บทที่ 5

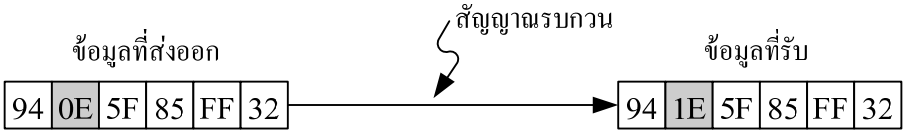
บูรณภาพของข้อมูล

ในการส่งข้อมูลในระบบสื่อสาร บางครั้งข้อมูลที่วงจรภาครับได้รับอาจจะไม่เหมือนกับข้อมูลที่ส่งมาจากวงจรภาคส่ง เนื่องมาจากการรบกวนชนิดต่างๆ เช่น สัญญาณรบกวน การแทรกสอด และการสูญเสียในอากาศ เป็นต้น โดยทั่วไประบบสื่อสารทุกระบบจะมีกระบวนการตรวจสอบข้อมูลที่ได้รับว่ามีความถูกต้องหรือไม่ ก่อนที่จะนำข้อมูลนี้ไปประมวลผลต่อไป หากข้อมูลที่ได้รับมีข้อผิดพลาดเกิดขึ้นก็จะทำการร้องขอไปยังวงจรภาคส่งให้ทำการส่งข้อมูลซ้ำใหม่อีกครั้งหนึ่ง ดังนั้นกระบวนการตรวจสอบข้อมูลจึงมีความสำคัญมากสำหรับระบบสื่อสารทุกระบบ รวมทั้งระบบ RFID

ฉะนั้นในบทนี้จะอธิบายกระบวนการตรวจสอบข้อมูลแบบต่างๆ ที่มีใช้ในระบบ RFID รวมทั้งอธิบายกระบวนการป้องกันการชนกันของข้อมูล (ซึ่งเกิดขึ้นในกรณีที่ป้าย RFID มากกว่าหนึ่งป้ายส่งข้อมูลไปยังเครื่องอ่านในเวลาเดียวกัน) เพื่อให้ผู้อ่านได้เข้าใจถึงความสามารถของระบบ RFID ในการรองรับฟังก์ชันการทำงานที่ซับซ้อนเหล่านี้ได้

5.1 กระบวนการผลรวมตรวจสอบ

ภาพที่ 5.1 แสดงการส่งข้อมูลในระบบสื่อสารที่มีข้อผิดพลาดเกิดขึ้นเนื่องมาจากสัญญาณรบกวน ดังนั้นโดยทั่วไปวงจรภาครับจะมีการนำกระบวนการผลรวมตรวจสอบ (checksum procedure) มาใช้ในการตรวจสอบว่ามีข้อผิดพลาดเกิดขึ้นในการส่งข้อมูลหรือไม่ ถ้ามีก็จะได้ดำเนินการแก้ไขโดยการขอให้วงจรภาคส่ง



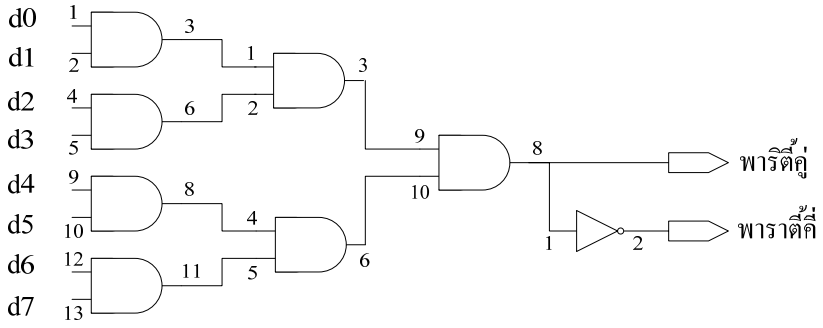
ภาพที่ 5.1 ข้อผิดพลาดที่เกิดขึ้นหว่างการส่งข้อมูลในระบบสื่อสาร

ทำการส่งบล็อกข้อมูล (data block) ที่เกิดข้อผิดพลาดมาให้ใหม่อีกครั้ง กระบวนการผลรวมตรวจสอบที่นิยมใช้งานในทางปฏิบัติ ได้แก่ การตรวจสอบพาริตี้ (parity check), การตรวจสอบ LRC (longitudinal redundancy check) หรือการตรวจสอบด้วยส่วนซ้ำซ้อนตามยาว, และการตรวจสอบ CRC (cyclic redundancy check) หรือการตรวจสอบด้วยส่วนซ้ำซ้อนแบบวน ซึ่งมีรายละเอียดดังนี้

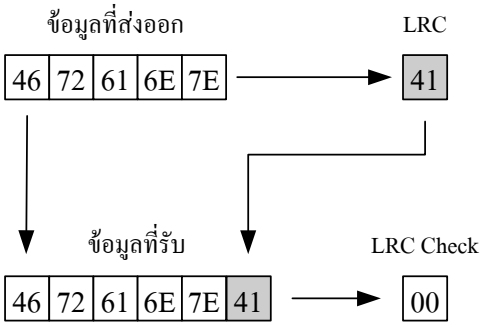
5.1.1 การตรวจสอบพาริตี้

การตรวจสอบพาริตี้เป็นกระบวนการผลรวมตรวจสอบที่นิยมใช้งานทั่วไป เพราะว่าเป็นวิธีการตรวจสอบข้อมูลที่ง่ายและมีความซับซ้อนน้อย ในทางปฏิบัติการตรวจสอบพาริตี้แบ่งออกเป็น 2 ประเภท คือการตรวจสอบพาริตี้แบบคู่ (even parity) และการตรวจสอบพาริตี้แบบคี่ (odd parity) โดยก่อนส่งข้อมูลแต่ละไบต์ (จำนวน 8 บิต โดยที่แต่ละบิตมีค่าที่เป็นไปได้คือ 0 และ 1) จะมีการเพิ่มบิตพาริตี้จำนวน 1 บิตเข้าไปต่อท้ายข้อมูลหนึ่งไบต์นั้น จากนั้นจึงส่งข้อมูลจำนวน 9 บิตไปยังวงจรภาครับ บิตพาริตี้จะมีค่าเป็น 0 หรือ 1 ขึ้นอยู่กับวิธีการตรวจสอบพาริตี้ว่าเป็นแบบคี่หรือแบบคู่ กล่าวคือถ้าใช้การตรวจสอบพาริตี้แบบคู่ ผลรวมของบิตข้อมูลทั้ง 9 บิตจะต้องเป็นเลขคู่ แต่ถ้าใช้การตรวจสอบพาริตี้แบบคี่ ผลรวมของบิตข้อมูลทั้ง 9 บิตก็จะต้องเป็นเลขคี่ ตัวอย่างเช่น กำหนดให้ข้อมูลหนึ่งไบต์คือ “E5h” ในรูปของเลขฐานสิบหกหรือ “11100101” (1 + 1 + 1 + 0 + 0 + 1 + 0 + 1 = 5) ในรูปของเลขฐานสอง ดังนั้นถ้าใช้การตรวจสอบพาริตี้แบบคู่จะได้บิตพาริตี้ที่มีค่าเท่ากับ 1 (5 + 1 = 6 เป็นเลขคู่) ในทางกลับกันถ้าใช้การตรวจสอบพาริตี้แบบคี่จะได้บิตพาริตี้ที่มีค่าเท่ากับ 0 (5 + 0 = 5 เป็นเลขคี่)

จากที่อธิบายข้างต้นพบว่าการตรวจสอบพาริตี้มีหลักการทำงานที่ง่ายและสามารถสร้างขึ้นได้โดยอาศัยตัวดำเนินการ XOR (exclusive OR) ดังแสดงในภาพที่ 5.2 อย่างไรก็ตามข้อเสียของการตรวจสอบพาริตี้คือการตรวจสอบพาริตี้จะสามารถตรวจหา (detect) ว่ามีข้อผิดพลาดเกิดขึ้นในการส่งข้อมูลได้ ก็ต่อเมื่อจำนวนบิตข้อมูลที่เกิดข้อผิดพลาดมีค่าเป็นเลขคี่เท่านั้น เพราะถ้าจำนวนบิตข้อมูลที่เกิด



ภาพที่ 5.2 หลักการทำงานของการตรวจสอบพาริตี้



ภาพที่ 5.3 หลักการทำงานของการตรวจสอบ LRC

ข้อผิดพลาดมีค่าเป็นเลขคู่แล้ว ข้อผิดพลาดที่เกิดขึ้นก็จะหักล้างกันทำให้วิธีการตรวจสอบพาริตี้นี้ไม่สามารถตรวจหาข้อผิดพลาดที่เกิดขึ้นในระบบได้

5.1.2 การตรวจสอบ LRC

การตรวจสอบ LRC จัดว่าเป็นกระบวนการผลรวมตรวจสอบที่อาศัยตัวดำเนินการ XOR และมีขั้นตอนการคำนวณที่ง่ายและรวดเร็วตามภาพที่ 5.3 กล่าวคือบล็อกข้อมูล (ประกอบด้วยข้อมูลจำนวน 5 ไบต์) จะถูกนำมาใช้ในการคำนวณหาค่า LRC หรือข้อมูลส่วนเกิน (redundancy data) จำนวน 1 ไบต์ เพื่อจะได้เพิ่มเข้าไปในบล็อกข้อมูลก่อนที่จะส่งออกไปยังวงจรภาครับ ซึ่งวิธีการคำนวณทำได้ง่ายโดยการนำข้อมูลไบต์ที่ 1 มา XOR กับข้อมูลไบต์ที่ 2 โดยผลลัพธ์ที่ได้ก็จะถูกนำมา XOR กับข้อมูลไบต์ที่ 2 จากนั้น

ก็ทำผลลัพธ์ที่ได้มา XOR กับข้อมูลไบต์ที่ 3 ทำแบบนี้ไปเรื่อยๆ จนครบจำนวนข้อมูลภายในบล็อกข้อมูล นั้นก็จะได้ค่า LRC ที่จะส่งไปพร้อมกับบล็อกข้อมูล

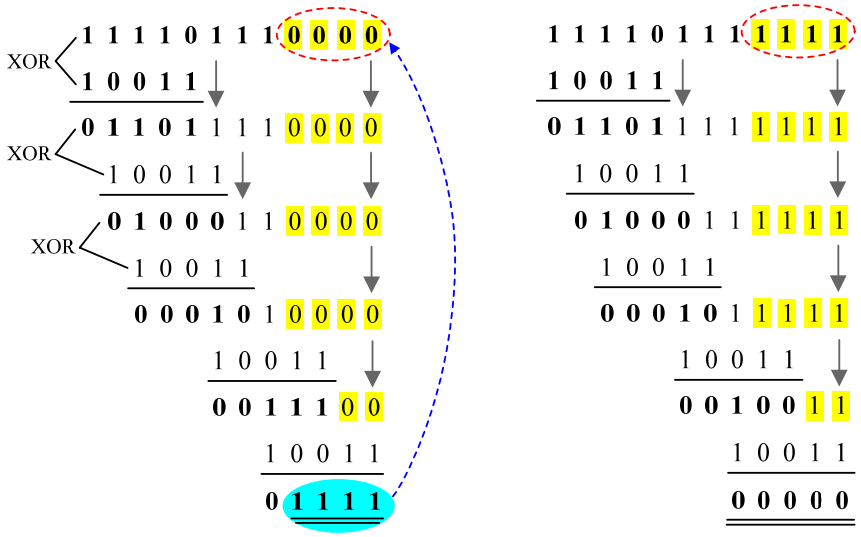
การตรวจสอบข้อผิดพลาดที่วงจรภาครับทำได้ง่ายเช่นกัน โดยการคำนวณหาค่า LRC จากบล็อกข้อมูลที่ได้รับ จากนั้นก็นำค่า LRC ที่คำนวณได้มา XOR กับค่า LRC เดิมที่ส่งมาจากวงจรภาคส่ง ถ้าผลลัพธ์ที่ได้เป็นค่า “00h” ก็แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้นในการส่งข้อมูล แต่ถ้าผลลัพธ์ที่ได้เป็นค่าอื่นก็แสดงว่ามีข้อผิดพลาดเกิดขึ้นในการส่งข้อมูล อย่างไรก็ตามถึงแม้ว่าการตรวจสอบ LRC เป็นวิธีการที่ง่ายและรวดเร็ว แต่ก็มีข้อเสียเช่นเดียวกับการตรวจสอบพาริตี้ นั่นคือในกรณีที่บล็อกข้อมูลที่ได้รับมีข้อผิดพลาดหลายบิต ก็อาจจะเป็นไปได้ที่ข้อผิดพลาดเหล่านี้อาจจะเกิดการหักล้างกันทำให้วิธีการตรวจสอบ LRC นี้ไม่สามารถตรวจหาข้อผิดพลาดได้ โดยทั่วไปวิธีการตรวจสอบ LRC นิยมนำมาใช้กับงานประยุกต์ที่ต้องการหาผลรวมตรวจสอบที่รวดเร็วของบล็อกข้อมูลที่มีจำนวนข้อมูลน้อยๆ เช่น 32 ไบต์ เป็นต้น

5.1.3 การตรวจสอบ CRC

การตรวจสอบ CRC เป็นวิธีการนำข้อมูลชุดหนึ่งมาคำนวณเป็นตัวเลขตัวหนึ่งเพื่อใช้ในการตรวจสอบว่าข้อมูลชุดนั้นมีการเปลี่ยนแปลงหรือไม่ หลังจากผ่านกระบวนการอย่างใดอย่างหนึ่ง เช่น การจัดเก็บและอ่านข้อมูลภายในอุปกรณ์ฮาร์ดดิสก์ไดรฟ์ การส่งข้อมูลผ่านระบบเครือข่าย หรือการรับส่งข้อมูลในระบบ RFID เป็นต้น โดยทั่วไปการตรวจสอบข้อมูลด้วยวิธี CRC จะมีความถูกต้องแม่นยำมากกว่าวิธีการหาผลรวมตรวจสอบ (checksum) มาก นอกจากนี้การตรวจสอบ CRC สามารถใช้งานได้ดีกับบล็อกข้อมูลขนาดใหญ่ที่มีจำนวนข้อมูลหลายๆ ไบต์ได้อย่างมีประสิทธิภาพ

หลักการทำงานของการตรวจสอบ CRC จะเริ่มจากการจัดข้อมูลให้อยู่ในรูปของพหุนาม (polynomial) เช่น ข้อมูล “10111” จะมีรูปพหุนามคือ x^4+x^2+x+1 จากนั้นนำข้อมูลที่ต้องการจะคำนวณหาผลรวมตรวจสอบ CRC มาหารด้วยพหุนามตัวกำเนิด (generating polynomial) โดยเศษที่ได้จากการหารก็คือค่า CRC นั่นเอง โดยทั่วไปถ้าต้องการค่า CRC ขนาด n บิต ก็จะต้องใช้พหุนามตัวกำเนิดในรูปของเลขฐานสองที่มีจำนวน $n+1$ บิต ในทางปฏิบัติการหาค่า CRC สามารถทำได้ง่ายโดยอาศัยตัวดำเนินการ XOR ซึ่งมีวิธีการคำนวณ ตามตัวอย่างที่แสดงต่อไปนี้

กำหนดให้ข้อมูลที่ต้องการหาค่า CRC มีจำนวน 8 บิต คือ “11110111” (หรือในรูปของเลขฐานสิบหกคือ “F7h”) และใช้พหุนามตัวกำเนิดคือ x^4+x+1 หรือ “10011” ในรูปของเลขฐานสอง (แสดงว่า

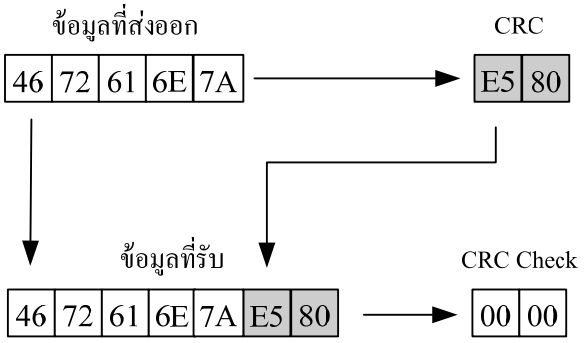


(ก) การคำนวณหาค่า CRC (ข) การตรวจสอบค่า CRC

ภาพที่ 5.4 ตัวอย่างขั้นตอน (ก) การคำนวณหาค่า CRC และ (ข) การตรวจสอบค่า CRC

หาข้อมูล CRC ขนาด $n = 4$ บิต) ขั้นตอนแรกให้นำข้อมูลทั้งหมดมาเรียงตามแนวนอนและเพิ่มค่าเริ่มต้นเท่ากับจำนวนบิตของค่า CRC ที่ต้องการ นั่นคือ 4 บิตต่อท้ายบิตข้อมูล ซึ่งในตอนแรกค่าเริ่มต้นจะมีค่าเท่ากับค่าศูนย์ทั้งหมด[10] ดังแสดงในภาพที่ 5.4 (ก) จากนั้นนำพหุนามตัวกำเนิดมาวางไว้ใต้ข้อมูล โดยให้บิตข้อมูลตัวแรกอยู่ในตำแหน่งที่ตรงกัน แล้วทำการ XOR ก็จะได้ผลลัพธ์เป็น “011011110000” ถ้าบิตข้อมูลตัวแรกทางด้านซ้ายมือของผลลัพธ์ที่ได้มีค่าเป็น 0 ก็ให้เลื่อนพหุนามตัวกำเนิดไปทางด้านขวา 1 บิต จนกระทั่งพบกับบิตข้อมูลของผลลัพธ์ที่ได้มีค่าเป็น 1 แล้วจึงจะทำการ XOR กันอีกครั้ง ทำการคำนวณแบบนี้ไปเรื่อยๆ จนกระทั่งบิตด้านขวาสุดของพหุนามตัวกำเนิดอยู่ในตำแหน่งเดียวกันกับบิตตัวสุดท้ายของข้อมูล ก็ให้ทำการ XOR กันอีกเป็นครั้งสุดท้าย ก็จะได้ผลลัพธ์เป็นค่า CRC จำนวน 4 บิตตามที่ต้องการ ซึ่งในที่นี้ค่า CRC คือ “1111”

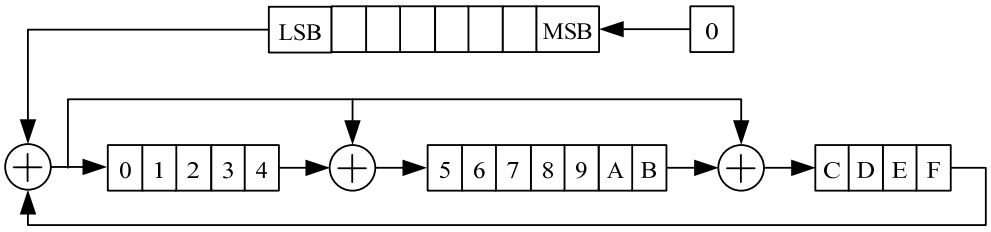
[10] การคำนวณหาค่า CRC ของข้อมูลที่ต่อกันหลายๆ บล็อกจะถูกกระทำทีละบล็อก โดยในตอนแรกค่าเริ่มต้นมีค่าเท่ากับค่าศูนย์ทั้งหมดสำหรับการคำนวณหาค่า CRC ของบล็อกข้อมูลที่หนึ่ง จากนั้นค่า CRC ที่ได้จะถูกนำมาใช้เป็นค่าเริ่มต้นของบล็อกข้อมูลที่สอง และค่า CRC ที่ได้ก็จะถูกนำมาใช้เป็นค่าเริ่มต้นของบล็อกข้อมูลที่สาม ทำแบบนี้ไปเรื่อยๆ จนถึงบล็อกข้อมูลสุดท้าย ก็จะได้ค่า CRC ของข้อมูลทั้งหมด



ภาพที่ 5.5 ตัวอย่างขั้นตอนการตรวจสอบ CRC ของข้อมูลที่มีหลายๆ บล็อก

เมื่อต้องการส่งบล็อกข้อมูลไปยังวงจรภาครับ ก็จะนำค่า CRC ที่คำนวณได้มาต่อท้ายบล็อกข้อมูลก่อนที่จะส่งไปยังวงจรภาครับ ในการตรวจสอบว่าข้อมูลที่วงจรภาครับได้รับว่ามีข้อผิดพลาดเกิดขึ้นหรือไม่ ก็สามารถทำได้โดยการคำนวณหาค่า CRC ของบล็อกข้อมูลที่ได้รับ โดยข้อมูลที่ได้รับจะไม่มีข้อผิดพลาดก็ต่อเมื่อผลลัพธ์ที่ได้จากการคำนวณหาค่า CRC มีค่าเป็นค่าศูนย์ทั้งหมด ดังแสดงในภาพที่ 5.4 (ข) นอกจากนี้ภาพที่ 5.5 แสดงตัวอย่างขั้นตอนการตรวจสอบ CRC ของข้อมูลที่มีหลายๆ บล็อก ซึ่งสามารถตรวจสอบความถูกต้องของข้อมูลได้อย่างรวดเร็ว และวงจรภาครับไม่ต้องทราบล่วงหน้าว่าค่า CRC ที่ส่งมาจากวงจรภาคส่งคือค่าอะไร กล่าวคือถ้าค่า CRC ที่วงจรภาครับคำนวณได้มีค่าเท่ากับค่าศูนย์ทั้งหมด แสดงว่าข้อมูลที่รับมาทั้งหมดไม่มีข้อผิดพลาดเกิดขึ้น

ในทางปฏิบัติการตรวจสอบ CRC เพื่อคำนวณหาค่าศูนย์จะมีความซับซ้อนน้อยกว่าการเปรียบเทียบค่าผลรวมตรวจสอบ จึงเป็นเหตุผลทำให้การตรวจสอบ CRC เป็นที่นิยมใช้งานในหลายๆ งานประยุกต์ ข้อดีของการตรวจสอบ CRC คือมีความน่าเชื่อถือสูงในการตรวจหาข้อผิดพลาดของข้อมูล โดยข้อมูล CRC ขนาด 16 บิต สามารถใช้ในการตรวจสอบบล็อกข้อมูลขนาดความยาว 4 กิโลไบต์ ได้อย่างมีประสิทธิภาพ อย่างไรก็ตามเนื่องจากข้อมูลที่รับส่งในระบบ RFID มีขนาดความยาวน้อยกว่า 4 กิโลไบต์ จึงทำให้สามารถใช้งานการตรวจสอบ CRC แบบ 8 หรือ 12 บิตก็เพียงพอ ตัวอย่างพหุนามตัวกำเนิดที่ใช้ในการหาค่า CRC แบบ 8 บิต คือ $x^8 + x^4 + x^3 + x^2 + 1$ และ CRC แบบ 16 บิต (ตามมาตรฐาน CCITT) คือ $x^{16} + x^{12} + x^5 + 1$ [40] ภาพที่ 5.6 แสดงโครงสร้างของวงจรที่ใช้ในการสร้างข้อมูล CRC แบบ 16 บิต (CCITT) ซึ่งจะประกอบไปด้วยรีจิสเตอร์แบบเลื่อน (shift register) และตัวดำเนินการ XOR



ภาพที่ 5.6 โครงสร้างของวงจรที่ใช้ในการสร้างข้อมูล CRC แบบ 16 บิต (CCITT)

5.2 กระบวนการป้องกันการชนกันของข้อมูล

โดยทั่วไปการทำงานของระบบ RFID มักจะอยู่ในสถานะการที่ป้าย RFID หลายป้ายอยู่ในพื้นที่การอ่าน (reader's field) ของเครื่องอ่านเพียงตัวเดียว ซึ่งในกรณีนี้สามารถจัดลักษณะการทำงานของระบบ RFID ออกได้เป็น 2 แบบคือ

1) การกระจาย (broadcasting) หมายถึงการส่งข้อมูลจากเครื่องอ่านเพียงหนึ่งเครื่องไปยังป้าย RFID หลายๆ ป้ายในเวลาเดียวกัน ดังแสดงในภาพที่ 5.7 (ก)

2) การเข้าถึงหลากหลาย (multi-access) หมายถึงการส่งข้อมูลจากป้าย RFID หลายๆ ป้ายไปยังเครื่องอ่านเพียงหนึ่งเครื่องในเวลาเดียวกัน ดังแสดงในภาพที่ 5.7 (ข)

ในทางปฏิบัติประสิทธิภาพของการส่งข้อมูลในระบบการสื่อสารจะถูกกำหนดด้วย "ความจุของช่องสัญญาณ (channel capacity)" ซึ่งเป็นพารามิเตอร์ที่บ่งบอกถึงอัตราการส่งข้อมูลสูงสุดของช่องสัญญาณที่ยอมรับได้ ดังนั้นช่องสัญญาณควรจะต้องถูกจัดสรรให้กับป้าย RFID แต่ละป้าย เพื่อให้สามารถส่งข้อมูลมายังเครื่องอ่านได้โดยไม่เกิดปัญหาเรื่องการแทรกสอดหรือการชนกัน (collision) ของข้อมูล โดยทั่วไปป้าย RFID ไม่สามารถถอดรหัสหรืออ่านกลุ่มข้อมูล (data packet) ที่ส่งโดยป้าย RFID อื่นไปยังเครื่องอ่านได้ จึงทำให้ป้าย RFID ไม่สามารถตรวจสอบได้ว่ามีป้าย RFID อื่นอยู่ในพื้นที่การอ่านหรือไม่ และเมื่อป้าย RFID มากกว่าหนึ่งป้ายส่งข้อมูลไปยังเครื่องอ่านในเวลาเดียวกัน ก็จะก่อให้เกิดการชนกันของข้อมูล (ทำให้ต้องเริ่มส่งข้อมูลใหม่อีกครั้งซึ่งจะเสียเวลามาก) ฉะนั้นจึงเป็นหน้าที่ของเครื่องอ่านที่จะต้องมีกระบวนการป้องกันการชนกันของข้อมูล (anti-collision) เพื่อใช้ในการจัดสรรช่องสัญญาณให้กับป้าย RFID ที่ต้องการส่งข้อมูลมายังเครื่องอ่าน