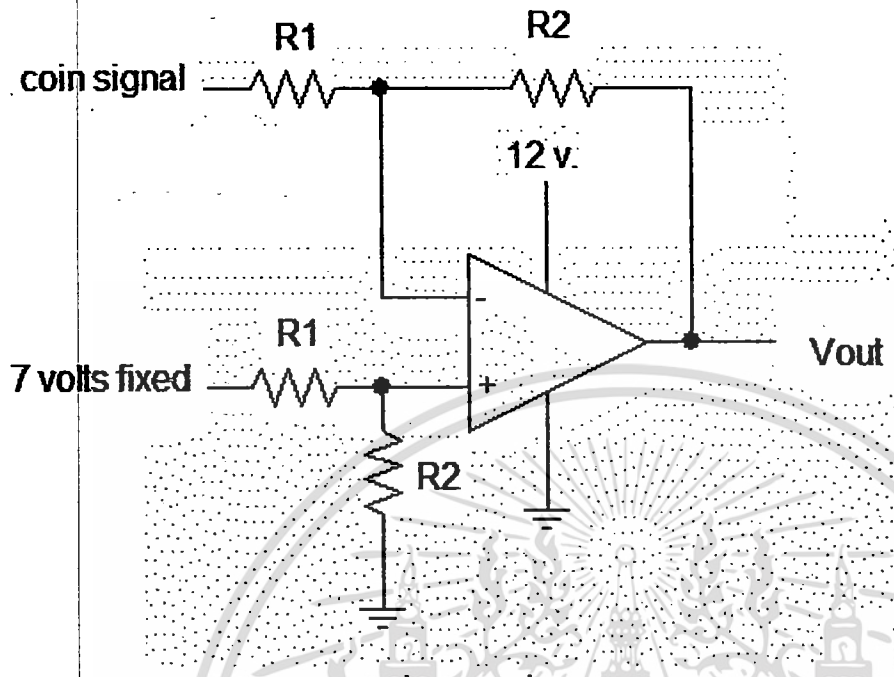
การใช้งานเครื่องหยอดเหรียญจึงต้องนำเหรียญตัวอย่าง 1 เหรียญใส่ไว้กับเครื่องหยอดเหรียญเสมอ เพื่อเป็นเหรียญอ้างอิง เมื่อมีเหรียญถูกหยอดลงในเครื่องและเครื่องสามารถตรวจสอบได้ว่าไม่ใช่เหรียญปลอม

เครื่องหยอดเหรียญจะส่งสัญญาณเป็น สัญญาณพัลส์ (pulse signal) ขนาด 12 โวลต์ เป็นช่วงเวลา 30 ms สัญญาณของเหรียญนี้มีสองลักษณะ คือแบบ high และ แบบ low



รูปที่ 3.5 สัญญาณของเหรียญแบบ LOW และแบบ HIGH

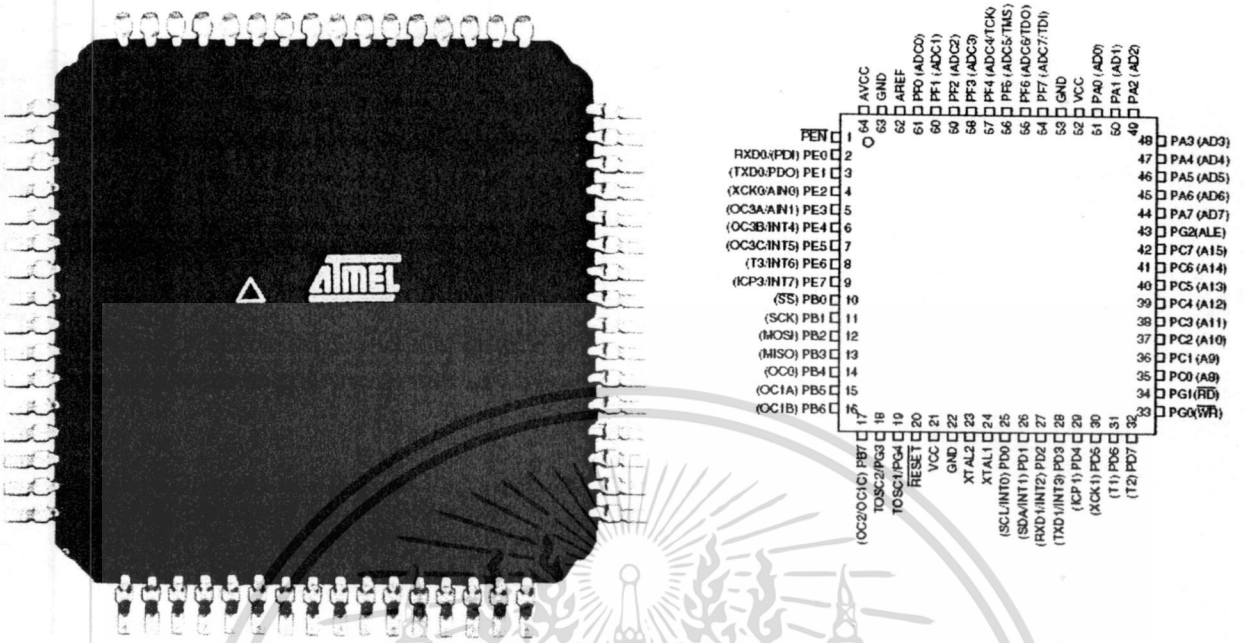
สัญญาณที่ได้จากเครื่องหยอดเหรียญยังไม่สามารถใช้กับ ไมโครคอนโทรลเลอร์ได้ เพราะเป็นสัญญาณขนาด 12 โวลต์ แต่ไมโครคอนโทรลเลอร์ต้องการสัญญาณขนาด 5 โวลต์ จึงจำเป็นต้องมีการแปลงสัญญาณให้เหลือ 5 โวลต์ก่อนที่จะใช้เป็นสัญญาณอินพุตให้กับไมโครคอนโทรลเลอร์ โดยวงจรที่ใช้แปลงสัญญาณนี้จะใช้วงจร ขยายผลต่างแรงดันแบบไม่กลับเฟส จาก Opamp



รูปที่ 3.6 วงจรที่ใช้ในการแปลงสัญญาณ

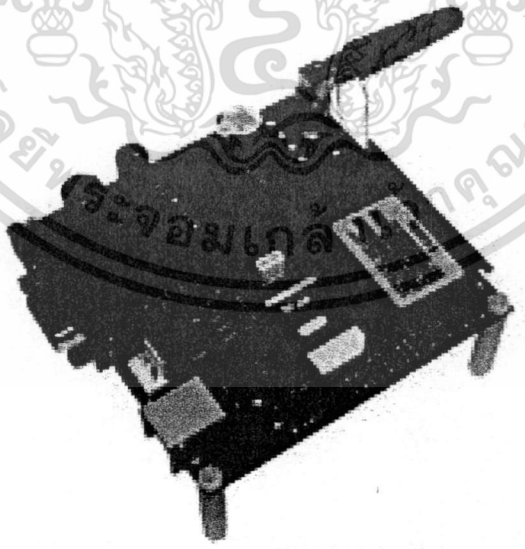
เมื่อได้สัญญาณที่สามารถใช้กับไมโครคอนโทรลเลอร์ แล้วการพัฒนาขั้นต่อไปคือการเขียนโปรแกรมลงบนไมโครคอนโทรลเลอร์ เพื่อประมวลผล และจัดการกับกระบวนต่างๆ เช่น สั่งเครื่องใช้ไฟฟ้าให้ทำงาน ตัดการทำงานของเครื่องใช้ไฟฟ้า แสดงผลในส่วนติดต่อกับผู้ใช้ รับข้อมูลต่างๆของผู้ใช้ เป็นต้น

ไมโครคอนโทรเลอร์ที่ใช้ในการพัฒนาเป็นไมโครคอนโทรลเลอร์ ในตระกูล AVR รุ่น ATMega128 ซึ่ง Atmega128 เป็นไมโครคอนโทรลเลอร์ 8 บิต แบบ CMOS ที่ใช้พลังงานต่ำ มีสถาปัตยกรรมภายในแบบ AVR RISC ประมวลผลคำสั่งได้ใน 1 รอบสัญญาณเวลา หรือ มีความเร็ว 1 MIPS ต่อ 1MHz มีระบบจัดการพลังงาน ทำให้ผู้พัฒนาสามารถเลือกระดับการใช้พลังงานของไมโครคอนโทรลเลอร์ได้ตามต้องการ



รูปที่ 3.7 ไมโครคอนโทรเลอร์ ATMega128

ความสามารถอีกอย่างหนึ่งของเครื่องหยอดเหรียญอัจฉริยะคือการส่งข้อความเข้าสู่โทรศัพท์มือถือของผู้ใช้ เพื่อแจ้งเตือนหรือ แจ้งข้อมูลต่างๆเกี่ยวกับการให้บริการของระบบ ดังนั้นในการพัฒนาจึงต้องมีอุปกรณ์อีกอย่างหนึ่งที่ทำหน้าที่ส่งข้อมูลต่าง สู่โทรศัพท์มือถือผู้ใช้ โดยผ่านทางโมดูลโทรศัพท์



รูปที่ 3.8 โมดูลโทรศัพท์ที่ใช้ในการพัฒนา

บทที่ 4

ผลการทดลอง

การทดลองที่ 1 ทดลองใช้เครื่องหยอดเหรียญควบคุมเครื่องใช้ไฟฟ้า

การทดลองนี้จะนำเครื่องหยอดเหรียญที่ได้พัฒนาไปต่อเข้ากับเครื่องใช้ไฟฟ้าต่างๆ เพื่อควบคุมการทำงานของเครื่องใช้ไฟฟ้า ให้ได้ตามเวลาที่กำหนด โดยเครื่องใช้ไฟฟ้าตัวอย่างที่นำมาใช้ในการทดลองนี้ คือ พัดลมไฟฟ้าเพื่อกำหนดช่วงเวลาในการใช้งานพัดลมไฟฟ้า ให้ทำงานและหยุดทำงานในเวลาที่กำหนด ในที่นี้คือ 2 นาที

ตารางแสดงผลการทดลอง

ครั้งที่	เวลา
1	125.23
2	120.59
3	121.43
4	130.96
5	121.27
6	120.31
7	122.92
8	124.12
9	126.43
10	120.23
เฉลี่ย	123.35

เปอร์เซ็นความคลาดเคลื่อนจากการทดลองปิดเปิดเครื่องใช้ไฟฟ้าในเวลาที่กำหนด มีค่าเท่ากับ 2.79 % ซึ่งเป็นค่าความคลาดเคลื่อนที่ยอมรับได้

การทดลองที่ 2 ทดลองส่งข้อความสั้นเข้าสู่โทรศัพท์เคลื่อนที่ของผู้ใช้

การทดลองนี้จะทำการส่งข้อความสั้นเข้าสู่โทรศัพท์เคลื่อนที่ของผู้ใช้ ที่ได้ทำการป้อนให้กับเครื่องหยอดเหรียญเมื่อเริ่มใช้บริการ เมื่อเครื่องใช้ไฟฟ้าทำงานเสร็จสิ้น ในการทดลอง ทำการทดลอง 120 ครั้ง ผลเป็นดังนี้

จากการทดลองส่งข้อความสั้น 120 ครั้ง ส่งข้อความเข้าโทรศัพท์เคลื่อนที่ของผู้ใช้บริการได้สำเร็จ 83 ครั้ง คิดเป็น เปอร์เซ็นความสำเร็จ 69.17 %

บทที่ 5

สรุปและอธิปรายผลการวิจัย

ในการวิจัยระบบการจัดการให้บริการของเครื่องใช้ไฟฟ้าแบบหยอดเหรียญ

1. สามารถอำนวยความสะดวกให้กับผู้รับบริการได้ สามารถขยายการให้บริการเกี่ยวกับเครื่องใช้ไฟฟ้าแบบหยอดเหรียญ ในที่นี้คือเครื่องซักผ้าแบบหยดเหรียญ เมื่อผู้ใช้บริการได้รับข้อความสั้นจะมารับผ้าเมื่อถึงเวลาลดปัญหาการลืมผ้าทำให้เกิดปัญหาเสื้อผ้าอับชื้น ผู้ที่มารอรับบริการท่านอื่นสามารถใช้เครื่องซักผ้าต่อได้ ช่วยเพิ่มรายได้ให้กับผู้ให้บริการ และผู้ให้ปริการสามารถรับข้อความสั้นเพื่อตรวจสอบจำนวนผู้ที่มาใช้บริการ ช่วยเพิ่มปลอดภัยจากการโจรกรรมทรัพย์สิน

2. ช่วยลดต้นทุนของผู้ให้บริการในกรณีที่ต้องการขยายการให้บริการ เช่น เพิ่มจำนวนเครื่องใช้ไฟฟ้าในระบบ เพราะไม่ต้องเสียต้นทุนในการซื้อเครื่องหยอดเหรียญเพื่อควบคุมเครื่องใช้ไฟฟ้าเพิ่ม