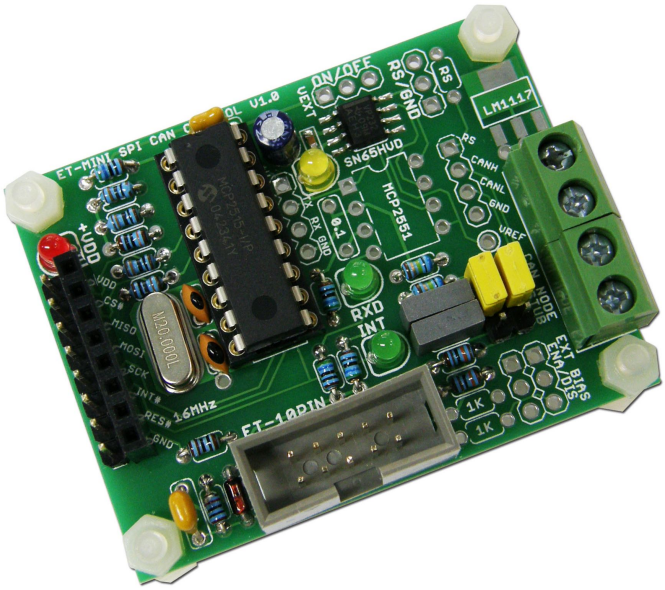
ET-MINI SPI CAN CONTROL V1.0



CAN หรือ **C**ontroller **A**rea **N**etwork เป็นระบบสื่อสารซึ่งได้รับการ คิดค้น พัฒนา ขึ้นมาใช้งานในแวดวงอุตสาหกรรมการผลิต และ อุตสาหกรรมยานยนต์ เป็นเวลาหลายสิบปี มาแล้ว แต่ในระยะแรกๆยังเป็นที่รู้จักกันในวงแคบๆ เฉพาะกลุ่มเท่านั้น แต่ในปัจจุบันระบบการสื่อสาร CAN เริ่มกับมามีความคึกคักและถูกนำมาพัฒนาใช้งานกันอย่างแพร่หลายมากขึ้น โดยเฉพาะอย่างยิ่งในอุตสาหกรรมยานยนต์ การสื่อสารแบบ CAN เริ่มถูกใช้กำหนดเป็นมาตรฐาน สำหรับสื่อสารกับ สมองกล ECU ของยานยนต์รุ่นใหม่ๆ รวมทั้งในแวดวงอุตสาหกรรมอื่นๆ ระบบการสื่อสารแบบ CAN ก็ถูกนำไปประยุกต์ใช้งานอย่างหลากหลายเช่นเดียวกัน ซึ่งในแวดวงของ ไมโครคอนโทรลเลอร์เอง ผู้พัฒนาชิพ จากค่ายต่างๆ ก็ได้มีการบรรจุ วงจรการสื่อสารแบบ CAN ไว้เป็นอุปกรณ์มาตรฐานในตัว MCU กันอย่างแพร่หลายเช่นเดียวกัน

ET-MINI SPI CAN CONTROL V1.0 เป็นบอร์ด CAN Controller แบบใช้การเชื่อมต่อผ่าน SPI โดยเลือกใช้ CAN Controller เบอร์ MCP2515 ของ Microchips รองรับการสื่อสารกับระบบ CAN ภายใต้มาตรฐาน CAN2.0B กล่าวคือ รองรับการสื่อสาร CAN ทั้งแบบ Standard Frame, Extend Frame และ Remote Frame โดยบอร์ด ET-MINI SPI CAN CONTROL V1.0 เหมาะสำหรับนำไปปรับปรุงระบบเดิมเพื่อเพิ่มเติมประยุกต์ใช้ในการสื่อสารผ่านระบบ CAN กับงานเดิมที่ใช้กับไมโครคอนโทรลเลอร์เบอร์ที่ผู้ใช้รู้จักและคุ้นเคยมาก่อนแล้ว แต่ภายในไมโครคอนโทรลเลอร์นั้นยังไม่มีโมดูล CAN บรรจุไว้ภายในตัวด้วย

แต่สำหรับกรณีของผู้ที่คิดจะเริ่มต้นใหม่กับไมโครคอนโทรลเลอร์ และมีความสนใจอยากจะศึกษาเรียนรู้ระบบการสื่อสารของ CAN ด้วย อาจไปเลือกใช้ ไมโครคอนโทรลเลอร์เบอร์ที่มีการบรรจุวงจรสื่อสารของ CAN รวมไว้ภายในตัวไมโครคอนโทรลเลอร์โดยตรงจะสะดวกและเหมาะสมมากกว่า

คุณสมบัติของ CAN Controller เบอร์ MCP2515 ของ Microchips

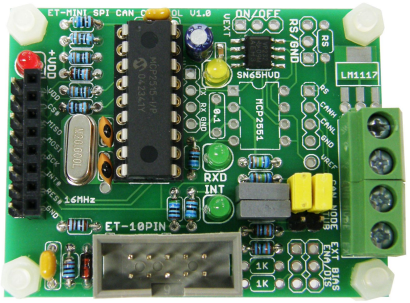
- รองรับมาตรฐาน Protocol การสื่อสาร CAN2.0B สามารถกำหนดขนาดข้อมูลในการสื่อสารได้ตั้งแต่ 0-8 Byte ทั้งแบบ Standard Frame และ Extended Frame และ Remote Frame
- มี 29 Bit Filter จำนวน 6 ชุด และ 29 Bit พร้อม Mask จำนวน 2 ชุด สำหรับภาครับข้อมูล
- มีระบบ Data Byte Filter เพื่อคัดกรองข้อมูลที่ไม่ต้องการรับทิ้งไป
- มี Transmitter Buffer สำหรับบรรจุเฟรมข้อมูลสำหรับส่งข้อมูลจำนวน 3 ชุด
- เชื่อมต่อกับ Micro Controller ผ่าน SPI ด้วยความเร็วสูงสุด 10 MHz
- มีสัญญาณ Start-of-Frame(SOF) สำหรับแสดงสถานะของการตรวจจับข้อมูลที่รับเข้ามาได้
- มีสัญญาณ Interrupt ซึ่งสามารถกำหนดและควบคุมการทำงานจากโปรแกรมได้
- มีสัญญาณแสดงสถานะของ Buffer Full
- มีสัญญาณ Request-to-Send(RTS) สำหรับควบคุมการส่งข้อมูล
- ทำงานที่แรงดัน 2.5V-5.5V

คุณสมบัติของ CAN Transceiver เบอร์ SN65HVD232D ของ Texas Instruments

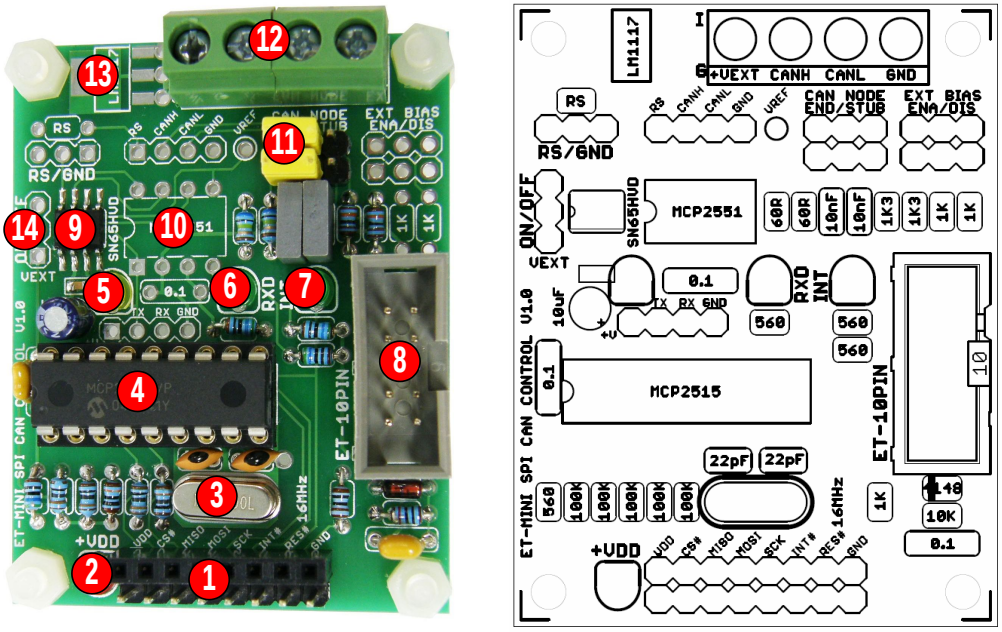
- รองรับการเชื่อมต่อกับ CAN Controller Logic ทั้งระบบ 5V และ 3.3V
- รองรับข้อกำหนดตามมาตรฐานสัญญาณ CAN ISO-11898 (standard physical layer)
- มีวงจร Termination แบบ Split Termination ภายในบอร์ด สามารถเลือกกำหนดจาก Jumper ได้ทั้งแบบ End Node(120 Ω) และ Stub Node(2.6KΩ)
- รองรับความเร็ว Bus ระหว่าง 62.5Kb/s(ระยะทาง 1000 เมตร) – 1Mb/s(ระยะทาง 30 เมตร)

คุณสมบัติของบอร์ด ET-MINI SPI CAN CONTROL V1.0

1. ใช้ CAN Controller เบอร์ MCP2515 ของ Microchips
2. ใช้ CAN Transceiver เบอร์ SN65HVD232D ของ Texas Instruments
3. รองรับการเชื่อมต่อกับ Micro Controller ผ่านทาง SPI Logic ทั้งระบบ 5V และ 3.3V
4. รองรับมาตรฐาน Protocol การสื่อสาร CAN2.0B สามารถกำหนดขนาดข้อมูลในการสื่อสารได้ตั้งแต่ 0-8 Byte ทั้งแบบ Standard Frame และ Extended Frame และ Remote Frame
5. รองรับข้อกำหนดตามมาตรฐานสัญญาณ CAN ISO-11898 (standard physical layer)
6. เชื่อมต่อสั่งงานกับ Micro Controller ผ่าน SPI ด้วยความเร็วสูงสุด 10 MHz
7. มี LED แสดง สถานะแหล่งจ่าย (+VDD) ของบอร์ด
8. มี LED แสดง สถานะของการรับข้อมูล (RX) ของ CAN Controller
9. มี LED แสดง สถานะของการส่งข้อมูล (TX) ของ CAN Controller
10. มี LED แสดง สถานะของการเกิด Interrupt จาก INT ของ CAN Controller
11. มีวงจร Termination แบบ Split Termination ภายในบอร์ด สามารถเลือกกำหนดจาก Jumper ได้ทั้งแบบ End Node(120 Ω) และ Stub Node(2.6KΩ)
12. รองรับความเร็ว Bus ระหว่าง 62.5Kb/s(ระยะทาง 1000 เมตร) – 1Mb/s(ระยะทาง 30 เมตร) โดยใช้สายสัญญาณ STP หรือ UTP
13. สัญญาณเชื่อมต่อด้าน Logic ใช้ขั้วต่อ Pin Header ขนาด 1 x 8 Male และ 1x8 Female ระยะ Pitch 2.54mm หรือ IDE 10 Pin Header Block
14. สัญญาณเชื่อมต่อด้าน CAN BUS ใช้ Terminal 4Pin (+VEXT,CANH,CANL,GND)
15. มีวงจร Regulate ขนาด 800mA เป็น Option สำหรับในกรณีที่ต้องการใช้แหล่งจ่ายไฟจาก CAN Bus (+VEXT,GND) เป็นแหล่งจ่ายไฟเลี้ยงให้กับวงจรภายในบอร์ด ซึ่งสามารถเพิ่มเติม IC Regulate เบอร์ LM1117-3.3(SOT-223) หรือ LM1117-5.0(SOT-223)
16. ขนาด PCB Size 4.4 mm x 5.6 mm



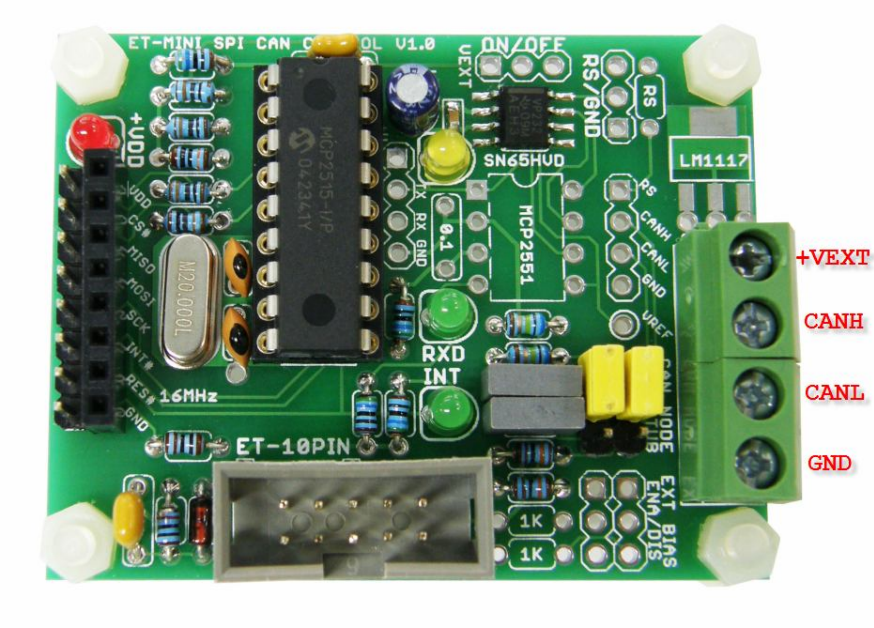
โครงสร้างบอร์ด ET-MINI SPI CAN CONTROL V1.0



รูปแสดง ตำแหน่งของอุปกรณ์ต่างๆในบอร์ด ET-MINI SPI CAN CONTROL V1.0

- **หมายเลข 1,8 คือ** ขั้วต่อ สัญญาณ Logic สำหรับเชื่อมต่อกับ Microcontroller สามารถเลือกใช้ได้ทั้งแบบ Header 1x8 Male และ Female หรือ IDE Header 10 Pin Block ซึ่งสัญญาณจากขั้วต่อทั้ง 3 ชุดนี้เป็นสัญญาณที่ขนานกันอยู่ ผู้ใช้สามารถเลือกใช้ได้ตามความสะดวก
- **หมายเลข 2 คือ** LED แสดงสถานะของ แหล่งจ่าย Power(+VDD)
- **หมายเลข 3 คือ** Crystal สำหรับ MCP2515 สามารถใช้ 16MHz หรือ 20MHz ได้ สำหรับบอร์ดมาตรฐานจะติดตั้งค่า 20MHz
- **หมายเลข 4 คือ** CAN Controller เบอร์ MCP2515 ของ Microchips
- **หมายเลข 5 คือ** LED แสดง สถานของการส่งข้อมูลของ CAN Controller
- **หมายเลข 6 คือ** LED แสดง สถานะของการรับข้อมูลของ CAN Controller
- **หมายเลข 7 คือ** LED แสดง สถานะ Interrupt จาก INT# ของ CAN Controller
- **หมายเลข 9,10 คือ** CAN Transceiver เบอร์ SN65HVD232D ของ Texas Instruments และ CAN Transceiver เบอร์ MCP2551 ของ Microchips ซึ่งต้องเลือกติดตั้งใช้งานเพียงเบอร์ใดเบอร์หนึ่ง สำหรับบอร์ดมาตรฐานจะติดตั้งเบอร์ SN65HVD232D
- **หมายเลข 11** คือ Jumper สำหรับเลือกกำหนดรูปแบบของวงจร Termination ให้กับวงจร

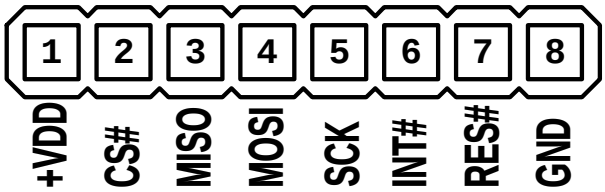
- **หมายเลข 12 คือ** ขั้วต่อ CAN Bus
  - **+VEXT** เป็นไฟเลี้ยงจากภายนอกสำหรับต่อให้บอร์ด สามารถใช้ได้กับแรงดัน 7-12V
  - **CANH** เป็นขาเชื่อมต่อสัญญาณ CAN Bus
  - **CANL** เป็นขาเชื่อมต่อสัญญาณ CAN Bus
  - **GND** เป็นจุดอ้างอิงระดับสัญญาณ Logic และแหล่งจ่ายไฟ ของบอร์ด
- **หมายเลข 13,14 คือ** IC Regulate เบอร์ LM1117 และ Jumper +VEXT(ON/OFF) ส่วนนี้เป็น Option ใช้ในกรณีที่ต้องการใช้ +VEXT จาก CAN Bus มาเป็นแหล่งจ่ายให้กับบอร์ด



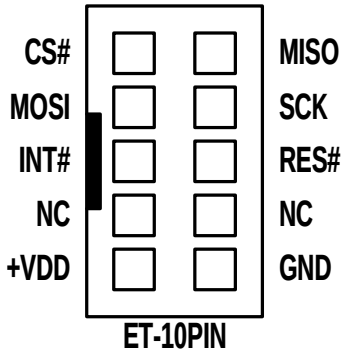
การเลือกแหล่งจ่ายไฟของบอร์ด

สำหรับบอร์ด ET-MINI SPI CAN CONTROL V1.0 ได้รับการออกแบบให้มีความอ่อนตัวต่อการใช้งาน สามารถเลือกต่อแหล่งจ่ายไฟเลี้ยงวงจรให้กับบอร์ดได้ หลายช่องทาง โดยบอร์ดสามารถเชื่อมต่อกับไมโครคอนโทรลเลอร์ได้ทั้งระบบ +3.3V และ +5V

- เชื่อมต่อไฟเลี้ยงมาจากบอร์ดไมโครคอนโทรลเลอร์ผ่านทางขั้วต่อHeader 1x8 Male/1x8 Female



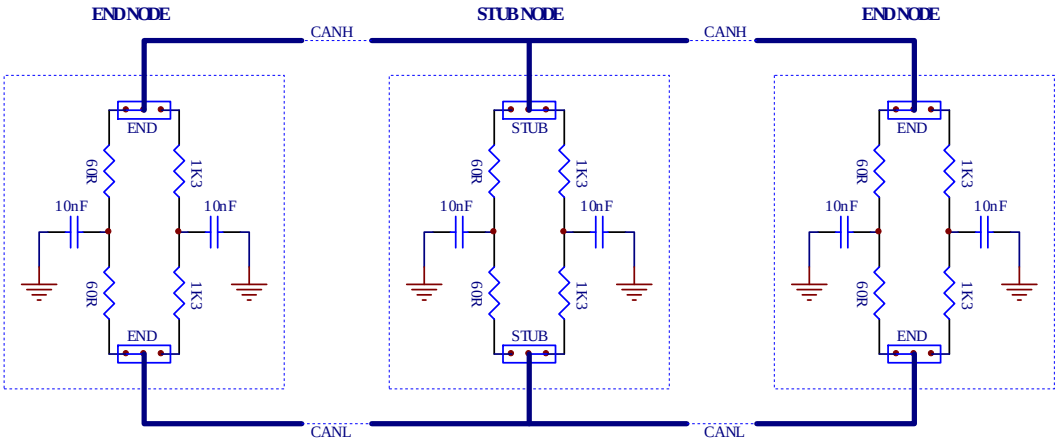
- เชื่อมต่อไฟเลี้ยงมาจากบอร์ดไมโครคอนโทรลเลอร์ผ่านทางขั้วต่อIDE Header 10Pin(ET-10PIN)



- เชื่อมต่อไฟเลี้ยงมาจาก CAN BUS โดยในกรณีจะใช้ได้กับระบบ CAN BUS ที่มีส่วน OPTION สำหรับจ่ายไฟเลี้ยงให้กับอุปกรณ์ภายนอกด้วย ซึ่งในกรณีนี้ผู้ใช้ต้องทำการติดตั้ง IC Regulate เบอร์ LM1117-3.3(SOT-223) ในกรณีต้องการใช้ไฟเลี้ยง +3.3V หรือ ใช้ LM1117-5.0(SOT-223) ในกรณีต้องการใช้ไฟเลี้ยงวงจรขนาด +5V พร้อมกับทำการเลือก Jumper VEXT(ON/OFF)ไว้ทางด้านตำแหน่ง ON ด้วย ซึ่งในกรณีนี้ระบบทั้งหมดรวมทั้งบอร์ดไมโครคอนโทรลเลอร์ต้องใช้กระแสไม่เกิน 800mA ด้วย เพราะ LM1117 สามารถจ่ายกระแสได้สูงสุด 800mA เท่านั้น

การเลือก Node สำหรับ CAN BUS

ในระบบ CAN BUS นั้นจำเป็นต้องมีการกำหนดการทำงานของวงจร Termination เพื่อชดเชยความผิดเพี้ยนของสัญญาณซึ่งเกิดจากความต้านทานในสายสัญญาณให้กับวงจร CAN Transceiver โดยถ้าเป็น CAN Transceiver ที่อยู่ตำแหน่ง ต้นสาย และ ปลายสาย ต้องกำหนดเป็น END Node แต่ถ้าเป็นวงจรของ CAN Transceiver ชุดที่อยู่ระหว่าง ต้นทาง กับ ปลายทาง จะกำหนดเป็น STUB Node ซึ่งในบอร์ด ET-MINI SPI CAN CONTROL V1.0 เองจะใช้ Jumper (END/STUB) เป็นตัวกำหนด Node ดังตัวอย่าง



รูปแสดง การเลือก Termination และผังการเชื่อมต่อ CAN BUS

ระบบ CAN BUS รองรับการเชื่อมต่อ CAN BUS ได้จำนวนสูงสุด 120 Node และมีระยะทางไกลสุดประมาณ 30เมตร ถึง 1000 เมตร ขึ้นอยู่กับความเร็วที่เลือกใช้ โดยใช้สายสัญญาณแบบ STP หรือ UTP โดยระยะทางของการสื่อสารจะแปรผกผันกับความเร็วในการรับส่งข้อมูล ดังข้อมูลใน ตาราง

Bit Rate (kb/s)	Bus Length (เมตร)
1000	30
500	100
250	250
125	500
62.5	1000

ตาราง เปรียบเทียบความเร็วและระยะทางของการสื่อสาร CAN BUS

การเชื่อมต่อกับ Micro Controller (MCU)

MCP2515 เป็นชิพ CAN Controller ซึ่งมีระบบ การทำงานเป็นไปตาม มาตรฐาน ของ การสื่อสาร ตามข้อกำหนด (Protocol) ของ CAN2.0B ทุกประการ กล่าวคือ MCP2515 จะมีขีดความสามารถในการ รับส่งข้อมูล CAN Bus ทั้งแบบ Standard Frame และ Extended Frame และ Remote Frame นอกจากนี้ แล้วยังมีระบบคัดกรองข้อมูล เพื่อทำการแยกข้อมูลที่ไม่ต้องการรับออกไปจากภาครับ โดยสามารถแยก กำหนดรูปแบบการคัดกรองการรับข้อมูลได้ 2 ชุด พร้อมกับมี Buffer สำหรับจัดเก็บข้อมูลที่ผ่านการคัด กรองตามเงื่อนไขไว้แล้วจำนวน 6 ชุด ซึ่งความสามารถในการคัดกรองข้อมูลนี้จะช่วยลดภาระในการรับและ ตรวจสอบข้อมูลของ ส่วนประมวลผลลงไปได้เป็นอย่างมาก คือแทนที่จะต้องให้หน่วยประมวลผลทำการรับ ข้อมูลทั้งหมดที่วิ่งอยู่บนบัสแล้วมาคัดกรองเอง เพื่อแยกเอาเฉพาะข้อมูลที่ต้องการมาใช้ ข้อมูลส่วนที่ไม่ ต้องการก็โยนทิ้งไป ซึ่งถ้าหากว่าในระบบบัส มีอุปกรณ์เชื่อมต่ออยู่จำนวนมาก เมื่อมีการสื่อสารกันในบัส หน่วยประมวลผลก็ต้องทำการรับข้อมูลทั้งหมดเพื่อนำมาคัดกรองวิเคราะห์แยกแยะข้อมูลอยู่ตลอดเวลา ซึ่ง จะเป็นภาระอย่างมากสำหรับหน่วยประมวลผล

แต่เมื่อมีระบบคัดกรองข้อมูล เราก็สามารถจะกำหนดเงื่อนไขในการคัดกรองข้อมูลต่างๆที่วิ่งอยู่บน บัส เพื่อให้วงจรด้านรับทำการตรวจจับและคัดกรองเอาเฉพาะข้อมูลที่ต้องการจะใช้เข้ามารอไว้ใน Buffer ซึ่งชุดประมวลผลก็เพียงแต่คอยตรวจสอบว่ามีข้อมูลถูกรับเข้ามาใน Buffer หรือยัง ถ้ามีก็ไปอ่านเอาข้อมูล นั้นมาใช้งาน ส่วนข้อมูลที่ไม่ต้องการไม่เกี่ยวข้องก็จะถูกคัดกรองและโยนทิ้งออกไปจากส่วนคัดกรองข้อมูล เองโดยอัตโนมัติ ซึ่งในส่วนของการกำหนดเงื่อนไขและตั้งค่าการทำงานให้กับ MCP2515 เพื่อให้ทำการ รับส่งข้อมูลกับ CAN Bus ได้นั้น จะกระทำผ่านรีจิสเตอร์ต่างๆของ CAN Controller ซึ่งจะมีรูปแบบและ ข้อกำหนดเป็นไปตามมาตรฐานของ CAN2.0B ซึ่งระบบบัสของ MCP2515 สามารถจะใช้ติดต่อสื่อสารกับ อุปกรณ์ใดๆก็ได้ที่มีคุณสมบัติ การทำงานตรงตามมาตรฐาน CAN2.0B ไม่ว่าจะเป็น CAN Controller ด้วย กันเอง หรือเป็นไมโครคอนโทรลเลอร์ที่ได้บรรจุวงจรสื่อสารแบบ CAN ไว้ภายในตัวเอง ของค่ายต่างๆ

สิ่งที่ MCP2515 มีความแตกต่างจากไมโครคอนโทรลเลอร์ที่บรรจุวงจร CAN ไว้ภายใน คือ วิธีการ ติดต่อและเข้าถึงรีจิสเตอร์ของ CAN Controller ซึ่งถ้าเป็นไมโครคอนโทรลเลอร์ที่มีการบรรจุวงจร CAN ไว้ ภายในแล้ว จะมีการเชื่อมบัส ของไมโครคอนโทรลเลอร์เข้ากับบัสของ CAN Controller ภายในเรียบร้อย แล้ว การเข้าถึงรีจิสเตอร์ต่างๆก็จะสามารถอ้างถึงรีจิสเตอร์เหล่านั้นผ่านทางคำสั่งของไมโครคอนโทรลเลอร์ ได้โดยตรง แต่สำหรับกรณีของ MCP2515 นั้น การจะเข้าถึงรีจิสเตอร์ต่างๆได้ จะต้องผ่านทางช่องทางการ สื่อสารภายนอกตัว MCU ซึ่งจะใช้การสื่อสารอนุกรมแบบ SPI ดังนั้นจะต้องมีการกำหนดการทำงานให้กับ SPI ของไมโครคอนโทรลเลอร์ พร้อมกับเขียนโปรแกรมคำสั่งเพื่อสั่งไมโครคอนโทรลเลอร์ส่งคำสั่งและข้อมูล ออกมาบน SPI Bus ให้มีรูปแบบตรงกับข้อกำหนดที่ MCP2515 กำหนดไว้ เพื่อให้ MCP2515 สามารถรับรู้ ความต้องการของไมโครคอนโทรลเลอร์ ว่าต้องการจะอ่านเขียนข้อมูลกับรีจิสเตอร์หรือหน่วยความจำ

ตำแหน่งใดของ CAN Controller ซึ่งในการเชื่อมต่อบอร์ด ET-MINI SPI CAN CONTROL V1.0 เข้ากับบอร์ดไมโครคอนโทรลเลอร์นั้น จะใช้สัญญาณ TTL Logic จำนวน 4-6 เส้น โดยใช้สัญญาณ Logic แบบ SPI จำนวน 4 เส้น และเป็นสัญญาณ Interrupt (INT#) กับ Reset (RES#) อีก 2 เส้น ซึ่ง INT# และ RES# จะใช้หรือไม่ใช้ก็ได้ โดยสัญญาณในการเชื่อมต่อของบอร์ดมีดังนี้

- **+VDD** เป็นแหล่งจ่ายไฟเลี้ยงวงจรของบอร์ด ควรใช้กับแหล่งจ่ายไฟเลี้ยงที่มีขนาดแรงดันชุดเดียวกับไมโครคอนโทรลเลอร์ (MCU) ซึ่งสามารถใช้ได้กับระบบ +3.3V หรือ +5V ได้
- **CS#** เป็นสัญญาณ เลือกการทำงานของ MCP2515 ทำงานที่ Logic “0” โดยต้องต่อสัญญาณ Output Logic จาก MCU เพื่อส่งไปควบคุมสั่งงาน MCP2515 ด้วย
- **MISO** เป็นสัญญาณ รับข้อมูล SPI ส่งออกจาก MCP2515 ออกมาให้กับ MCU
- **MOSI** เป็นสัญญาณส่งข้อมูล SPI โดยส่งออกจาก MCU ไปยัง MCP2515
- **SCK** เป็นสัญญาณนาฬิกาของ SPI ส่งออกจาก MCU ไปยัง MCP2515
- **INT#** เป็นขา Interrupt ส่งออกจาก MCP2515 มายัง MCU ทำงานที่ Logic “0”
- **RES#** เป็นขา Reset ของ MCP2515 ทำงานที่โลจิก “0” โดยขาสัญญาณนี้จะใช้หรือไม่ก็ได้ เนื่องจากในบอร์ดมีวงจร Auto Reset ให้กับ MCP2515 ไว้อยู่แล้ว แต่ถ้าต้องการให้สามารถสั่ง Reset MCP2515 จาก MCU ได้ ก็ให้ต่อสัญญาณ Output Logic จาก MCU เพื่อส่งสัญญาณไปควบคุมการ Reset ของ MCP2515 ได้ตามต้องการ
- **GND** เป็นจุดอ้างอิงสัญญาณและระดับแหล่งจ่ายให้กับวงจร

ในการสื่อสารระหว่าง MCU กับบอร์ด ET-MINI SPI CAN CONTROL V1.0 นั้น จะใช้การติดต่อสื่อสารแบบ SPI ซึ่งจะใช้ MCU ทำหน้าที่เป็น Master และให้ MCP2515 ทำหน้าที่เป็น Slave โดยที่ MCU จะส่งข้อมูลให้ MCP2515 ผ่านทางขา MOSI ไปยังขา SI ของ MCP2515 ในจังหวะที่สัญญาณนาฬิกา SCK เป็นขอบขาขึ้น (Rising Edge) ซึ่งถ้าต้องมีการตอบรับจาก MCP2515 กลับมายัง MCU ด้วย MCP2515 ก็จะส่งข้อมูลย้อนกลับมาทางขา SO มายังขา MISO ของ SPI(MCU) ในจังหวะที่สัญญาณนาฬิกา SCK เป็นขอบขาลง(Falling Edge)

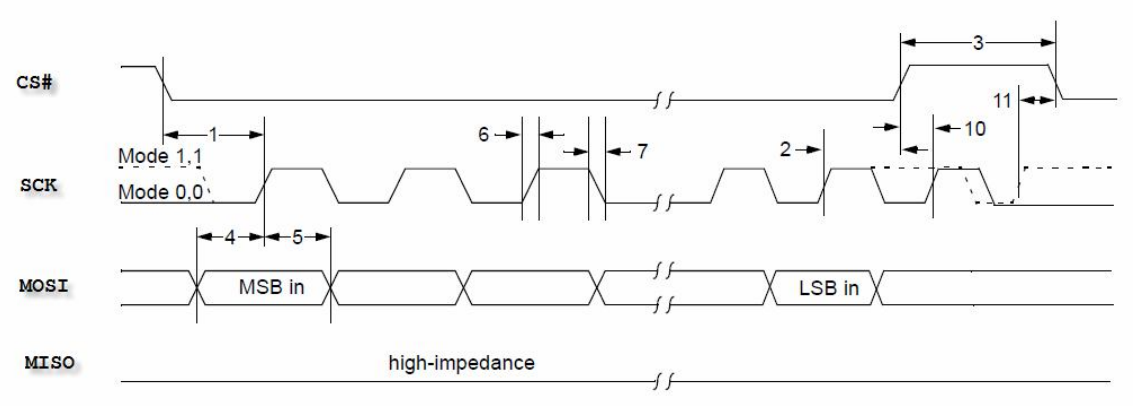
โดยในการสื่อสาร SPI จะกำหนดจังหวะเริ่มต้นและสิ้นสุดกันด้วยสัญญาณ CS# โดยสัญญาณนาฬิกา SCK จะถูกสร้างจาก SPI ด้าน MCU เสมอ โดยการลำเลียงข้อมูล SPI นั้นจะควบคุมจากจังหวะของสัญญาณนาฬิกา SCK โดยเริ่มต้นจากบิตข้อมูลที่มีนัยสำคัญสูงสุด (MSB) ก่อนเป็นลำดับแรก ซึ่งข้อมูลไบท์แรกจะเริ่มต้นส่งโดยเริ่มนับจากจังหวะที่สัญญาณ CS# เริ่มต้นเป็น Logic “0” โดยไบท์แรกจะเป็นรหัส คำสั่ง (Instruction) เสมอ โดยที่ MCU จะส่งข้อมูลออกไปให้กับ MCP2515 ผ่านทางขา MOSI โดยสร้างสัญญาณนาฬิกา SCK สำหรับกำหนดจังหวะการส่งข้อมูลแต่ละบิต ในขณะเดียวกัน ถ้าหากว่าทาง MCP2515 ต้องการส่งข้อมูลย้อนกลับมาให้กับ MCU มันก็จะส่งสัญญาณออกมาทางขา SO ผ่านไป

ยังขา MISO ของ SPI(MCU) ในจังหวะของสัญญาณนาฬิกา SCK ลูกเดียวกัน ต่างกันที่ MCU ส่งข้อมูลในจังหวะที่ SCK เป็นขอบขาขึ้น (Rising Edge) แต่ MCP2515 จะส่งข้อมูลย้อนกลับมาในจังหวะที่สัญญาณนาฬิกาของ SCK เป็นขอบขาลง (Falling Edge) เท่านั้นเอง ดังนั้นเมื่อ MCU ต้องการอ่านค่าข้อมูลจาก MCP2515 ก็จะต้องมีการส่งข้อมูลใดๆ ซึ่งปรกติควรจะเป็น 0xFF (1111 1111) ออกไปยัง MCP2515 ด้วยเสมอซึ่งข้อมูลที่ส่งออกไปทางขา MOSI ในจังหวะที่ต้องการอ่านข้อมูลจาก MCP2515 นี้จะไม่มีผลใดๆต่อการทำงานของ MCP2515 เพราะ MCP2515 จะไม่สนใจข้อมูลเหล่านี้ เพียงแต่MCP2515 จะอาศัยจังหวะของสัญญาณนาฬิกา SCK เพื่อทำการส่งข้อมูลย้อนกลับมาให้MCU ทางขา MISO เท่านั้นเอง

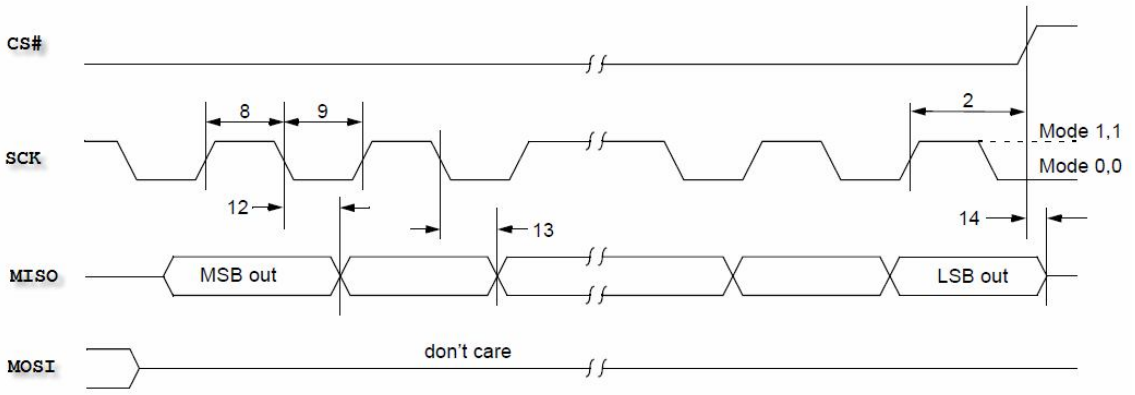
สำหรับในกรณีที่ต้องการติดต่อสั่งอ่านหรือเขียนข้อมูลกับ MCP2515 ซ้ำๆกันในคำสั่ง หลังจากที่มีการส่งรหัสคำสั่งในไบท์แรกและส่งค่าตำแหน่งแอดเดรสในไบท์ที่2 แล้ว ไบท์ที่3 จะเป็นค่าของข้อมูลที่ต้องการจะอ่านหรือเขียนกับ MCP2515 ในตำแหน่งแอดเดรสที่กำหนดไว้ในไบท์ที่2 แต่ถ้ามีการอ่านหรือเขียนข้อมูลไบท์ที่4 ตามไปอีกจะทำให้ค่าตำแหน่งแอดเดรสของข้อมูลหรือรีจิสเตอร์นั้นๆ จะถูกเพิ่มค่าขึ้นครั้งละ 1 ตำแหน่งโดยอัตโนมัติ เมื่อมีการสั่งอ่านหรือเขียนข้อมูลไบท์ถัดไปตามมาอีกค่าตำแหน่งแอดเดรสนี้ก็จะถูกเพิ่มค่าขึ้นไปอีกครั้งละ 1 ตำแหน่งเสมอโดยอัตโนมัติ ซึ่งผู้ใช้สามารถสั่งอ่านหรือเขียนข้อมูลแบบเรียงลำดับอย่างต่อเนื่องจากตำแหน่งแอดเดรสปัจจุบันกับ MCP2515 ได้ทันที โดยไม่จำเป็นต้องส่งรหัสคำสั่ง และค่าตำแหน่งแอดเดรสเริ่มต้นใหม่ให้เสียเวลา ซึ่งการสื่อสารในแต่ละครั้งจะเริ่มต้นเมื่อสัญญาณ CS# เปลี่ยนจาก “1” เป็น “0” และจะสิ้นสุดลงเมื่อ CS# ถูกเปลี่ยนกลับจาก “0” ไปเป็น “1” เสมอ โดยรูปแบบของชุดคำสั่งต่างๆมีดังนี้

คำสั่ง	รูปแบบ	รายละเอียดคำสั่ง
RESET	**1100 0000**	สั่งรีเซ็ตการทำงานของ MCP2515
READ	**0000 0011**	อ่านข้อมูลจากรีจิสเตอร์ จากตำแหน่งที่เลือกไว้
READ RX BUFFER	**1001 0nm0**	อ่านข้อมูลจาก RX Buffer ชุดที่ระบุโดย nm
WRITE	**0000 0010**	เขียนข้อมูลให้รีจิสเตอร์ ไปยังตำแหน่งที่เลือกไว้
LOAD TX BUFFER	**0100 0abc**	โหลดข้อมูลให้ TX Buffer ชุดที่ระบุตำแหน่งโดย abc
RTS(REQUEST TO SEND)	**1000 0nnn**	สั่งให้ MCP2515 เริ่มส่งข้อมูลใน TX Buffer ชุดที่ nnn
READ STATUS	**1010 0000**	อ่านค่าสถานะจาก MCP2515
RX STATUS	**1011 0000**	อ่านค่าสถานะของชุดข้อมูลที่ตรงกับ รูปแบบที่กำหนดไว้
BIT MODIFY	**0000 0101**	สั่งเปลี่ยนแปลงแก้ไขค่ารีจิสเตอร์เฉพาะบิตที่กำหนด

ตารางแสดง SPI Command ของ MCP2515



รูปแสดง รูปแบบสัญญาณในการเขียนข้อมูลให้ MCP2515



รูปแสดง รูปแบบสัญญาณในการอ่านข้อมูลจาก MCP2515

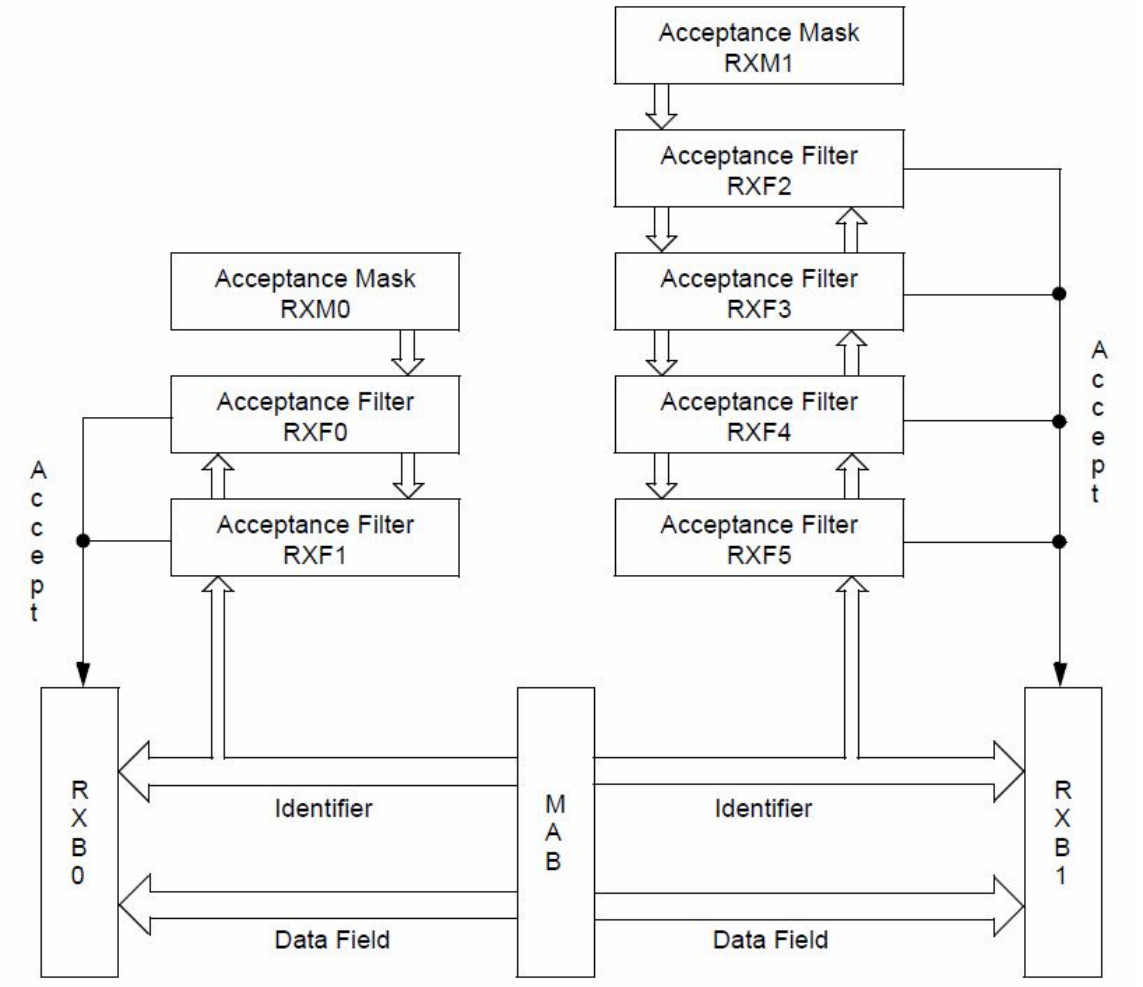
TX Buffer

สำหรับ TX Buffer ของ MCP2515 จะมีจำนวน 3 ชุด คือ TX Buffer0,TX Buffer1 และ TX Buffer2 โดย TX Buffer แต่ละชุด มีขนาด 14 ไบท์ โดยมีตำแหน่งแอดเดรสเริ่มต้นของ TX Buffer ที่ 0x30,0x40 และ 0x50 ตามลำดับ โดย TX Buffer ทั้ง 3 ชุดมี ขนาด และ โครงสร้างการจัดเก็บข้อมูลเหมือนกัน คือ

- ไบท์แรก ของ TX Buffer ใช้บรรจุข้อมูลสำหรับใช้กำหนดเงื่อนไขการส่งข้อมูล ซึ่งข้อมูลในตำแหน่งนี้ก็คือข้อมูลที่กำหนดให้กับรีจิสเตอร์TXB0CTRL, TXB1CTRL และ TXB2CTRL นั่นเอง โดยจะมีตำแหน่งแอดเดรส 0x30,0x40 และ 0x50 ตามลำดับ
- ไบท์ที่2 ถึง ไบท์ที่6 ใช้บรรจุค่า ID ซึ่งมีขนาด 5 ไบท์ โดยมีตำแหน่งแอดเดรสเริ่มต้นอยู่ที่ตำแหน่ง 0x31,0x41 และ 0x51 ตามลำดับ
- ไบท์ที่7 ถึง ไบท์ที่14 ใช้บรรจุข้อมล ซึ่งมีขนาดสูงสุด8 ไบท์ โดยมีตำแหน่งแอดเดรสเริ่มต้นอยู่ที่ตำแหน่ง 0x36,0x46 และ 0x56 ตามลำดับ

RX Buffer

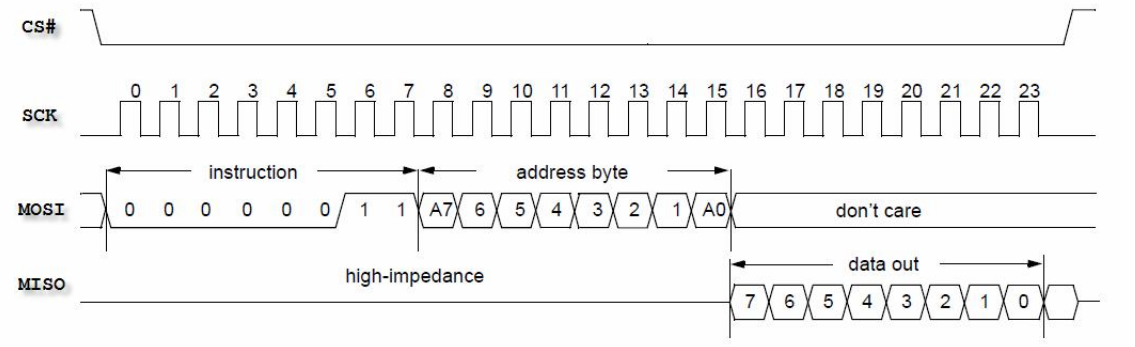
สำหรับ RX Buffer ของ MCP2515 จะมีจำนวน 2ชุด คือ RX Buffer0 และ RX Buffer1 โดย MCP2515 จะมีกลไกลในการตรวจจับและจัดเก็บข้อมูลที่รับมาจากบัสไว้ใน RX Buffer0 และ RX Buffer1 ซึ่งเรียกว่า MAB(Message Assembly Buffer) ซึ่ง MAB จะมีกลไกลในการคัดกรองข้อมูลที่รับเข้ามาตามเงื่อนไขที่ผู้ใช้กำหนดไว้แล้ว แล้วนำไปจัดเก็บรอไว้ใน Buffer RXF0 ถึง RXF5 อีกชั้นหนึ่ง เพื่อทำให้การรับข้อมูลทำได้อย่างต่อเนื่องและรวดเร็ว โดย RX Buffer0 จะมีชุด Buffer สำหรับจัดเก็บข้อมูลที่ผ่านการคัดกรองแล้วจำนวน 2 ชุด คือ RXF0 และ RXF1 ส่วน RX Buffer1 จะมีชุด Buffer สำหรับใช้จัดเก็บข้อมูลที่ผ่านการคัดกรองแล้วอีกจำนวน 4 ชุด คือ RXF2,RXF3,RXF4 และ RXF5 ตามลำดับ ดังรูป



รูปแสดง โครงสร้าง RX Buffer ของ MCP2515

READ INSTRUCTION

คำสั่งนี้ใช้สำหรับสั่งอ่านค่าข้อมูลจากรีจิสเตอร์ของ MCP2515 โดยรูปแบบของคำสั่งนี้จะมีด้วยกัน 3 Byte หรือมากกว่าในกรณีที่ต้องการอ่านแบบเรียงลำดับต่อเนื่อง โดยข้อมูลไบท์แรกเป็นรหัสคำสั่งมีค่าเป็น 0x03 (0000 0011) ส่วนไบท์ที่ 2 เป็นตำแหน่งแอดเดรสของรีจิสเตอร์ที่ต้องการอ่านมีขนาด 8 บิต ส่วนไบท์ที่ 3 เป็นค่าข้อมูลที่ส่งกลับมาจาก MCP2515 ซึ่งถ้าต้องการอ่านข้อมูลลำดับที่อยู่ต่อเนื่องจากตำแหน่งปัจจุบันอีกก็สามารถสั่งอ่านค่าข้อมูลไบท์ต่อไปได้อีกเรื่อยๆ โดยค่าตำแหน่งแอดเดรสจะถูกเพิ่มค่าให้เองครั้งละ 1 ตำแหน่งโดยอัตโนมัติ โดยคำสั่งนี้มีรูปแบบสัญญาณของการรับส่งเป็นดังรูป



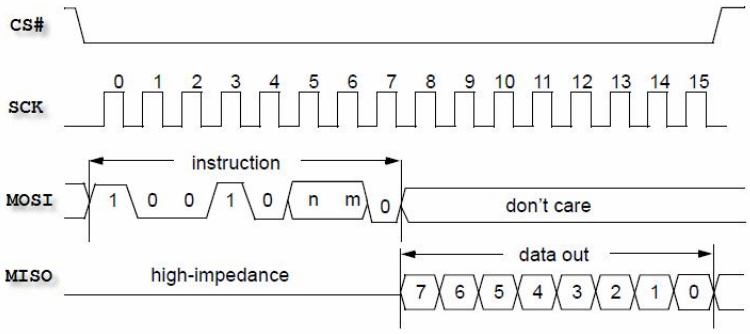
แสดง รูปสัญญาณ SPI ของคำสั่ง READ INSTRUCTION

READ RX BUFFER INSTRUCTION

คำสั่งนี้ใช้สำหรับสั่งอ่านค่า RX Buffer ชุดที่ระบุในคำสั่ง โดยมีขนาด 2 ไบท์ หรือมากกว่าในกรณีที่ต้องการอ่านแบบเรียงลำดับต่อเนื่องกันไป ซึ่งไบท์แรก จะเป็นรหัสคำสั่งมีค่า 0x90|0,n,m,0 (1001 0nm0) โดย nm ในชุดคำสั่งจะเป็นค่าสำหรับระบุชุดของ RX Buffer ที่ต้องการเริ่มต้นทำการอ่านข้อมูล ซึ่งสามารถเลือกอ่าน RX Buffer ได้ 2 ชุด คือ RX Buffer0 และ RX Buffer1 แต่ใน RX Buffer แต่ละชุด สามารถเลือกกำหนดตำแหน่งเริ่มต้นสำหรับทำการอ่านข้อมูลได้อีกว่าจะให้เริ่มต้นอ่านข้อมูลไบท์แรกที่ตำแหน่ง Buffer ที่เก็บค่า ID หรือ จะให้เริ่มต้นอ่านข้อมูลใน Buffer ที่ตำแหน่งเริ่มต้นที่เก็บข้อมูลไบท์แรก(Data) ดังนี้

n	m	Address Pointer	Address
0	0	RX Buffer0 เริ่มต้นอ่านข้อมูลตำแหน่งที่ใช้เก็บค่าID High(RXB0SIDH)	0x61
0	1	RX Buffer0 เริ่มต้นอ่านข้อมูลตำแหน่งที่ใช้เก็บค่าข้อมูลไบท์แรก(RXB0D0)	0x66
1	0	RX Buffer1 เริ่มต้นอ่านข้อมูลตำแหน่งที่ใช้เก็บค่าID High(RXB1SIDH)	0x71
1	1	RX Buffer1 เริ่มต้นอ่านข้อมูลตำแหน่งที่ใช้เก็บค่าข้อมูลไบท์แรก(RXB1D0)	0x76

โดยข้อมูลไบท์ที่2 จะเป็นข้อมูลที่ส่งกลับมาจาก MCP2515 โดยมีขนาด 8 บิต ซึ่งคำสั่งนี้สามารถอ่านข้อมูลต่อเนื่องไปได้อีก โดยส่งข้อมูลใดๆต่อเนื่องออกมาให้ MCP2515 ทางขา MOSI ของ MCU ไปยังขา SI ของ MCP2515 ซึ่ง MCP2515 ก็จะส่งค่าข้อมูลใน RX Buffer ตำแหน่งถัดไปออกมาให้ทางขา SO ให้กับ MCU ทางขา MISO อย่างต่อเนื่องตามลำดับเช่นเดียวกัน เมื่อต้องการสิ้นสุดการอ่านจึง กำหนดให้สัญญาณ CS# กลับจาก “0” เป็น “1” ดังรูป



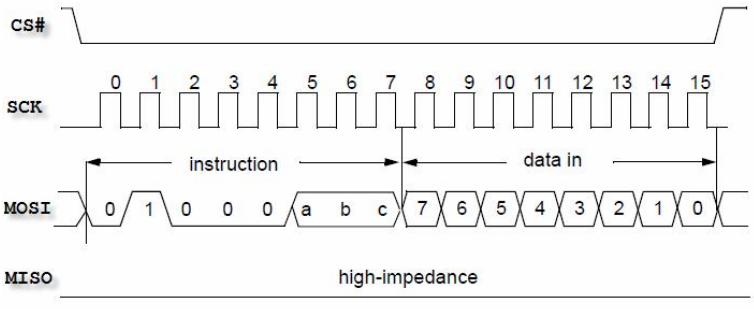
n	m	Address Points to	Address
0	0	Receive Buffer 0, Start at RXB0SIDH	0x61
0	1	Receive Buffer 0, Start at RXB0D0	0x66
1	0	Receive Buffer 1, Start at RXB1SIDH	0x71
1	1	Receive Buffer 1, Start at RXB1D0	0x76

แสดง รูปสัญญาณ SPI ของคำสั่ง READ RX BUFFER INSTRUCTION

LOAD TX BUFFER

คำสั่งนี้ใช้สำหรับสั่งเขียนข้อมูลซึ่งเป็นค่าตัวชี้ตำแหน่งแอดเดรส(Pointer) ของ TX Buffer ให้กับ MCP2515 เพื่อชี้ตำแหน่งเริ่มต้นสำหรับเตรียมเขียนข้อมูลไปยัง TX Buffer ในลำดับต่อไป โดยชุดคำสั่งนี้จะมีขนาด 2 ไบท์ โดยไบท์แรกจะเป็นรหัสคำสั่งมีค่า 0x40|abc (0100 0abc) โดย abc ในรหัสคำสั่งใช้เป็นค่ากำหนดตัวเลือกชุด TX Buffer ที่ต้องการ โดย MCP2515 จะมี TX Buffer ทั้งหมดจำนวน 3 ชุด ด้วยกัน คือ TX Buffer0,TX Buffer1 และ TX Buffer2 โดย โดยในคำสั่งนี้จะยอมให้ผู้ใช้สามารถเลือกกำหนดค่าตัวชี้ตำแหน่งเริ่มต้นใน TX Buffer แต่ละชุดได้ชุดละ 2 ตำแหน่ง คือชี้ไปทำตำแหน่งเริ่มต้นสำหรับเก็บค่า ID หรือชี้ไปที่ตำแหน่งเริ่มต้นสำหรับเก็บค่าข้อมูล(Data) ซึ่งในชุดคำสั่งสามารถเลือกกำหนดค่าได้ดังนี้

a	b	c	ตำแหน่ง Pointer	ค่าตำแหน่งแอดเดรส
0	0	0	TX Buffer0 ชี้ที่ตำแหน่งเริ่มต้นของ ID High(TXB0SIDH)	0x31
0	0	1	TX Buffer0 ชี้ที่ตำแหน่งเริ่มต้นของ Data ไบท์แรก(TXB0D0)	0x36
0	1	0	TX Buffer1 ชี้ที่ตำแหน่งเริ่มต้นของ ID High(TXB1SIDH)	0x41
0	1	1	TX Buffer1 ชี้ที่ตำแหน่งเริ่มต้นของ Data ไบท์แรก(TXB1D0)	0x46
1	0	0	TX Buffer2 ชี้ที่ตำแหน่งเริ่มต้นของ ID High(TXB2SIDH)	0x51
1	0	1	TX Buffer2 ชี้ที่ตำแหน่งเริ่มต้นของ Data ไบท์แรก(TXB2D0)	0x56



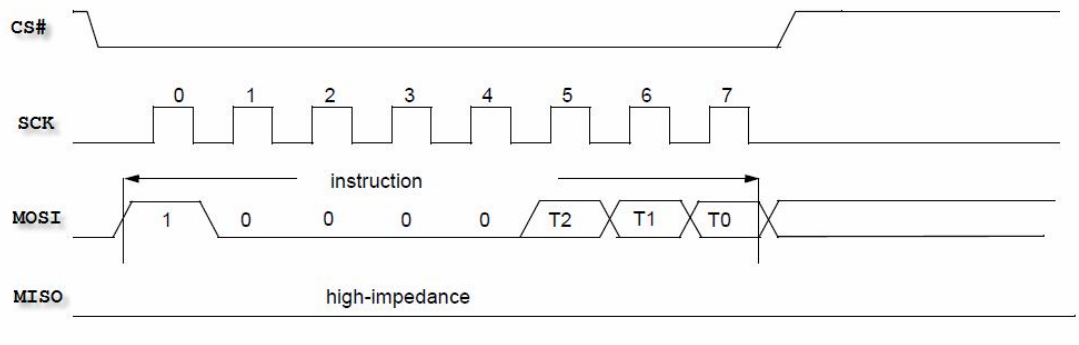
a	b	c	Address Points to	Addr
0	0	0	TX buffer 0, Start at TXB0SIDH	0x31
0	0	1	TX buffer 0, Start at TXB0D0	0x36
0	1	0	TX buffer 1, Start at TXB1SIDH	0x41
0	1	1	TX buffer 1, Start at TXB1D0	0x46
1	0	0	TX buffer 2, Start at TXB2SIDH	0x51
1	0	1	TX buffer 2, Start at TXB2D0	0x56

แสดง รูปสัญญาณ SPI ของคำสั่ง LOAD TX BUFFER

REQUEST-TO-SEND (RTS) INSTRUCTION

คำสั่งนี้ใช้สำหรับ สั่งให้ MCP2515 ทำการส่งข้อมูลใน TX Buffer ไปยังบัส โดยคำสั่งนี้จะมีขนาดเพียง 1 ไบท์ และมีรหัสคำสั่งเป็น 0x80|T0|T1|T2 (1000 0T2T1T0) ซึ่งค่า T2,T1 และ T0 จะเป็นค่าสำหรับเลือกกำหนดว่าจะให้ MCP2515 ทำการส่งข้อมูลของ TX Buffer ชุดใดออกไปในบัส โดยต้องกำหนดให้บิตของ T2 หรือ T1 หรือ T0 มีค่าเป็น “1” เพื่อให้มีการส่งข้อมูลใน TX Buffer ชุดที่ถูกเลือกออกไปในบัส ถ้าส่งรหัสคำสั่งนี้โดยกำหนดให้ค่าของ T2,T1 และ T0 เป็น “0” ทั้งหมด คำสั่งนี้จะไม่ถูกตอบสนอง

- T0 หมายถึง กำหนดให้ส่งข้อมูลในTX Buffer0
- T1 หมายถึง กำหนดให้ส่งข้อมูลในTX Buffer1
- T2 หมายถึง กำหนดให้ส่งข้อมูลในTX Buffer2



แสดง รูปสัญญาณ SPI ของคำสั่ง REQUEST TO SEND INSTRUCTION

BIT MODIFY INSTRUCTION

คำสั่งนี้ใช้สำหรับสั่งเปลี่ยนแปลงแก้ไขข้อมูลในรีจิสเตอร์ของ MCP2515 เฉพาะในตำแหน่งบิตที่ต้องการเท่านั้น ทั้งนี้ก็เนื่องจากว่าโดยปรกติแล้ว รีจิสเตอร์ของ MCP2515 มีขนาด 8 บิต แต่ในบางกรณีมีความจำเป็นต้องสั่งเปลี่ยนแปลงแก้ไขข้อมูลในรีจิสเตอร์เพียงบางบิต ส่วนบิตอื่นๆยังคงต้องการให้มีค่าคงเดิมไว้ แต่เนื่องจากโครงสร้างสถาปัตยกรรมของ MCP2515 ไม่สามารถเข้าถึงข้อมูลในระดับบิตเพื่อทำการแก้ไขเปลี่ยนแปลงค่าข้อมูลเป็นรายบิตได้เองโดยตรง ดังนั้น วิธีการตามปรกติเมื่อต้องการเปลี่ยนแปลงแก้ไขข้อมูลเพียงบางบิตนั้น โดยทั่วไปจะต้องทำการสั่งอ่านค่าข้อมูลเดิมออกมาจากรีจิสเตอร์แล้วทำการสั่งกระทำทางโลจิกเพื่อเปลี่ยนแปลงแก้ไขข้อมูลบิตที่ต้องการ เมื่อเสร็จเรียบร้อยแล้ว จึงทำการสั่งเขียนข้อมูลนั้นกลับไปยังรีจิสเตอร์ของ MCP2515 ใหม่อีกครั้งหนึ่ง ซึ่งในการสั่งอ่านและเขียนข้อมูลให้กับรีจิสเตอร์ของ MCP2515 นั้นต้องกระทำผ่าน SPI ซึ่งต้องมีการส่งลำดับคำสั่ง ตำแหน่งรีจิสเตอร์ และ ข้อมูล เรียงลำดับกันไปครั้งละ 24บิต ถ้าต้องมีการสั่งเปลี่ยนแปลงค่าข้อมูลเพียงบิตใดบิตหนึ่งซ้ำๆกันหลายๆครั้ง จะทำให้เกิดความล่าช้าเสียเวลาเป็นอย่างมาก

แต่เมื่อใช้คำสั่ง BIT MODIFY นี้จะสามารถช่วยลดเวลาในการติดต่อสื่อสารเพื่อสั่งงาน MCP2515 ได้เป็นอย่างมาก ทำให้การทำงานมีความรวดเร็วมากขึ้น แต่อย่างไรก็ตามคำสั่งนี้จะไม่สามารถเข้าถึงรีจิสเตอร์ของ MCP2515 ได้ทั้งหมด แต่จะสามารถเข้าถึงรีจิสเตอร์ได้เพียงบางรีจิสเตอร์และบางบิตเท่านั้น ซึ่งผู้ใช้สามารถตรวจสอบได้จากตาราง รีจิสเตอร์MAP ใน Data Sheet ดังรูปตาราง

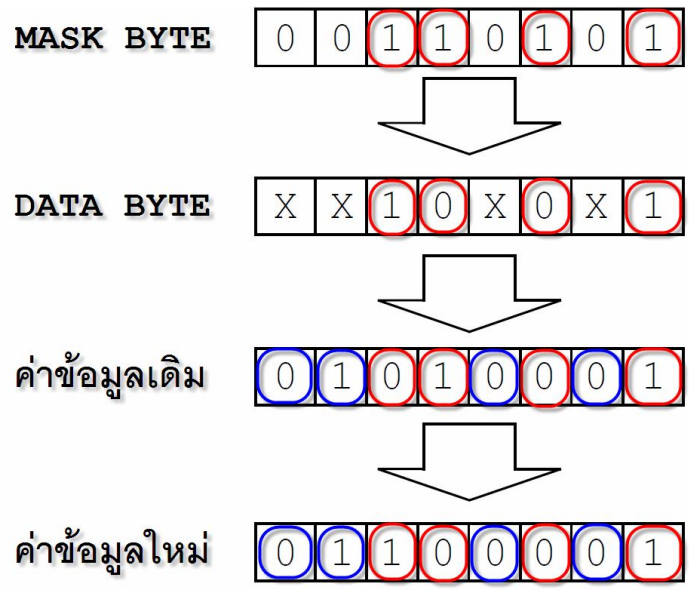
Lower Address Bits	Higher-Order Address Bits							
	0000 xxxx	0001 xxxx	0010 xxxx	0011 xxxx	0100 xxxx	0101 xxxx	0110 xxxx	0111 xxxx
0000	RXF0SIDH	RXF3SIDH	RXM0SIDH	TXB0CTRL	TXB1CTRL	TXB2CTRL	RXB0CTRL	RXB1CTRL
0001	RXF0SIDL	RXF3SIDL	RXM0SIDL	TXB0SIDH	TXB1SIDH	TXB2SIDH	RXB0SIDH	RXB1SIDH
0010	RXF0EID8	RXF3EID8	RXM0EID8	TXB0SIDL	TXB1SIDL	TXB2SIDL	RXB0SIDL	RXB1SIDL
0011	RXF0EID0	RXF3EID0	RXM0EID0	TXB0EID8	TXB1EID8	TXB2EID8	RXB0EID8	RXB1EID8
0100	RXF1SIDH	RXF4SIDH	RXM1SIDH	TXB0EID0	TXB1EID0	TXB2EID0	RXB0EID0	RXB1EID0
0101	RXF1SIDL	RXF4SIDL	RXM1SIDL	TXB0DLC	TXB1DLC	TXB2DLC	RXB0DLC	RXB1DLC
0110	RXF1EID8	RXF4EID8	RXM1EID8	TXB0D0	TXB1D0	TXB2D0	RXB0D0	RXB1D0
0111	RXF1EID0	RXF4EID0	RXM1EID0	TXB0D1	TXB1D1	TXB2D1	RXB0D1	RXB1D1
1000	RXF2SIDH	RXF5SIDH	CNF3	TXB0D2	TXB1D2	TXB2D2	RXB0D2	RXB1D2
1001	RXF2SIDL	RXF5SIDL	CNF2	TXB0D3	TXB1D3	TXB2D3	RXB0D3	RXB1D3
1010	RXF2EID8	RXF5EID8	CNF1	TXB0D4	TXB1D4	TXB2D4	RXB0D4	RXB1D4
1011	RXF2EID0	RXF5EID0	CANINTE	TXB0D5	TXB1D5	TXB2D5	RXB0D5	RXB1D5
1100	BFPCTRL	TEC	CANINTF	TXB0D6	TXB1D6	TXB2D6	RXB0D6	RXB1D6
1101	TXRTSCTRL	REC	EFLG	TXB0D7	TXB1D7	TXB2D7	RXB0D7	RXB1D7
1110	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT
1111	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL

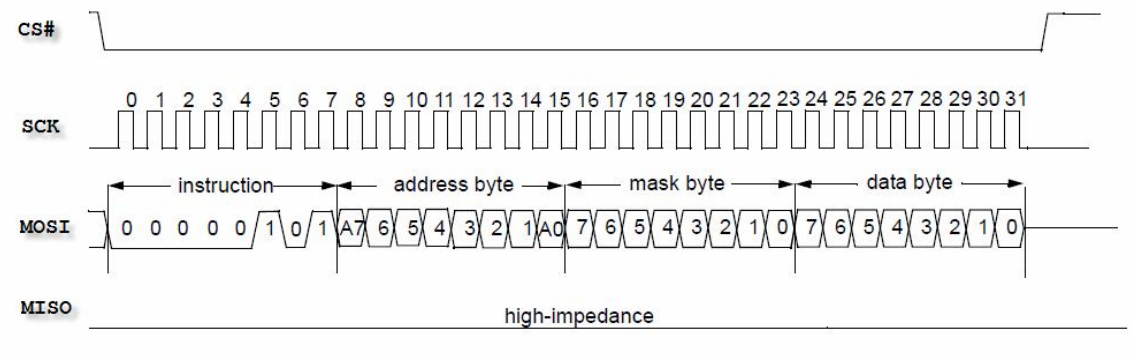
แสดง ตำแหน่งและบิตข้อมูลของรีจิสเตอร์ ที่สามารถใช้กับคำสั่ง BIT MODIFY ได้

ซึ่งการเข้าถึงของข้อมูลแบบบิต ของ MCP2515 นั้นจะไม่ได้ใช้การเข้าถึงข้อมูลระดับบิตโดยตรง เนื่องจากไม่มีรูปแบบคำสั่งการเข้าถึงแบบบิตดังได้กล่าวไปแล้ว แต่การทำงานของคำสั่ง BIT MODIFY นี้จะใช้เทคนิคพิเศษ โดยทำงานร่วมกับ Buffer พิเศษที่ออกแบบมารองรับการทำงานโดยเฉพาะ คือ Buffer ของ MASK BYTE และ DATA BYTE เพื่อนำค่าข้อมูลใน Buffer ทั้ง 2 ชุดนี้ไปกระทำทางโลจิกกับข้อมูลเดิมในรีจิสเตอร์ แล้วนำผลลัพธ์ที่ได้เขียนกลับไปให้รีจิสเตอร์ใหม่ ซึ่งกระบวนการทั้งหมดนี้กระทำอยู่ภายในชิพ ในส่วนของผู้ใช้ก็เพียงแต่กำหนดค่า DATA BYTE และ MASK BYTE ให้คำสั่งเท่านั้น

โดยคำสั่ง BIT MODIFY นี้จะมีขนาด 4 ไบท์ โดยไบท์แรกเป็นรหัสคำสั่งมีค่า 0x05 (0000 0101) ส่วนไบท์ที่ 2 เป็นค่าตำแหน่งรีจิสเตอร์ และ ค่าไบท์ที่ 3 และ ไบท์ที่4 จะเป็นค่าของ MASK BYTE และ DATA BYTE ตามลำดับ

- MASK BYTE เป็นค่าสำหรับกำหนดตำแหน่งบิตข้อมูลที่ต้องการเปลี่ยนแปลงค่า โดยถ้าต้องการเปลี่ยนแปลงแก้ไขข้อมูลในบิตใดของรีจิสเตอร์ก็ให้กำหนดค่าใน MASK BYTE ในตำแหน่งบิตเดียวกันให้มีค่าเป็น “1” ส่วนตำแหน่งบิตใดที่ไม่ต้องการเปลี่ยนแปลงค่า ก็ให้กำหนดให้มีค่าเป็น “0” ไว้
- DATA BYTE เป็นค่าที่ต้องการกำหนดเป็นค่าข้อมูลชุดใหม่ที่จะเขียนไปยังรีจิสเตอร์ ซึ่งข้อมูลนี้จะมีผลไปเปลี่ยนแปลงแก้ไขค่าใหม่ให้กับรีจิสเตอร์ของMCP2515 เฉพาะตำแหน่งที่ตรงกันกับบิตที่มีค่าเป็น ”1” ใน MASK BYTE เท่านั้น เช่นถ้ากำหนดค่าของ DATA BYTE เป็น 0x21 (0010 0001) และ กำหนดค่า MASK BYTE เป็น 0x35 (0011 0101) จะทำให้ค่าในรีจิสเตอร์ใหม่มีค่าเป็น XX10 X0X1 โดยข้อมูลใน บิตที่ 1,3,6 และ 7 จะมีค่าคงเดิมไม่เปลี่ยนแปลงตามค่าใหม่ใน DATA BYTE ดังรูป

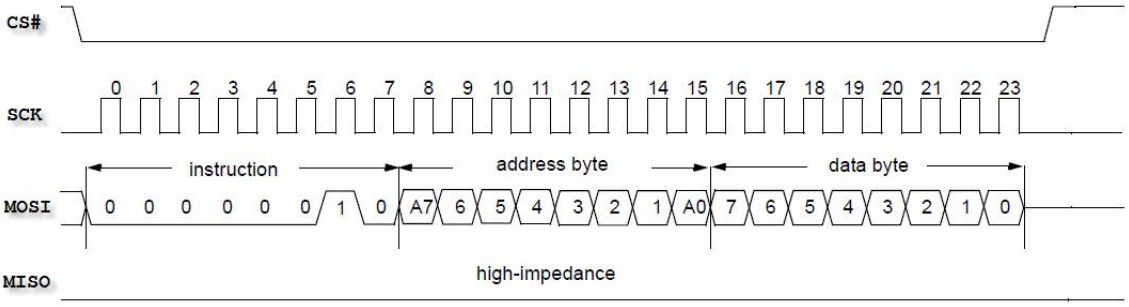




แสดง รูปสัญญาณ SPI ของคำสั่ง BIT MODIFY INSTRUCTION

BYTE WRITE INSTRUCTION

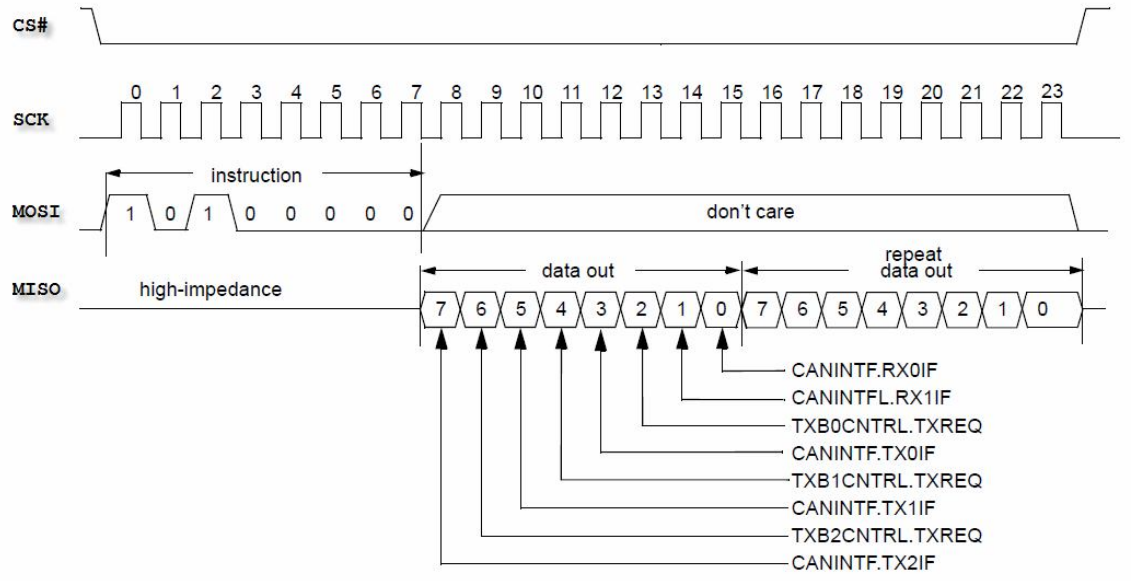
คำสั่งนี้ใช้สำหรับเขียนข้อมูลไปยังรีจิสเตอร์ของ MCP2515 โดยในชุดคำสั่งนี้จะมีจำนวน 3 ไบท์หรือมากกว่าในกรณีเขียนแบบเรียงลำดับต่อเนื่อง โดยไบท์แรกเป็นรหัสคำสั่ง มีค่าเป็น 0x02 (0000 0010) ส่วนไบท์ที่ 2 เป็นค่าตำแหน่งแอดเดรสเริ่มต้นของรีจิสเตอร์ที่ต้องการเขียนข้อมูลให้ และไบท์ที่3 เป็นค่าข้อมูลไบท์แรกที่ต้องการเขียนไปยังรีจิสเตอร์ ซึ่งถ้าต้องการเขียนข้อมูลไบท์ถัดไปให้กับรีจิสเตอร์ตำแหน่งที่อยู่ต่อเนื่องกันไปจากตำแหน่งปัจจุบันก็สามารถเขียนข้อมูลไบท์ถัดไป เรียงลำดับกันไปได้อีกครั้งละ1 ไบท์ โดยไม่จำเป็นต้องส่งรหัสคำสั่งและค่าตำแหน่งแอดเดรสเริ่มต้นใหม่แต่อย่างใด โดยเมื่อต้องการสิ้นสุดการเขียนก็ให้ทำการเปลี่ยนสัญญาณ CS# กลับเป็น “1” โดยมีรูปแบบของสัญญาณในการสื่อสารดังนี้



แสดง รูปสัญญาณ SPI ของคำสั่ง BYTE WRITE INSTRUCTION

READ STATUS INSTRUCTION

คำสั่งนี้ใช้สำหรับสั่งอ่านค่าสถานะ Status เพื่อนำค่ามาตรวจสอบผลการเปลี่ยนแปลงของบิตแสดงสถานะต่างๆ สำหรับตรวจสอบการ รับ หรือ ส่ง ข้อมูล โดยชุดคำสั่งนี้จะมีขนาด 2 ไบท์ หรือมากกว่าในกรณีต้องการวนรอบอ่านค่าสถานะอย่างต่อเนื่อง โดยไบท์แรกจะเป็นรหัสคำสั่ง มีค่าเป็น 0xA0 (1010 0000) ส่วนไบท์ที่ 2 จะเป็นค่าของ STATUS ที่ส่งกลับมาจาก MCP2515 ซึ่งถ้ายังต้องการอ่านค่าของสถานะซ้ำใหม่อีกก็สามารถเขียนค่าข้อมูลใดๆไปให้กับ MCP2515 ต่อเนื่องไปอีก โดยลำดับจากนี้ไป MCP2515 จะไม่สนใจค่าของข้อมูลที่ MCU เขียนออกไปให้ทางขา MOSI แต่จะส่งข้อมูลสวนทางกลับออกมาให้ทางขา MISO ตามจังหวะของสัญญาณ SCK ที่ได้รับ โดยค่าของข้อมูลแต่ละไบท์ที่ส่งย้อนกลับออกมานั้นจะเป็นค่าของ สถานะ STATUS ที่มีการเปลี่ยนแปลงล่าสุดในขณะนั้น โดยเราสามารถวนอ่านข้อมูลนี้ออกมาได้อย่างต่อเนื่องไม่มีจำกัดจำนวน โดยเมื่อต้องการสิ้นสุดการทำงานของคำสั่งก็ให้ MCU สั่งเปลี่ยนสัญญาณของ CS# ให้กลับเป็น “1” เท่านั้นเอง

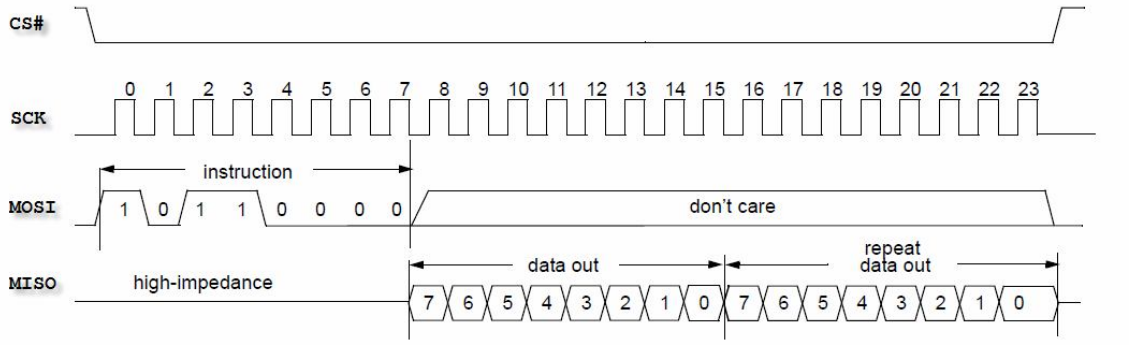


แสดง รูปสัญญาณ SPI ของคำสั่ง READ STATUS INSTRUCTION

RX STATUS INSTRUCTION

คำสั่งนี้ใช้สำหรับสั่งให้ MCP2515 ตรวจสอบสถานะของ ชุดข้อมูลที่รับได้ใน RX Buffer โดยเมื่อ MCP2515 ได้รับคำสั่งนี้จะส่งค่าไบท์ข้อมูล ซึ่งใช่บ่งบอก ความหมายของข้อมูล เช่น ชนิดของชุดข้อมูลที่ตรวจจับได้ ว่าเป็นแบบ Standard Frame หรือ Extended Frame หรือ Remote Frame เป็นต้น โดยคำสั่งนี้จะมีขนาด 2 ไบท์ หรือมากกว่าได้ในกรณีต้องการวนรอบอ่านซ้ำอย่างต่อเนื่อง

โดยไบท์แรกจะเป็นรหัสคำสั่ง มีค่า 0xB0 (1011 0000) ซึ่งเมื่อ MCP2515 ได้รับรหัสคำสั่งนี้แล้ว ในไบท์ที่2 มันจะส่งค่าสถานะ (RX Status) ของ ชุดข้อมูลที่ตรวจจับได้ในขณะนั้นกลับออกมาให้



7	6	Received Message
0	0	No RX message
0	1	Message in RXB0
1	0	Message in RXB1
1	1	Messages in both buffers*

CANINTF.RXnIF bits are mapped to bits 7 and 6.

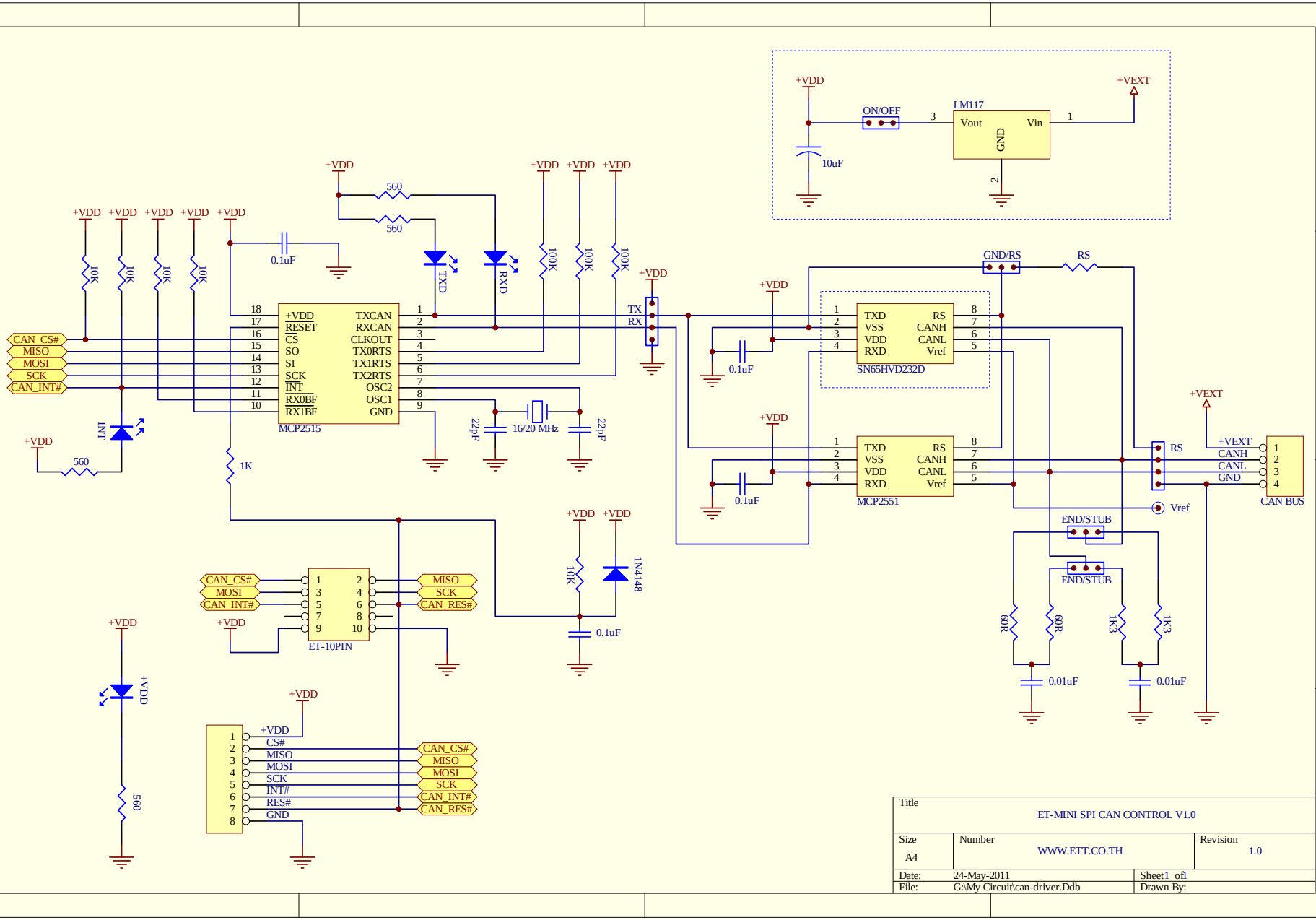
4	3	Msg Type Received
0	0	Standard data frame
0	1	Standard remote frame
1	0	Extended data frame
1	1	Extended remote frame

The extended ID bit is mapped to bit 4. The RTR bit is mapped to bit 3.

2	1	0	Filter Match
0	0	0	RXF0
0	0	1	RXF1
0	1	0	RXF2
0	1	1	RXF3
1	0	0	RXF4
1	0	1	RXF5
1	1	0	RXF0 (rollover to RXB1)
1	1	1	RXF1 (rollover to RXB1)

* Buffer 0 has higher priority, therefore, RXB0 status is reflected in bits 4:0.

แสดง รูปสัญญาณ SPI ของคำสั่ง READ RX STATUS INSTRUCTION


+VDD
+VEXT
ON/OFF
LM117
Vout
Vin
GND
10uF
MCP2515
SN65HVD232D
MCP2551
ET-10PIN
CAN BUS
CAN_CS#
MISO
MOSI
SCK
CAN_INT#
CAN_RES#
TXD
RXD
INT
RS
GND/RS
END/STUB
Vref
16/20 MHz
22pF
0.1uF
0.01uF
560
1K
10K
100K
60R
1K3
1N4148
Title
ET-MINI SPI CAN CONTROL V1.0
Size
A4
Number
WWW.ETT.CO.TH
Revision
1.0
Date:
24-May-2011
Sheet1 of1
File:
G:\My Circuit\can-driver.Ddb
Drawn By: