บทที่ 2
หลักการและทฤษฎี

2.1 ทฤษฎีรหัสวงเวียน

รหัสวงเวียนเป็นซับคลาส (subclass) หนึ่งของรหัสลิเนียร์บล็อค รหัสลิเนียร์บล็อคมีขั้นตอนในการเลือกเมตริกซ์ตัวกำเนิดสำหรับรหัสที่แก้ไขบิตที่ผิดเพียง 1 บิต ได้ง่าย แต่ขั้นตอนนี้ไม่สะดวกที่จะนำมาใช้ในการสร้างรหัสที่แก้ไขบิตที่ผิดมากกว่า 1 บิต รหัสวงเวียนจึงถูกนำมาใช้อย่างกว้างขวางเพราะมีโครงสร้างทางคณิตศาสตร์ที่เกื้ออำนวยให้สามารถออกแบบรหัสที่แก้ไขบิตที่ผิดมากกว่า 1 บิตได้ และการเข้ารหัสและการคำนวณซินโดรมก็สามารถสร้างขึ้นมาได้ง่ายๆ โดยใช้ตัว (shift register) ได้

รหัสวงเวียนสร้างคำรหัสโดยเลื่อนบิตข้อมูลผ่านตัวชิฟท์รีจิสเตอร์ เช่นถ้า

$$c = (c_0, c_1, c_2, \ldots, c_{n-1})$$

เป็นรหัสเวกเตอร์ของรหัสวงเวียน ถ้าเลื่อนรหัสเวกเตอร์ C เชิงวงเวียนไปทางขวา i ตำแหน่ง รหัสเวกเตอร์ที่ได้ก็ยังคงเป็นรหัสเวกเตอร์ของรหัสวงเวียนอีก ดังนี้

$$c^{(i)} = (c_{n-i}, c_{n-i+1}, \ldots, c_0, c_1, \ldots, c_{n-i-1})$$

คุณสมบัติของรหัสวงเวียนข้อนี้ ทำให้เราสามารถเขียนรหัสวงเวียนในรูปของโพลิโน-เมียล (polynomial) ที่มีกำลัง (degree) n-1 ได้

$$c(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}) \qquad (2.1)$$

สัมประสิทธิ์ของโพลิโนเมียลจึงเป็น '0' หรือ '1' และเป็นไปตามกฎของผลบวกและผลคูณของเลขฐาน 2 ดังนี้

$0+0=0 \qquad 0.0=0$

$0+1=1 \qquad 0.1=0$

$1+0=0 \qquad 1.0=0$

$1+1=0 \qquad 1.1=1$

ในกรณีที่เป็นผลบวกและผลคูณของรหัสโพลิโนเมียล 2 ตัว

$$p(x) = a_0 + a_1 x + \ldots + a_m x^m$$

$$q(x) = b_0 + b_1 x + \ldots + b_n x^n \quad , m > n$$

จะได้

$$p(x)q(x) = \sum_{i=0}^{m}\sum_{j=0}^{n} a_i b_j x^{i+j}$$

$$p(x)q(x) = (a_0 + b_0) + (a_1 + b_1)x + \ldots + (a_n + b_n)x^m + a_{n+1}x^{n+1} + \ldots + a_m x^m$$

คุณสมบัติที่น่าสนใจของรหัสโพลิโนเมียลคือ เมื่อ $x^l c(x)$ หารด้วย $x^n + 1$ เศษที่เหลือจะเท่ากับ $c^{(l)}(x)$ ซึ่งจะพิสูจน์ให้เห็นจริงดังต่อไปนี้

$$xc(x) = c_0 x + c_1 x^2 + c_2 x^3 + \ldots + c_{n-1}x^n$$

$$\begin{array}{r} c_{n-1} \\ x^n + 1 \overline{)\, c_{n-1}x^n + c_{n-2}x^{n-1} + \ldots + c_0 x} \\ \underline{c_{n-1}x^n + c_{n-1}} \\ c_{n-2}x^{n-1} + c_{n-3}x^{n-2} + \ldots + c_0 x + c_{n-1} \leftarrow \text{เศษที่เหลือ} \end{array}$$

ถ้าเลื่อนไป 1 ตำแหน่ง เศษที่เหลือก็จะได้ $c^{(1)}(x)$ ถ้าเลื่อนไป i ตำแหน่ง เศษที่เหลือ

$$c^{(l)}(x) = c_{n-l} + c_{n-i-1}x + \ldots + c_0 x^l + c_1 x^{l+1} + \ldots + c_{n-i-1}x^{n-i}$$

นั่นคือ

$$x^l c(x) = q(x)(x^{n+1}) + c^l(X) \qquad (2.2)$$

ทฤษฎีบท

ถ้า g(x) เป็นโพลิโนเมียลตัวกำเนิด (generator polynomial) ที่มีกำลัง (n-k) และเป็นแฟคเตอร์ของ $x^{n+1}g(x)$ และโพลิโนเมียลข้อมูล (data polynomial) d(x) ที่มีกำลัง (k-1) จะให้กำเนิดรหัส c(x) ของรหัสวงเวียน (n,k) ตามความสัมพันธ์ดังนี้

$$c(x) = d(x)g(x) \qquad (2.3)$$

พิสูจน์

พิจารณา k โพลิโนเมียล $g(x), xg(x), x^2g(x), \ldots, x^{k-1}g(x)$ ที่มีกำลัง (n-1) หรือน้อยกว่า พิจารณาโพลิโนเมียล

$$c(x) = d_0 g(x) + d_1 xg(x) + \ldots + d_{k-1}x^{k-1}g(x)$$

$$= d(x)g(x)$$

จะเห็นว่าเป็นโพลิโนเมียลที่มีกำลัง (n-1) หรือน้อยกว่าและเป็นผลคูณ (multiple) ของ g(x) ฉะนั้นเรามี 2^k โพลิโนเมียลที่ตรงกับ 2^k เวกเตอร์ข้อมูล(data vector)และรหัสเวกเตอร์ที่ตรงกับ 2^k โพลิโนเมียลจะกำเนิดรหัสวงเวียน (n,k)

เพื่อพิสูจน์ว่ารหัสดังกล่าวเป็นรหัสวงเวียน

ให้ $c(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1}$

เป็นรหัสโพลิโนเมียลในรหัสนี้

พิจารณา $xc(x) = c_0(x) + c_1x^2 + \ldots + x_{n-1}x^n$

$= c_{n-1}(x^n + 1) + (c_{n-1} + c_0x + \ldots + c_{n-2}x^{n-1})$

$= c_{n-1}(x^n + 1) + c^{(1)}(x)$

ในที่นี้ $c^{(1)}(x)$ เป็นการเลื่อนเชิงเวียน (cyclic shift) ของ c(x) 1 ตำแหน่ง เนื่องจาก xc(x) คือ xd(x)g(x) และ g(x) เป็นแฟคเตอร์ของ $x^n + 1, c^{(1)}(x)$ จะต้องเป็นผลคูณของ g(x) ด้วยและสามารถเขียนในรูป d(x)g(x) สำหรับเวกเตอร์ข้อมูล d บางค่า จึงเป็นการพิสูจน์ว่า $c^{(1)}(x)$ เป็นรหัสโพลิโนเมียล ฉะนั้นจากคำจำกัดความของรหัสวงเวียน รหัสใดที่กำเนิดโดย

$g(x), xg(x),\ldots, x^{k-1}g(x)$ เป็นรหัสวงเวียน (n,k)

โพลิโนเมียลตัวกำเนิด g(x) ของรหัสวงเวียน สามารถเขียนในรูปเป็นระบบ (systematic) ได้ดังนี้

$$c = (\underbrace{r_0, r_1, r_2, \ldots, r_{n-k-1}}_{\text{n-k บิตตรวจคู่ (parity check bits)}} \quad \underbrace{d_0, d_1, \ldots, d_{k-1}}_{\text{k บิตข้อมูล}}) \qquad (2.4)$$

ในที่นี้

$r(x) = r_0 + r_1x + r_2x^2 + \ldots + r_{n-k-1}x^{n-k-1}$

เป็นโพลิโนเมียลตรวจคู่ (parity check polynomial) สำหรับโพลิโนเมียลข้อมูล d(x)

โพลิโนเมียลตรวจคู่ r(x) เป็นเศษที่เกิดจากการหาร $x^{n-k}d(x)$ ด้วย g(x)

$x^{n-k}d(x) = q(x)g(x) + r(x)$

ในที่นี้ q(x) และ r(x) เป็นผลลัพธ์และเศษตามลำดับ

รหัสโพลิโนเมียล c(x) เกิดจาก

$$c(x) = r(x) + x^{n-k}d(x) \qquad (2.5)$$

พึงสังเกตว่าโครงสร้างของคำรหัส ประกอบด้วยดิจิต k ตัวแรก ที่ไม่ใช่ดิจิตข้อมูลหมด รหัสนี้จึงไม่ใช่รหัสมีระบบ (systematic code)

ในรหัสมีระบบ ดิจิต k ตัวแรกจะเป็นดิจิตข้อมูลและดิจิตที่เหลือ m = n-k จะเป็นดิจิตตรวจคู่ (parity-check digit)

2.2 การถอดรหัสวงเวียน

สมมติว่ารหัสเวกเตอร์ V ถูกส่งผ่านช่องสื่อสารที่มีเสียงรบกวน เวกเตอร์ที่รับได้ R อาจจะไม่ตรงกับรหัสเวกเตอร์ หน้าที่ของเครื่องถอดรหัสวงเวียนคือ หาเวกเตอร์ที่ส่งจากเวกเตอร์ที่รับได้

ก่อนอื่นเครื่องถอดรหัสจะทดสอบเวกเตอร์ที่รับได้ว่าเป็นรหัสเวกเตอร์หรือไม่ โดยการคำนวณซินโดรม (syndrome) ของคำที่รับได้ ถ้าซินโดรมเท่ากับศูนย์ เวกเตอร์ที่รับได้จะหารลงตัวโดยโพลิโนเมียลตัวกำเนิดของเวกเตอร์ที่รับได้คือ รหัสเวกเตอร์นั่นเอง เครื่องถอดรหัสจะยอมรับเวกเตอร์ที่รับได้เป็นรหัสเวกเตอร์ที่ส่งมา ถ้าซินโดรมไม่เท่ากับศูนย์แสดงว่าเกิดการผิดพลาดในการส่งซินโดรม S(x) ของเวกเตอร์ที่รับได้ R(x) คือเศษที่เกิดจากการหาร R(x) โดย g(x) ดังนี้

$$\frac{R(x)}{g(x)} = P(x) + \frac{S(x)}{g(x)} \qquad (2.6)$$

ในที่นี้ P(x) คือ ผลลัพธ์จากการหาร

ซินโดรม S(x) เป็นโพลิโนเมียลที่มีกำลัง n-k-1 หรือน้อยกว่า ให้ E(x) เป็นแพทเทอร์นความผิดพลาดที่เกิดจากช่องสื่อสาร จะได้

$$R(x) = V(x) + E(x)$$

และ

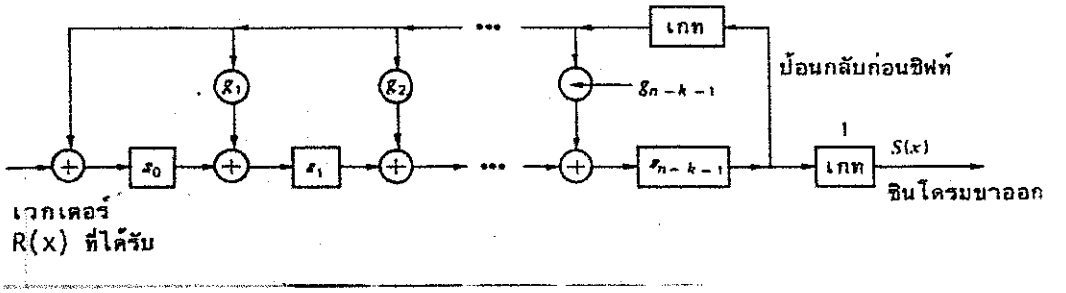
$$\frac{R(x)}{g(x)} = \frac{V(x)}{g(x)} + \frac{E(x)}{g(x)} \qquad (2.7)$$

แต่ V(x) = D(x)g(x) ในที่นี้ D(x) คือโพลิโนเมียลข้อมูล ฉะนั้นจากสมการ (2.6) และ (2.7) จะได้

$$E(x) = [P(x) + D(x)]\, g(x) + S(x) \qquad (2.8)$$

$$S(x) = \text{Rem}\,\frac{E(x)}{g(x)} \qquad (2.9)$$

ฉะนั้นซินโดรม R(x) เท่ากับเศษที่เกิดจากการหารแพทเทอร์นความผิดพลาดโดยโพลิโนเมียลตัวกำเนิด และซินโดรมจะมีข่าวสารเกี่ยวกับแพทเทอร์นความผิดพลาดที่สามารถใช้ในการแก้ไขความผิดพลาด (error correction) วงจรการหารที่ใช้ในการคำนวณซินโดรมนั้นแสดงในรูปที่ 2.1



รูปที่ 2.1 วงจรคำนวณซินโดรม (n-k) สำหรับรหัสวงเวียน

2.3 รหัสวงเวียนพิเศษ :รหัสโบเช – โชดโฮริ – ฮ็อกเคว็งเฮ็ม (BCH)

การออกแบบรหัสแก้ไขความผิดพลาดอย่างอ๊อพติมัมนั้นประกอบด้วยการออกแบบรหัสโดยมีความยาวของบล็อคสั้นที่สุด(n) ที่ค่าความยาวค่าหนึ่งที่ให้ของบล็อคข่าวสาร(k) และที่ค่าระยะทางสั้นที่สุด(d_{min}) ที่ต้องการสำหรับรหัสนั้น หรือพูดอีกนัยหนึ่งว่า สำหรับความยาวของรหัสค่าหนึ่งที่ให้(n) และประสิทธิภาพของรหัส (k/n) เราต้องการจะออกแบบให้มีค่า d_{min} มากที่สุดเท่าที่จะเป็นไปได้ นั่นคือเราต้องการจะออกแบบรหัสให้มีความสามารถในการแก้ไขความผิดพลาดได้ดีที่สุด รหัสวงเวียนเป็นรหัสที่การใช้งานมากที่สุด และมีประสิทธิภาพในการแก้ไขความผิดพลาดได้ดีที่สุดเท่าที่ทราบ

รหัส BCH เป็นชั้นรหัส (Class) ชั้นใหญ่ของรหัสวงเวียนที่ประกอบด้วย

ความยาวของบล็อค : $n = 2^{m-1}$

จำนวนบิตตรวจคู่ : $n-k <= mt$

ระยะทางที่สั้นที่สุด : $d_{min} >= 2t+1$

ในที่นี้ m และ t เป็นค่าอินทิเจอร์ใดๆที่เป็นบวก

โพลิโนเมียลตัวกำเนิด g(x) สำหรับรหัส BCH นั้น สามารถสร้างได้จากองค์ประกอบของ $2^{m-1} + 1$ ได้ ในการถอดรหัส BCH เรามักจะโปรแกรมวิธีการถอดรหัสกับเครื่องคอมพิวเตอร์ ในการสื่อสารข้อมูลในเครื่องคอมพิวเตอร์มักจะเป็นส่วนหนึ่งของเครือข่ายสื่อสารข้อมูล เราจึงนิยมใช้ซ็อฟแวร์ในการจำลองการทำงานของอัลกอลิทึมของการถอดรหัส BCH แทนที่จะใช้ฮาร์ดแวร์

2.4 DELPHI

โครงงานนี้ถูกเขียนขึ้นโดยใช้โปรแกรม Delphi 5.0 : Enterprise แบ่งออกเป็น 4 ส่วนสำคัญ แต่ละส่วนของโปรแกรมจะถูกแยกออกเป็น ยูนิต(Unit) เพื่อสะดวกในการแก้ไขและปรับปรุง

1. ยูนิตเมน(Unit Main) คือส่วนที่ใช้ในการติดต่อพอร์ต อาร์เอส-232 (RS-232) ส่วนนี้จะใช้ในการส่งข้อมูลและรับข้อมูล เพื่อทำให้เครื่องที่ทำหน้าที่ส่งข้อมูลและรับข้อมูลใช้โปรแกรมเดียวกันได้ แต่ในการส่งข้อมูลและรับข้อมูลต้องผลัดกันทำ คือเมื่อเครื่องหนึ่งส่งอีกเครื่องต้องรอรับข้อมูล

ยูนิตเมน(Unit Main) ประกอบโพรซีเจอร์(procedure)ทั้งหมด 25 โพรซีเจอร์

procedure FormCreate(Sender: Tobject); จะถูกใช้งานทันทีเมื่อทำการเปิดโปรแกรม ส่วนนี้จะใช้ในการกำหนดค่ามาตรฐานต่างๆของโปรแกรมเมื่อโปรแกรมถูกโหลดเข้าสู่หน่วยความจำ

procedure ButtonOpenClick(Sender: Tobject); ใช้ในการเปิดพอร์ตสื่อสารเพื่อเตรียมพร้อมสำหรับการส่งหรือรับข้อมูล

procedure ButtonCloseClick(Sender: Tobject); ใช้ในการปิดพอร์ตสื่อสารใช้เมื่อไม่มีการส่งข้อมูลออกทางพอร์ตแล้วก็ทำการปิดพอร์ตสื่อสาร

procedure ButtonResetClick(Sender: Tobject); ใช้ในการลบค่าต่างๆในช่องแสดงผลต่างๆ เมื่อเราเริ่มใช้งานแล้วจะมีการแสดงค่าต่างๆในช่องแสดงผล

procedure Comm1RxChar(Sender: Tobject; Count: Integer); เป็นส่วนที่ใช้ในการรับข้อมูล ส่วนนี้จะทำงานในเครื่องที่ทำหน้าที่รับข้อมูล โดยจะทำการรอรับค่าทีละบิทจนรับได้หมดทุกบิท แล้วจะทำการแบ่ง
บิตนั้นออกเป็นกลุ่มกลุ่มละ 15 บิต เพื่อนำข้อมูลนั้นไปเข้าสู่ส่วนของการถอดรหัส(Unit Decode)

procedure ButtonTransmitClick(Sender: Tobject); ใช้ในการส่งข้อมูลจากเครื่องส่งไปยังเครื่องรับ
หลังจากที่เราใส่ค่าที่ต้องการส่งลงในช่องรับค่าแล้ว

procedure MenuExit1Click(Sender: Tobject); ใช้ในการออกจากโปรแกรม เมื่อเลิกใช้งานโปรแกรมแล้ว

procedure N21MenuOpenClick(Sender: Tobject); เป็นการเรียกใช้ procedure ButtonCloseClick(Sender: Tobject) เป็นการปิดพอร์ตสื่อสารเมื่อไม่มีการส่งข้อมูล

procedure MenuTransmit1Click(Sender: Tobject); เป็นการเรียกใช้ procedure ButtonTransmitClick(Sender: Tobject); เป็นการส่งข้อมูลออกทางพอร์ตเมื่อใส่ค่าข้อมูลลงในช่องที่รับค่าแล้ว

procedure N1TestClick(Sender: Tobject); เป็นการทดสอบรหัส บีซีเอช(BCH) ว่าเป็นไปตาม ทฤษฎีหรือไม่อย่างไรคือตามทฤษฎีเมื่อมีบิตใดบิตหนึ่งเกิดผิดพลาดขึ้นทางเครื่องที่เป็นเครื่องรับจะทำการแก้ไขค่าให้อัตโนมัติ หรือ ผิดพลาดมากกว่า 2 บิต แต่น้อยกว่า 7 บิตแล้วทางฝ่ายเครื่อง

ที่รับจะตรวจพบว่าข้อมูลนั้นเกิดความผิดพลาดแต่ไม่สามารถแก้ไขได้ ถ้ามากกว่า 6 บิต แล้วทางเครื่องรับไม่สามารถตรวจสอบได้เลยว่าข้อมูลที่ได้รับถูกหรือผิด โดยการทดสอบนี้จะทำการแก้ไขค่าที่บิตที่ 12 ให้เป็นบิตที่ตรงกันข้าม หมายความว่าผิดบิตที่ 12 นั่นเอง แล้วจึงทำการส่งเพื่อให้ภาครับทำการแก้ไขว่าแก้ไขได้รึเปล่า

procedure N6MenuHelpClick(Sender: Tobject); แสดงส่วนช่วยเหลือการใช้ของโปรแกรม

procedure N14MenuAboutProjectClick(Sender: Tobject); แสดงส่วนที่เกี่ยวกับโครงงาน

procedure N12MenuOpenClick(Sender: Tobject); เป็นการเรียกใช้ procedure ButtonOpenClick(Sender: Tobject); เป็นการเปิดพอร์ตสื่อสาร

procedure N31MenuResetClick(Sender: Tobject); เป็นการเรียกใช้ procedure ButtonResetClick(Sender: Tobject); เป็นการลบค่าต่างๆที่แสดงในช่องแสดงค่า

procedure UpdateControls; เป็นการกำหนดค่าให้เป็นค่าที่พร้อมจะใช้งานได้

procedure HandleException(Sender: Tobject; E: Exception); เป็นส่วนที่ใช้ในการควบคุมโปรแกรม ถ้าโปรแกรมมีการเรียกใช้งานค่าที่ไม่ได้อยู่ในขอบเขตของโปรแกรม ส่วนนี้จะใช้ในการเตือนผู้ใช้

2. ยูนิตอัลกอลิทึม (Unit Algorithm) คือส่วนที่ใช้ในการเข้ารหัส 1 อักขระให้เป็นรหัสบีซีเอช(BCH) 15 บิต

ยูนิตอัลกอลิทึม(Unit Algorithm) ประกอบโพรซีเจอร์(procedure)ทั้งหมด 7 โพรซีเจอร์

Procedure DivPoly(Value:String;Di:String;Var Fdivpoly:string); เป็นอัลกอลิทึมที่ใช้ในการหารสมการโพลิโนเมียล

Procedure Syndrome(Data:String;Var Fsyndrome:String); เป็นอัลกอลิทึมที่ใช้ในการหาค่า เศษที่เกิดจากการหารสมการโพลิโนเมียล

Procedure Promp(P:String;Var Fpromp:String); เป็นอัลกอลิทึมที่ใช้ในการเพิ่มค่าบิตให้ครบ 15 บิต ถ้าค่าที่เกิดจากการเข้ารหัสแล้วค่าที่ได้ไม่เป็น 15 บิต ส่วนนี้จะทำการเพิ่ม 0 เข้าไปข้างหน้าจนเป็น 15 บิต

Procedure CrossPoly(Value:String;Cross:String;Var Fcrosspoly:String); เป็นอัลกอลิทึมที่ใช้ในการคูณสมการโพลิโนเมียล

Procedure Xnkdx(Binary:String;Var FXnkdx:String); เป็นอัลกอลิทึมที่ใช้ในการคูณค่าโพลิโนเมียลที่มีกำลังสูงสุดของโพลิโนเมียลตัวกำเนิดกับค่าโพลิโนเมียลที่เป็นข้อมูลที่จะทำการเข้ารหัส

ในอดีตกาล การสื่อสารด้วยพอร์ตอนุกรม RS-232 เป็นสิ่งที่มาตั้งแต่ต้น เริ่มตั้งแต่สมัยการโปรแกรมบนระบบปฏิบัติการ DOS ด้วยเครื่องที่ใช้ซีพียูระดับ 8 บิต เป็นต้นมา ในครั้งนั้นการเขียนโปรแกรมควบคุมพอร์ตสื่อสารชนิดนี้ต้องเขียนด้วยภาษาระดับต่ำ เช่นภาษาเครื่อง และผู้ที่เขียนจะต้องมีความรู้ในระดับฮาร์ดแวร์เกี่ยวกับอุปกรณ์ที่ใช้ค่อนข้างดี ต่อมาเมื่อไมโครซอฟต์ออกระบบปฏิบัติการวินโดวส์รุ่น 3.x ได้มีการเพิ่มเติมความสามารถในการเข้าถึงอุปกรณ์ชนิดนี้ โดยเตรียมชุดคำสั่งระดับสูงไว้สื่อสารกับพอร์ตโดยตรง แต่ยังคงต้องการความรู้ระดับฮาร์ดแวร์ในการกำหนดและติดตั้งพอร์ตสื่อสารให้สามารถใช้งานกับระบบ

ในปัจจุบัน ระบบปฏิบัติการวินโดวส์รุ่น 9x เป็นต้นไปได้ยกเลิกการติดต่อกับพอร์ตสื่อสารแบบเดิมที่ใช้ในวินโดวส์รุ่น 3.x และเพิ่มเติมดีไวซ์ไดรเวอร์ตัวใหม่เพื่อใช้ควบคุมพอร์ตสื่อสารนี้ โดยให้ผู้ใช้มองเห็นเป็นอุปกรณ์สื่อสารชนิดหนึ่ง และสามารถใช้โปรโตคอลมาตรฐานที่เตรียมไว้ติดต่อกับอุปกรณ์ชนิดนั้น โดยผู้ใช้ไม่จำเป็นต้องมีความรู้ระดับฮาร์ดแวร์ดังเดิม

ถึงแม้ระบบปฏิบัติการวินโดวส์รุ่นใหม่ จะเพิ่มเติมความสามารถในการควบคุมพอร์ตสื่อสารอนุกรม RS-232 ให้สามารถควบคุมและใช้งานได้ง่ายขึ้นก็ตาม แต่ในบางกรณี ผู้ใช้ต้องการใช้พอร์ตสื่อสารชนิดนี้ติดต่อกับอุปกรณ์ชนิดพิเศษอื่นๆ ที่วินโดวส์ไม่ได้กำหนดให้เป็นมาตรฐาน เช่น ติดต่อกับโฮสต์คอมพิวเตอร์ในแบบอิสระ, การติดต่อกับตู้ชุมสายโทรศัพท์ PABX (Private Automatic Branch Exchange), การต่อวงจรควบคุมอุปกรณ์ไฟฟ้า หรืออิเล็กทรอนิกส์อื่นๆ ทำให้ผู้ใช้มีความจำเป็นต้องเขียนโปรแกรมระดับต่ำกว่าในการสื่อสารกับอุปกรณ์ชนิดต่างๆนี้

<u>เตรียมการ</u>

ก่อนจะเริ่มทำการใดๆ ลองทำความรู้จักกับตัวฮาร์ดแวร์จริงชนิดนี้ก่อน ลักษณะของพอร์ตสื่อสารทางด้านกายภาพชนิดนี้เป็นรูปสี่เหลี่ยมคางหมูมุมมน ติดตั้งอยู่ด้านหลังเครื่องคอมพิวเตอร์ มีเข็มเล็กๆ ยื่นออกมาจากฐานจำนวน 9 เข็ม สำหรับพอร์ตสื่อสารชนิด 9 พิน เราเรียกพอร์ตชนิดนี้ว่า "หัวคอนเน็กเตอร์ ดีบีเก้าตัวผู้" (DB 9 Connector Male-Type) และอีกแบบมีเข็มจำนวน 25 เข็ม สำหรับพอร์ตสื่อสารชนิด 25 พิน เราเรียกว่า "หัวคอนเน็กเตอร์ ดีบียี่สิบห้า" (DB 25 Connector Male-Type) โดยปกติบนเครื่องคอมพิวเตอร์ทั่วไป จะกำหนดพอร์ตสื่อสารชนิด 9 พินเป็นพอร์ตสื่อสารที่ 1 (COM1) พอร์ตสื่อสารชนิด 25 พินเป็นพอร์ตสื่อสารที่ 2 (COM2) แต่ก็มีความเป็นไปได้ที่ทั้งสองพอร์ตจะเป็นชนิด 9 พินแบบเดียวกัน โดยปกติเราจะใช้พอร์ตสื่อสารชนิดที่ 1 ชนิด 9 พินนี้ต่อกับเมาส์ หรืออุปกรณ์ชี้นำอื่นๆ

การตรวจสอบทางด้านลอจิก บนระบบปฏิบัติการวินโดวส์ 9x ว่ามีการติดตั้งพอร์ตเหล่านี้ไว้หรือไม่ สามารถทำได้โดยตรวจสอบจากไอคอนซิสเต็มในหน้าต่างคอนโทรลพาเนล โดยเลือก

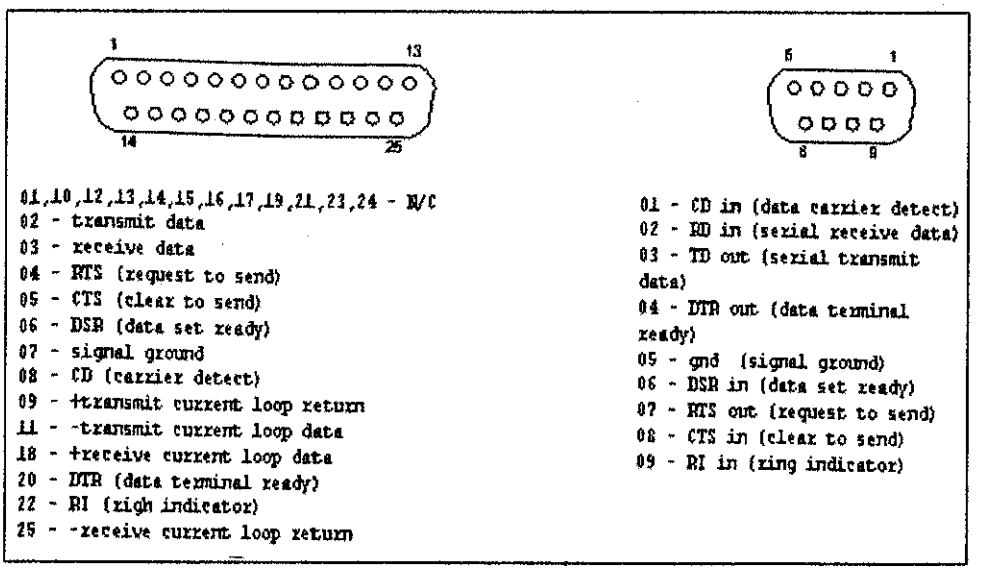
แสดงผลที่หน้า Device Manager และดับเบิลคลิกที่ "Port(COM&LPT)" ระบบ จะแสดงรายการ Communication Port ที่ติดตั้งไว้ในระบบ และหากดับเบิลคลิกที่รายการพอร์ตสื่อสารอีกครั้งระบบจะแสดงรายละเอียดทรัพยากรต่างๆ ของพอร์ตสื่อสารที่ใช้

ในกรณีที่ไม่พบพอร์ตสื่อสารใดๆ ติดตั้งอยู่ ขอให้ตรวจสอบจากหนังสือคู่มือของเครื่องคอมพิวเตอร์ที่ใช้ เพื่อทำการติดตั้งพอร์ตก่อนเริ่มดำเนินการต่อไป

สายสัญญาณ

สายสัญญาณเป็นปัจจัยที่สำคัญอย่างหนึ่งในการสื่อสานผ่านพอร์ตอนุกรม RS-232 เนื่องจากเป็นสิ่งที่เชื่อมโยงให้เครื่องต้นทางและปลายทางสามารถติดต่อถึงกันได้ การส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ให้ได้ผลลัพธ์ที่สมบูรณ์ ควรใช้สาย Null Modem ที่มีฉนวน (Shield) หุ้มสายสัญญาณ (Wire) ที่อยู่ด้านใน เพื่อป้องกันสัญญาณรบกวน แต่ในกรณีที่ไม่สามารถหาสายสัญญาณดังกล่าวได้ เราสามารถที่จะสร้างสายสัญญาณขึ้นเอง โดยต้องทำความเข้าใจก่อนว่าสายสัญญาณชนิดนี้มีอัตราความเสี่ยงต่อความผิดพลาดของข้อมูลค่อนข้างสูง เนื่องจากอาจเกิดจากสัญญาณรบกวนจากภายนอก และความไม่ได้มาตรฐานของสายที่ใช้ ดังนั้นขอแนะนำให้ใช้เพื่อทดสอบโปรแกรมเท่านั้น

การต่อสายสัญญาณ สิ่งที่ต้องทำก่อนคือ ทำความเข้าใจ "ขาสัญญาณ" ของหัวต่อ สำหรับขาสัญญาณที่นำไปใช้งานมีรายละเอียดดังนี้



รูปที่ 2.2 รายละเอียดของขาสัญญาณที่นำไปใช้งาน

ในการสร้างสายสัญญาณชั่วคราวนี้ สามารถใช้สายโทรศัพท์แบบ 4 เส้นสัญญาณ มีลักษณะเป็นสายกลม มีเส้นสัญญาณหุ้มปลอกสีแดง เหลือง เขียวและดำอยู่ภายใน การใช้สายสัญญาณประเภทนี้ต้องพึงระวังเรื่องสัญญาณรบกวนเป็นพิเศษ เนื่องจากไม่มีฉนวนป้องกันสัญญาณรบกวน

<u>หลักการ</u>

ในการใช้งานพอร์ตสื่อสาร ได้กำหนดลำดับขั้นตอนการดำเนินการออกเป็นส่วนย่อยๆ เพื่อใช้บริการพอร์ตสื่อสารกับระบบวินโดวส์ ทีลำดับการดำเนินงานดังต่อไปนี้

ตารางที่ 2.1 ลำดับการดำเนินงาน

เครื่องต้นทางทำหน้าที่ส่งข้อมูล	เครื่องปลายทางทำหน้าที่รับข้อมูล
1. เปิดพอร์ตสื่อสาร 2. กำหนดค่าเบื้องต้น 3. ตรวจสอบอินพุตบัฟเฟอร์ (Input Buffer (Rx)) ว่ามีข้อมูลอยู่หรือไม่ 4. ถ้ามี ทำการเคลื่อนย้ายข้อมูลออกจากบัฟเฟอร์ ดำเนินการประมวลผล และรอรับข้อมูลชุดต่อไป	1. เปิดพอร์ตสื่อสาร 2. กำหนดค่าเบื้องต้น 3. ตรวจสอบว่ามีเอาต์พุตบัฟเฟอร์ (Output Buffer (Tx)) เพียงพอที่จะส่งข้อมูลหรือไม่ 4. ถ้ามีดำเนินการส่งข้อมูล

กล่าวคือเมื่อดำเนินการเปิดพอร์ตสื่อสารแล้ว จะต้องดำเนินการกำหนดค่าเบื้องต้นให้กับพอร์ตสื่อสาร เช่น ความเร็วที่ใช้ในการติดต่อ ชนิดข้อมูล เป็นต้น และตรวจสอบบัฟเฟอร์ภายในว่ามีข้อมูลหรือเนื้อที่ว่างพอที่จะทำกิจกรรมการรับส่งข้อมูลหรือไม่ แล้วจึงดำเนินกิจกรรมตามที่ต้องการ

เป็นไปได้ยากเพราะเป็นการส่งข้อมูลในระยะใกล้ๆและสายที่ใช้ก็เป็นสายที่ป้องกันสัญญานรบกวนจากภายนอกได้ จึงได้ทำการทดสอบโดยการเข้ารหัสแล้วทำการแก้ค่าให้เป็นค่าที่ผิดก่อนที่จะทำการส่งออกทางพอร์ต อาร์เอส-232 ให้ทางภาครับทำการรับข้อมูลแล้วทำการตรวจสอบและแก้ไขเพื่อเป็นการทดสอบโปรแกรม