การประยุกต์ใช้ไมโครคอนโทรลเลอร์อาร์มคอร์เท็กเอ็ม 3 สำหรับระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์แบบไร้สาย

APPLICATION OF ARM CORTEX-M3 MICROCONTROLLER FOR WIRELESS CONTROLLER AREA NETWORK SYSTEM

วุฒิชัย บัวนาค

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
ปีการศึกษา 2554

การประยุกต์ใช้ไมโครคอนโทรลเลอร์อาร์มคอร์เท็กเอ็ม 3 สำหรับระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์แบบไร้สาย

วุฒิชัย บัวนาค

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
ปีการศึกษา 2554

หัวข้อวิทยานิพนธ์ การประยุกต์ใช้ไมโครคอนโทรลเลอร์อาร์มคอร์เท็กเอ็ม 3 สำหรับระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์แบบไร้สาย
Application of ARM Cortex-M3 Microcontroller for Wireless Controller Area Network System

ชื่อ – นามสกุล นายวุฒิชัย บัวนาค

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ที่ปรึกษา ดร. จักรี ศรีนนท์ฉัตร

ปีการศึกษา 2554

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ดร. อำนวย เรืองวารี)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. คมกฤตย์ ชุมสุวรรณ)

.....กรรมการ
(ดร. สุรินทร์ แหงมงาม)

.....กรรมการ
(ดร. จักรี ศรีนนท์ฉัตร)

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี อนุมัติวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(ผู้ช่วยศาสตราจารย์ ดร. สมหมาย ผิวสอาด)
วันที่ 18 เดือน มีนาคม พ.ศ. 2554

หัวข้อวิทยานิพนธ์	การประยุกต์ใช้ไมโครคอนโทรลเลอร์อาร์มคอร์เท็กเอ็ม 3 สำหรับระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์แบบไร้สาย
ชื่อ-นามสกุล	นายวุฒิชัย บัวนาค
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	ดร. จักรี ศรีนนท์ฉัตร
ปีการศึกษา	2554

บทคัดย่อ

ระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ (Controller Area Network) อาศัยการรับ-ส่งข้อมูลด้วยระบบบัส (Bus) ซึ่งมีข้อจำกัดการเชื่อมต่อของระบบในกรณีที่ต้องมีการทำงานในรูปแบบที่ต้องมีการเคลื่อนที่ วิทยานิพนธ์นี้นำเสนอการพัฒนาและออกแบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ในรูปแบบไร้สาย เพื่อเพิ่มประสิทธิภาพของการทำงานในรูปแบบที่ต้องมีการเคลื่อนที่

งานวิจัยนี้ใช้ไมโครคอนโทรลเลอร์อาร์มคอร์เท็กเอ็ม 3 (ARM Cortex-M3) ขนาด 32 บิต สำหรับการควบคุมรับ-ส่งข้อมูลในรูปแบบฐานเวลาจริง (Real-time) และในระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ในรูปแบบไร้สายใช้ย่านความถี่ 2.4 กิกะเฮิรตซ์ ระบบได้ถูกทดสอบด้วยการสร้างชุดการรับ-ส่งข้อมูลแบบไร้สายจำนวน 4 ชุด

ผลการทดสอบพบว่าไมโครคอนโทรเลอร์อาร์มคอร์เท็กเอ็ม 3 ขนาด 32 บิต สามารถนำมาออกแบบและพัฒนาระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์แบบไร้สายได้ ทั้งนี้มีอัตราความเร็วการรับ-ส่งข้อมูล 1 เมกกะบิตต่อวินาที และมีระยะในการส่งสูงสุดที่ 10 เมตร โดยมีค่าการหน่วงเวลาอยู่ที่ 360 ไมโครวินาที ทั้งนี้หากระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ ถูกบดบังหรือจำเป็นต้องส่งข้อมูลผ่านสิ่งกีดขวาง ระยะทางในการรับ-ส่งข้อมูลจะลดลงแต่อัตราความเร็วในการรับ-ส่งข้อมูลจะยังเท่าเดิม ดังนั้นการศึกษาและพัฒนารูปแบบการรับส่งข้อมูลของระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์แบบไร้สายนี้สามารถนำมาประยุกต์ใช้กับระบบสมองกลฝังตัวในอนาคตได้

คำสำคัญ : ระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ อาร์มคอร์เท็กเอ็ม 3
ระบบสมองกลฝังตัว

Thesis Title	Application of ARM Cortex-M3 Microcontroller for Wireless Controller Area Network System
Name-Surname	Mr. Wuttichai Buanark
Program	Electrical Engineering
Thesis Advisor	Dr. Jakkree Srinonchat
Academic Year	2011

ABSTRACT

Controller Area Network (CAN) system is operated in transmitted-received BUS communication which has a connectivity limitation because of the independent moving system. This thesis presents the development and design of the wireless CAN system to increase the efficiency of independent moving system.

This research uses the ARM Cortex-M3 microcontroller 32 bits for transmitted-received data control in real time system. Also the 2.4 GHz wireless communication technique is applied to this wireless CAN system. Four wireless CAN modules are designed to transmitted-received data for this experiment.

The experiment results show that the 32 bits microcontroller of ARM Cortex-M3 can use to design and develop the wireless CAN. It operates at 1Mbps for bit rate ratio and the maximum distance between each module is 10 m. There is a delay time at 360 μs. If the CAN module is blocked or transferred data through obstruction, the distance of operation will be reduced but the bit rate ratio is still the same. Therefore, the study and development of data communication of wireless CAN system can be apply for the future embedded system.

Keywords : Control Area Network, ARM Cortex-M3, Embedded System

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความเมตตากรุณาอย่างสูงจาก ดร. จักรี ศรีนนท์ฉัตร ที่กรุณาให้คำแนะนำและให้คำปรึกษาตลอดจนให้ความช่วยเหลือแก้ไขข้อบกพร่องต่างๆ เพื่อให้วิทยานิพนธ์ฉบับนี้มีความถูกต้องสมบูรณ์ และขอขอบคุณ ดร. อำนวย เรืองวารี ผู้ช่วยศาสตราจารย์ ดร. คมกฤตย์ ชุมสุวรรณ และ ดร. สุรินทร์ แหงมงาม ที่ได้ให้ข้อเสนอแนะและข้อคิดเห็นอื่นๆ ซึ่งผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบคุณเพื่อนนักศึกษาในระดับบัณฑิตศึกษาทุกท่านและขอขอบคุณ บริษัทอินโดเท็คออโตเมชั่น (ประเทศไทย) จำกัด ที่ให้คำปรึกษาและให้คำแนะนำในการทำงานวิจัยสำหรับงานวิจัยนี้ ผู้วิจัยขอขอบพระคุณคณาจารย์ที่ได้ถ่ายทอดความรู้อันประเสริฐในระหว่างที่ได้ทำการศึกษาในระดับบัณฑิตศึกษานี้

คุณค่าอันพึงมีจากวิทยานิพนธ์ฉบับนี้ ขอมอบเพื่อบูชาพระคุณบิดา มารดา ครู อาจารย์ และผู้มีพระคุณทุกท่าน

วุฒิชัย บัวนาค

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ค

บทคัดย่อภาษาอังกฤษ ..... ง

กิตติกรรมประกาศ ..... จ

สารบัญ ..... ฉ

สารบัญตาราง ..... ซ

สารบัญภาพ ..... ฌ

บทที่

1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 ความมุ่งหมายต่องานวิจัย 2

1.3 วัตถุประสงค์ของงานวิจัย 2

1.4 ขอบเขตการวิจัย 2

1.5 ข้อจำกัดหรือข้อยกเว้นในการวิจัย 2

1.6 ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย 3

2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 4

2.1 ระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ 4

2.2 ไมโครคอนโทรลเลอร์ แบบ ARM Cortex-M3 12

2.3 เครือข่ายไร้สายที่ใช้ในงานวิจัย 30

2.4 งานวิจัยที่เกี่ยวข้อง 33

3 วิธีดำเนินการวิจัย 35

3.1 บทนำ 35

3.2 หลักการทำงานของระบบ 35

3.3 แนวทางการออกแบบและพัฒนาระบบ 36

3.4 การออกแบบฮาร์ดแวร์ 36

3.5 การออกแบบซอฟแวร์ 45

4 ผลการวิจัย 50

4.1 การทดสอบการทำงานของอุปกรณ์หลักบนบอร์ดโมดูล 50

4.2 สภาพแวดล้อมในการทดสอบ 52

สารบัญ (ต่อ)

บทที่ หน้า

4.3 การทดสอบการทำงานของระบบเบื้องต้น..... 52

4.4 การทดสอบการทำงานของระบบ..... 55

5 สรุปผลการทดลอง..... 60

5.1 สรุปผลการทดลอง 60

5.2 แนวทางในการพัฒนาและปรับปรุงงานวิจัย..... 61

รายการอ้างอิง..... 62

ภาคผนวก..... 63

ภาคผนวก ก ซอร์ซโค้ด..... 64

ภาคผนวก ข วงจรสมบูรณ์..... 82

ภาคผนวก ค ผลงานตีพิมพ์เผยแพร่..... 84

ประวัติผู้เขียน..... 105

สารบัญตาราง

ตารางที่ หน้า

2.1 มาตรฐานของ CAN…..……..... 8

2.2 กระบวนการไป๊ปไลน์ของ Cortex-M3 โปรเซสเซอร์....……..…..…… 15

2.3 การเปรียบเทียบความถี่ที่ใช้ในช่องสัญญาณต่างๆ ตามมาตรฐาน IEEE 802.11g ที่ความถี่ 2.4 GHz……... 32

3.1 ขาคอนเน็คเตอร์ของโมดูลไร้สาย…….……..... 43

5.1 เปรียบเทียบระยะทางและผลการรับ-ส่งข้อมูล....…….…..... 61

สารบัญภาพ

ภาพที่ หน้า

2.1 รูปแบบการเชื่อมต่ออุปกรณ์บนบัสแคน…..... 5

2.2 กราฟคุณสมบัติสายสัญญาณที่สัมพันธ์กับความยาวของสายและอัตราการถ่ายทอดข้อมูลของบัสแคน..... 5

2.3 ระดับสัญญาณที่เกิดขึ้นบนบัสแคน..... 6

2.4 ตัวอย่างการเชื่อมต่ออุปกรณ์รับส่งบัสแคน หรือ แคนทรานซีฟเวอร์ (CAN Transceiver) เข้ากับบัสแคน..... 7

2.5 การแปลงสัญญาณ TTL เป็นสัญญาณข้อมูลบนบัสแคน ของแคนทรานซีฟเวอร์..... 7

2.6 สถาปัตยกรรมของอุปกรณ์บัสแคนภายใต้มาตรฐาน ISO11898……..…… 8

2.7 รูปแบบข้อมูลสำหรับการระบุอุปกรณ์สำหรับบัสแคนมาตรฐาน…..... 9

2.8 รูปแบบข้อมูลสำหรับการระบุอุปกรณ์สำหรับบัสแคนแบบขยาย……..……... 10

2.9 บล็อกไดอะแกรมไมโครคอนโทรลเลอร์แบบ Cortex-M3……..... 13

2.10 ไดอะแกรมการทำงานของ Cortex-M3 โปรเซสเซอร์……... 17

2.11 ไดอะแกรมระบบไมโครคอนโทรลเลอร์……..…….. 20

2.12 ไดอะแกรมของการเกิดรีเซตในไมโครคอนโทรลเลอร์ Cortex-M3……..…… 21

2.13 โครงสร้างและส่วนประกอบของไมโครคอนโทรลเลอร์ STM32F103……..... 23

2.14 การจัดสรรหน่วยความจำแฟลชของไมโครคอนโทรลเลอร์ STM32F103……..... 24

3.1 หลักการทำงานของระบบ…… 35

3.2 ผังงานการออกแบบฮาร์ดแวร์…… 37

3.3 วงจรการทำงานของภาคจ่ายไฟ 3.3V…… 38

3.4 วงจรการทำงานของแคนทรานซีฟเวอร์ ……. 38

3.5 วงจรการทำงานของไมโครคอนโทรลเลอร์……..…….. 40

3.6 คอนเนคเตอร์ JTAG ของไมโครคอนโทรลเลอร์…….. 41

3.7 วงจรอินพุต-เอาท์พุตของไมโครคอนโทรลเลอร์……..…….. 42

3.8 วงจรติดต่อสื่อสารแบบ RS232 ของไมโครคอนโทรลเลอร์……..…… 42

3.9 วงจรการติดต่อโมดูล RF แบบ SPI…… 42

3.10 วงจรภายในโมดูล NRF24L01……... 43

3.11 ลายวงจรของบอร์ดโมดูลที่ใช้ทดสอบ…….. 44

3.12 บอร์ดโมดูลสำเร็จสำหรับทดสอบการทำงานของระบบ……... 45

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

3.13 ผังการออกแบบซอฟแวร์..... 46

3.14 ผังการออกแบบซอฟแวร์ภาคประมวลผลข้อมูล..... 47

3.15 ผังการทำงานของภาครับส่งข้อมูลไร้สาย..... 49

4.1 การวัดสัญญาณ CAN High และ CAN Low..... 51

4.2 การวัดสัญญาณ SPI ระหว่างไมโครคอนโทรลเลอร์กับโมดูลไร้สาย..... 51

4.3 สภาพแวดล้อมการทดสอบของระบบที่ระยะทางแตกต่างกัน..... 52

4.4 การทดสอบโมดูลเบื้องต้น..... 53

4.5 วัดสัญญาณระหว่างไมโครคอนโทรลเลอร์กับโมดูล NRF24L01 (ส่งข้อมูล)..... 53

4.6 วัดสัญญาณระหว่างไมโครคอนโทรลเลอร์กับโมดูล NRF24L01 (รับข้อมูล)..... 54

4.7 การรับ-ส่งข้อมูลระหว่างสองโมดูล..... 55

4.8 การทดสอบโมดูล 4 ชุดในระยะใกล้กัน..... 56

4.9 ทดสอบส่งข้อความแคนให้กับโมดูล..... 56

4.10 วัดสัญญาณ SPI ระหว่างการรับ-ส่งข้อมูลไร้สายทั้งโมดูลตัวส่งและโมดูลตัวรับ..... 57

4.11 ข้อความแคนที่ผ่านการรับ-ส่งแล้ว..... 57

4.12 ทดสอบโมดูลโดยเพิ่มระยะทางที่แตกต่างกัน..... 58

4.13 ข้อความแคน ที่ผ่านการรับ-ส่งแล้วแบบระยะทางต่างกัน..... 58

4.14 ทดสอบโมดูลโดยกำหนดระยะทางที่แตกต่างกัน..... 59

4.15 ข้อความแคนที่สามารถรับ-ส่งได้แบบระยะทางต่างกัน..... 59

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

Controller Area Network หรือ แคน (CAN) เป็นชื่อของเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ ถูกนำมาประยุกต์ใช้งานร่วมกับระบบการเชื่อมต่อโครงข่ายควบคุม ได้รับการพัฒนาขึ้นโดย โรเบิร์ต บอสช (Robert Bosch) ตั้งแต่ปี ค.ศ. 1980 เพื่อใช้ในการสื่อสารข้อมูลระหว่างแผงควบคุมอิเล็กทรอนิกส์ในรถยนต์หรือยานพาหนะ โดยมีรูปแบบการทำงานหรือการสื่อสารหลายจุดเชื่อมต่อบนสายชุดเดียวกันหรือแบบบัส (Bus) ต่อมาได้รับการพัฒนาและมีมาตรฐานทางอุตสาหกรรมรองรับ ทำให้บัสแคน (Bus CAN) ได้รับความนิยมและมีการใช้งานอย่างแพร่หลายมากขึ้น ไม่เพียงแค่ในอุตสาหกรรมยานยนต์เท่านั้น

จากการที่ได้ศึกษาข้อมูลงานวิจัยที่เกี่ยวข้อง พบว่าการพัฒนาการสื่อสารในรูปแบบแคนเป็นแบบไร้สายได้มีผู้พัฒนาได้ทำการทดสอบไว้แต่การเปลี่ยนรูปแบบมาเป็นไร้สายนั้นมีข้อจำกัดอยู่มาก เพราะไม่สามารถนำรูปแบบการสื่อสารแบบแคน มาเปลี่ยนรูปแบบการสื่อสารเป็นรูปแบบไร้สายได้โดยตรงหรือโดยการแปลงให้อยู่ในรูปแบบอื่นก่อนแล้วส่งข้อมูลในรูปแบบไร้สาย เนื่องด้วยการออกแบบการสื่อสารของแคน นั้นถูกออกแบบให้อยู่ในรูปแบบของบัส จึงเกิดปัญหาเวลามีการรับส่งสัญญาณพร้อมกันหลายจุดเพราะไม่สามารถรับประกันได้ว่าข้อมูลนั้นจะส่งจากจุดใดไปจุดใด ดังนั้นการเปลี่ยนรูปแบบการสื่อสารของแคน มาเป็นมาเป็นรูปแบบไร้สาย สามารถทำได้แต่ต้องรับ-ส่งแบบจุดต่อจุดหรือตัวใดตัวหนึ่งเป็นตัวส่งและอีกตัวเป็นตัวรับ ซึ่งทำให้สูญเสียคุณลักษณะที่ดีของระบบแคน และความเร็วในการทำงานของระบบนั้นก็ถูกลดทอนลงไป กล่าวคือ ความสามารถในการรับส่งของแคนนั้นสามารถรับส่งได้ถึง 1 เมกกะบิตต่อวินาทีแต่จากที่ได้ศึกษามาสามารถรับส่งได้ไม่เกิน 500 กิโลบิตต่อวินาที ซึ่งเป็นแค่มาตรฐานการเชื่อมต่อเท่านั้น ในงานวิจัยนี้ ได้นำเสนอการพัฒนารูปแบบและการออกแบบเพื่อสร้างอุปกรณ์เพื่อทดสอบการทำงานโดยใช้ไมโครคอนโทรลเลอร์ขนาด 32 บิต แบบ ARM Cortex-M3 และโมดูล RF 2.4GHz ในการทดสอบการทำงานของงานวิจัย โดยมุ่งเน้นเพื่อนำผลของการพัฒนางานวิจัยมาใช้งานได้ และสามารถนำไปพัฒนาต่อเพิ่มเติมได้อีกต่อไป

1.2 ความมุ่งหมายต่องานวิจัย

ข้อจำกัดของการสื่อสารข้อมูลแบบแคน นั้นโดยหลักแล้วจะทำงานในรูปแบบบัส ถึงแม้จะมีความยาวและมีการเชื่อมต่อโนดหลายตัวแต่ถ้ามีอุปกรณ์บางชนิดนั้นต้องมีการเคลื่อนที่อยู่ตลอด

การเชื่อมต่อในรูปแบบนี้ก็จะไม่สามารถตอบสนองความต้องการได้ ดังนั้นการพัฒนารูปแบบการสื่อสารข้อมูลในรูปแบบของแคน นั้นจะช่วยให้สามารถทำการเชื่อมต่ออุปกรณ์ที่ใช้ได้โดยที่ไม่ต้องมีการเปลี่ยนแปลงรูปแบบการสื่อสารเดิมไปเป็นรูปแบบอื่นซึ่งเป็นการเพิ่มภาระให้กับการออกแบบโปรแกรมหรือระบบหลักให้ครอบคลุมอุปกรณ์ที่ต้องใช้รูปแบบการสื่อสารแบบไร้สาย จากการพัฒนาวิธีสื่อสารแคนแบบไร้สาย ที่ได้ศึกษานั้นไม่สามารถรับส่งในกรณีที่มีปริมาณของข้อมูลและมีโนดที่ใช้ในการเชื่อมต่อหลายจุด จากปัญหาการรับส่งข้อมูลโดยใช้ ไมโครคอนโทรลเลอร์ที่มีการประมวลผลที่ 8 บิต ทำให้เกิดการสูญเสียของข้อมูลสูงและไม่สามารถทำงานในกรณีที่มีการรับ-ส่งข้อมูลที่มีปริมาณมาก จึงเกิดแนวคิดในการพัฒนารูปแบบการทำงานโดยใช้ ไมโครคอนโทรลเลอร์ที่มีการประมวลผลที่ 32 บิต เพื่อที่จะทำให้การรับส่งมีโครงสร้างและรูปแบบการทำงานที่ดีขึ้นในการจัดวางรูปแบบการรับ-ส่งข้อมูลเพื่อให้เกิดการสูญเสียที่น้อยลงและความเร็วในการรับ-ส่งระหว่างกันมีมากขึ้น

1.3 วัตถุประสงค์ของงานวิจัย

พัฒนารูปแบบการสื่อสารข้อมูลในรูปแบบแคน ให้อยู่ในรูปแบบไร้สายโดยใช้ไมโครคอนโทรลเลอร์ขนาด 32 บิตแบบ ARM Cortex-M3

1.4 ขอบเขตการวิจัย

1.4.1 ออกแบบและประยุกต์ใช้ไมโครคอนโทรลเลอร์แบบ ARM Cortex-M3 สำหรับระบบแคนไร้สาย

1.4.2 สามารถรับ-ส่งข้อมูลแคนไร้สายด้วยความเร็ว 1 เมกกะบิตต่อวินาที ที่ระยะทางไม่เกิน 10 เมตร ภายในบริเวณที่สามารถรับสัญญาณได้

1.4.3 ข้อมูลแคนที่ใช้ในการรับ-ส่งต้องมีความถูกต้องของข้อมูลโดยสมบูรณ์

1.5 ข้อจำกัดหรือข้อยกเว้นในการวิจัย

1.5.1 ไม่สามารถรับส่งข้อมูลในรูปแบบของแคนที่อัตราการถ่ายทอดข้อมูลเปลี่ยนแปลงได้ เนื่องจากต้องมีการระบุอัตราการถ่ายทอดข้อมูลของโมดูลให้ตรงกับที่ใช้งานเท่านั้น

1.5.2 ระยะทางและพื้นที่ในการรับส่งข้อมูลต้องอยู่ภายในพื้นที่โดยรอบที่สามารถรับสัญญาณได้ไม่เกิน 10 เมตร และมีสัญญาณรบกวนน้อย

1.5.3 ประสิทธิภาพของโมดูลอาจลดลงถ้ามีการเพิ่มจำนวนโนดระหว่างต้นทางและปลายทางมากขึ้น

1.6 ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย

1.6.1 ทำให้ทราบรูปแบบและวิธีการพัฒนาสื่อสารข้อมูลในรูปแบบแคนสำหรับระบบไร้สายโดยใช้ไมโครคอนโทรลเลอร์แบบ ARM Cortex-M3

1.6.2 สามารถนำมาทดแทนการสื่อสารในรูปแบบบัสแคนได้โดยตรง โดยไม่ต้องทำการเปลี่ยนแปลงระบบเดิม

1.6.3 สามารถนำงานวิจัยมาประยุกต์ใช้งานร่วมกับระบบอุตสาหกรรมได้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ระบบเครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์

เครือข่ายของการติดต่อระหว่างไมโครคอนโทรลเลอร์ หรือ แคน (Controller Area Network): CAN) [1-2] เป็นเครือข่ายที่ใช้ติดต่อกันระหว่างไมโครคอนโทรลเลอร์ (Micro Controller Unit: MCU) ซึ่งเป็นระบบการเชื่อมต่อโครงข่ายควบคุม ได้รับการพัฒนาขึ้น โดย โรเบิร์ต บอสช (Robert Bosch) จาก Bosch บริษัทยักษ์ใหญ่ในอุตสาหกรรมชิ้นส่วนยานยนต์จากเยอรมัน ตั้งแต่ปี ค.ศ. 1980 เพื่อใช้ในการสื่อสารข้อมูลระหว่างแผงควบคุมอิเล็กทรอนิกส์ในรถยนต์หรือยานพาหนะ ต่อมาได้รับการพัฒนาและมีมาตรฐานทางอุตสาหกรรมรองรับ ทำให้แคน ได้รับความนิยมและมีการใช้งานอย่างแพร่หลายมากขึ้น ไม่เพียงแค่ในอุตสาหกรรมยานยนต์เท่านั้น ระบบแคน ยังถูกนำมาประยุกต์ใช้งานในรูปแบบอื่นๆ เช่นระบบควบคุมภายในโรงงานอุตสาหกรรม ระบบสมองกลฝังตัว ระบบการทำงานอัตโนมัติเป็นต้น

2.1.1 คุณสมบัติทางเทคนิคที่สำคัญของระบบแคน

1) เป็นระบบโครงข่ายที่รองรับการทำงานแบบมัลติมาสเตอร์

2) สายสัญญาณหลักสำหรับการถ่ายทอดข้อมูล 2 เส้น คือ CAN High และ CAN Low

3) อัตราการถ่ายทอดข้อมูลสูงถึง 1 เมกะบิตต่อวินาที (Mbps) ที่ความยาวสายสัญญาณ 40 เมตร ส่วนอัตราถ่ายทอดข้อมูลปกติอยู่ที่ 40,960 บิตต่อวินาทีหรือ 40 กิโลไบต์ต่อวินาที

4) ความยาวของสายสัญญาณสูงสุด 10 กิโลเมตร มีอัตราการถ่ายทอดข้อมูล 5 กิโลบิตต่อวินาที

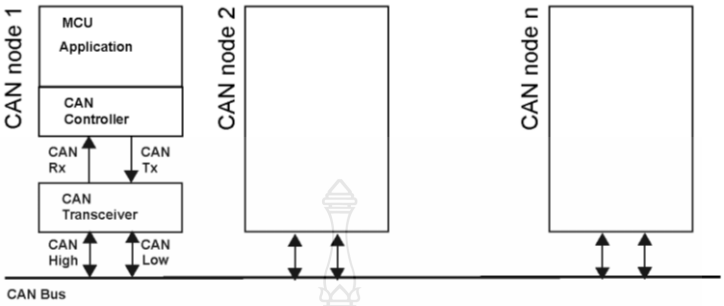
5) ค่าเวลารับรู้ข้อความน้อยกว่า 120 ไมโครวินาทีที่อัตราการถ่ายทอดข้อมูล 1 เมกะบิตต่อวินาที

6) มีความน่าเชื่อถือสูงเนื่องจากมีระบบการตรวจสอบข้อมูลผิดพลาดที่เคร่งครัด และสามารถถ่ายทอดข้อมูลซ้ำได้อย่างอัตโนมัติ หากข้อมูลเกิดการสูญหาย และถูกทำลายจากความผิดพลาด

7) สามารถปิดการติดต่อกับโนดที่ไม่ทำงานได้อัตโนมัติ

8) เนื่องจากการติดต่อไม่ได้อ้างถึงแอดเดรสที่ตายตัว แต่ใช้ข้อมูลที่เรียกว่า ข้อความ แคน หรือ CAN Message เพื่อจัดระดับนัยสำคัญแทน ดังนั้นในทางทฤษฎีจึงสามารถต่ออุปกรณ์เข้ามาในโครงข่ายได้ไม่จำกัด

9) มาตรฐานที่ใช้งานในปัจจุบันคือ ISO 11898 เป็นบัส CAN2.0B หรือ Extended CAN ใช้ข้อความแคน ขนาด 29 บิต ในการระบุโนด และมีอัตราการถ่ายทอดข้อมูลสูงสุด 1 เมกะบิตต่อวินาที

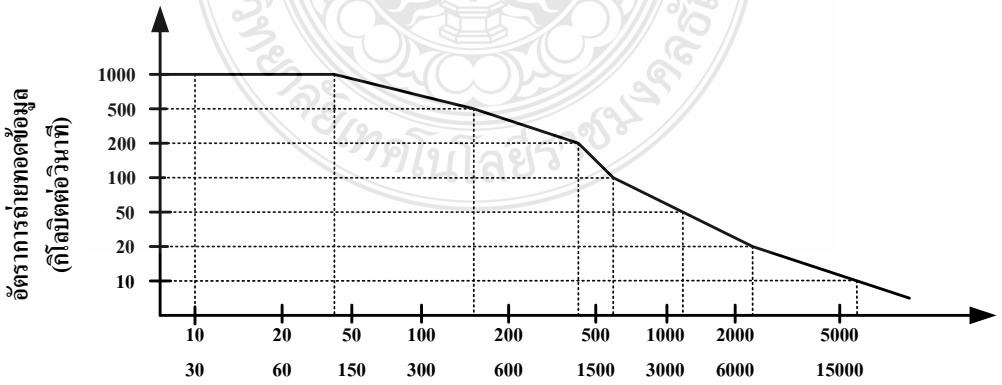


ภาพที่ 2.1 รูปแบบการเชื่อมต่ออุปกรณ์บนบัสแคน

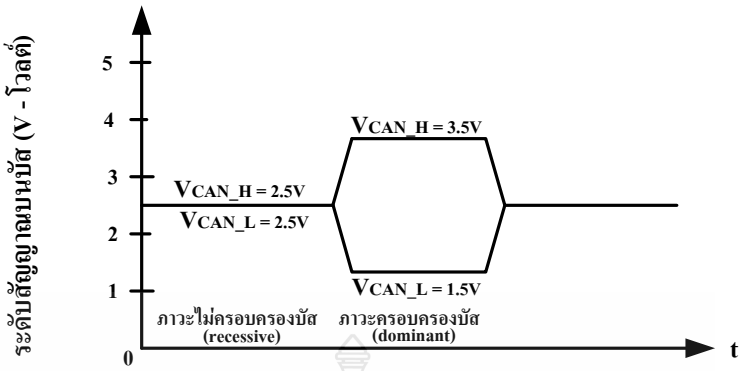
ความยาวของสายสัญญาณ มีสายสัญญาณหลัก 2 เส้นคือ CAN_H และ CAN_L โดยระดับแรงดันในขณะทำงานที่สาย CAN_H คือ 3.5V ส่วนที่สาย CAN_L มีระดับแรงดันที่ 1.5V

ความยาวบัส (L) มีความยาวสูงสุด 120 ฟุต หรือ 40 เมตร มีความยาวของสายเคเบิลจากอุปกรณ์แต่ละโนดที่นำมาต่อเข้ากับบัส (l) มีความยาวสูงสุด 1 ฟุตหรือ 30 เซนติเมตร ระยะห่างระหว่างอุปกรณ์ในแต่ละโนด (d) สูงสุด 120 ฟุต หรือ 40 เมตร

ความยาวของสายสัญญาณสามารถเพิ่มขึ้นได้ แต่ต้องลดอัตราการถ่ายทอดข้อมูลลง นั่นคือถ้ายอมรับอัตราเร็วในการถ่ายทอดข้อมูลที่ต่ำลงจากค่า 1 เมกะบิตต่อวินาที ก็จะสามารถเพิ่มความยาวของสัญญาณและบัสได้ ที่ความยาวของบัสมากกว่า 100 เมตร หรือ 300 ฟุต อัตราการถ่ายทอดข้อมูลจะไม่มีทางสูงกว่า 600 กิโลบิตต่อวินาที



ภาพที่ 2.2 กราฟคุณสมบัติสายสัญญาณที่สัมพันธ์กับความยาวของสายและอัตราการถ่ายทอดข้อมูลของบัสแคน



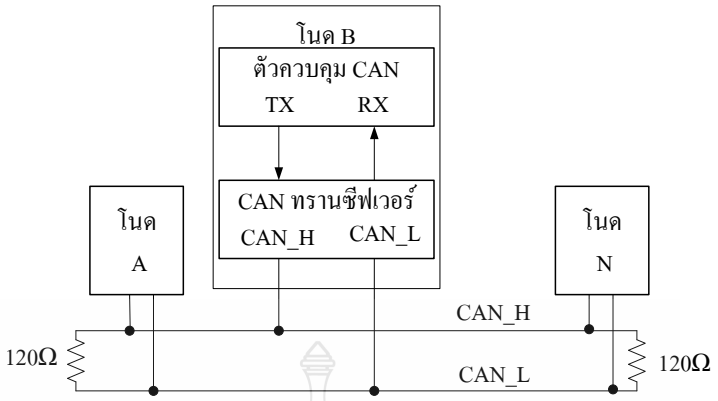
ภาพที่ 2.3 ระดับสัญญาณที่เกิดขึ้นบนบัสแคน

2.1.3 ระดับสัญญาณบนบัสแคน

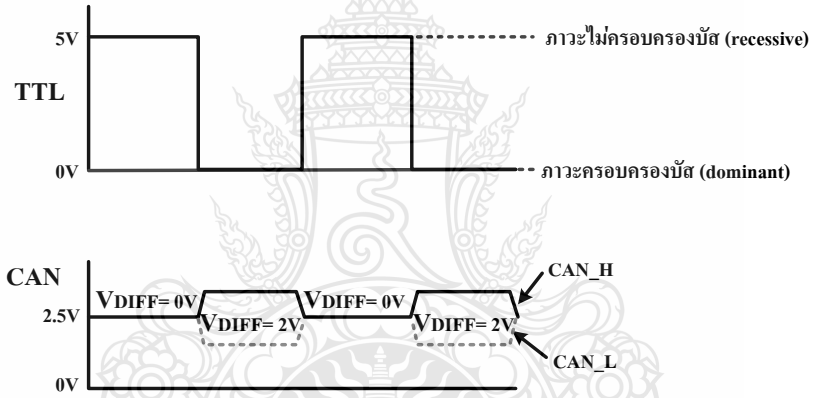
จากภาพที่ 2.3 ระดับสัญญาณบนบัสแคน สัญญาณที่เกิดขึ้นบนบัสแคน เมื่อมีการถ่ายทอดข้อมูลหรือเกิดภาวะครอบครองบัสของโนดใดๆ (Dominant) ในภาวะปกติหรือภาวะที่โนดถอนตัวจากบัส (Recessive) ระดับแรงดันบนสายสัญญาณ CAN_H และ CAN_L จะเท่ากันคือ 2.5V ทำให้ความต่างศักย์ (Vdiff) บนบัสเป็น 0V ทำให้โอกาสที่สัญญาณจากภายนอกจะเข้ามารบกวนการทำงานบนบัสเกิดขึ้นได้ยาก เมื่อเกิดการถ่ายทอดข้อมูลหรือภาวะครอบครองบัส ระดับสัญญาณบนสาย CAN_H จะสูงขึ้นเป็น 3.5V และสาย CAN_L มีระดับแรงดันลดลงเป็น 1.5V ความต่างศักย์หรือ Vdiff จะเป็น 2V อันเป็นแสดงให้เห็นถึงการเกิดปรากฏตัวของบิตข้อมูลบนบัสแคน

2.1.4 การเชื่อมต่ออุปกรณ์เข้ากับบัสแคน

อุปกรณ์ที่จำเป็นอย่างยิ่งในการช่วยให้ไมโครคอนโทรลเลอร์หรืออุปกรณ์ทางดิจิตอลสามารถเชื่อมต่อกับบัสแคนคือ อุปกรณ์รับส่งบัสแคนหรือ แคนทรานซีฟเวอร์ (CAN Transceiver) ดังแสดงตัวอย่างการเชื่อมต่อโดย แคนทรานซีฟเวอร์จะแปลงสัญญาณข้อมูลจากไมโครคอนโทรลเลอร์ที่ส่วนใหญ่เป็นระดับ TTL ทั้งแบบ +5V และ +3.3V ให้กลายเป็นสัญญาณข้อมูลสำหรับบัสแคน แสดงการเชื่อมต่ออุปกรณ์รับส่งบัสแคน ในภาพที่ 2.4 และการแปลงสัญญาณ TTL เป็นสัญญาณข้อมูลบนบัสแคนในภาพที่ 2.5



ภาพที่ 2.4 ตัวอย่างการเชื่อมต่ออุปกรณ์รับส่งบัสแคน หรือ แคนทรานซีฟเวอร์ (CAN transceiver) เข้ากับบัสแคน

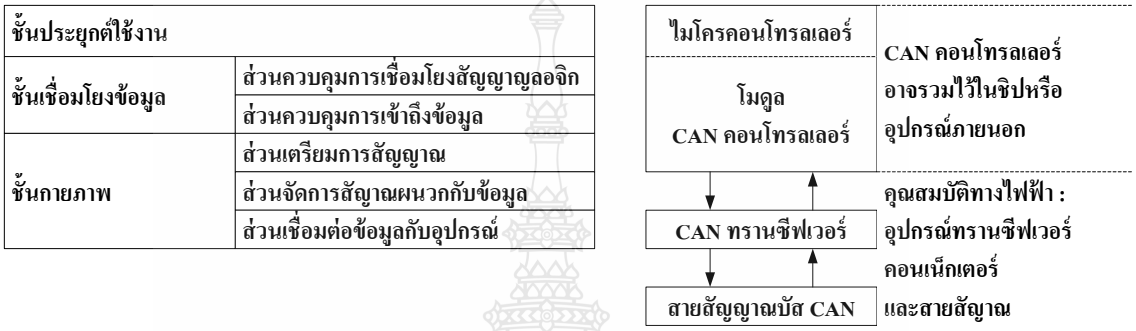


ภาพที่ 2.5 การแปลงสัญญาณ TTL เป็นสัญญาณข้อมูลบนบัสแคน ของแคนทรานซีฟเวอร์

2.1.5 สถาปัตยกรรมของอุปกรณ์บัสแคน

อุปกรณ์ที่นำมาเชื่อมต่อกับบัสแคน จะต้องมีสถาปัตกรรมที่เข้ากันได้ตามมาตรฐาน ISO 11898 ซึ่งว่าด้วยการถ่ายทอดข้อมูลระหว่างอุปกรณ์ในระบบโครงข่ายที่สอดคล้องตามโมเดล OSI (Open System Inter - Connection) โดยอุปกรณ์ระบบบัสแคนจะต้องทำงานให้สอดคล้องกับชั้นการทำงานชั้นล่างสุดของโมดูล OSI นั่นคือชั้นที่ 2 ชั้นเชื่อมโยงข้อมูล (Data Link Later - L2) เป็นชั้นการทำงานของการแก้ไขความผิดพลาดทั้งหมดที่อาจเกิดขึ้นระหว่างจุดเชื่อมต่อต่างๆ ของระบบ และชั้นที่ 1 ชั้นกายภาพ (Physical Layer -LI) เป็นชั้นการทำงานเพื่อกำหนดโหมดหรือลักษณะการถ่ายทอดข้อมูลระหว่างโครงข่ายที่เฉพาะเจาะจง

เมื่อนำมาใช้พัฒนาเป็นอุปกรณ์บัสแคน ชั้นเชื่อมโยงข้อมูลจึงได้รับการกำหนดหน้าที่ใหม่ เป็นชั้นการทำงานที่จัดการเกี่ยวกับโปรโตคอลที่ใช้ในการสื่อสารข้อมูล ทั้งยังรวมถึงการตรวจสอบความผิดพลาด ในการสื่อสารข้อมูลบนบัสแคนด้วย ส่วนชั้นกายภาพจะจัดการเกี่ยวกับฮาร์ดแวร์ทั้งหมด ซึ่งครอบคลุมถึงลักษณะคุณสมบัติของสัญญาณไฟฟ้าบนบัส อันประกอบด้วย ระดับสัญญาณและจังหวะเวลา ในภาพที่ 2.6 แสดง สถาปัตยกรรมของอุปกรณ์บัสแคนภายใต้มาตรฐาน IS011898



ภาพที่ 2.6 สถาปัตยกรรมของอุปกรณ์บัสแคนภายใต้มาตรฐาน ISO11898

2.1.6 มาตรฐานของบัส CAN

โปรโตคอลหรือรูปแบบการสื่อสารบนบัสแคนจัดว่าเป็นหนึ่งในโปรโตคอลแบบ CSMA/CD+AMP (Carrier-Sense Multiple-Access protocol with Collision Detection and Arbitration on Message Priority) โดย CSMA เป็นการกำหนดให้แต่ละโนดบนบัสต้องรอให้เกิดช่วงเวลาบัสว่างก่อนที่ส่งข้อมูลลงบนบัส ส่วน CD+AMP เป็นการแก้ไขปัญหาการชนกันของข้อมูลด้วยการตีความระดับนัยสำคัญของข้อความที่ส่งลงบนบัสนับถึงปัจจุบัน (ต้นปี ค.ศ. 2010) บัสแคนมีมาตรฐานรองรับ 3 เวอร์ชั่นในตารางที่ 2.1 ดังนี้

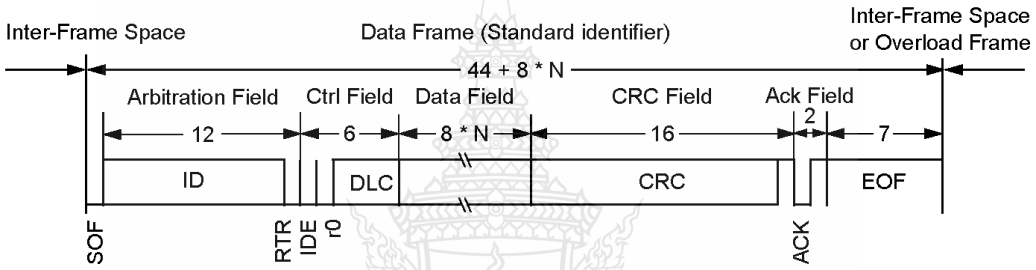
ตารางที่ 2.1 มาตรฐานของ CAN

ชื่อบัส	เลขมาตรฐาน	อัตราการถ่ายทอดข้อมูล / bps	จำนวนบิตระบุอุปกรณ์ / bit
Low-Speed CAN	ISO 11519	125 กิโลบิตต่อวินาที	11 บิต
CAN 2.0A	ISO 11898:1993	1 กิโลบิตต่อวินาที	11 บิต
CAN 2.0B	ISO 11898:1995	1 กิโลบิตต่อวินาที	29 บิต

สำหรับในปัจจุบันบัสแคนนิยมใช้ 2 มาตรฐานคือ CAN2.0A หรือบางครั้งเรียก บัสแคนมาตรฐาน (Standard CAN) และ CAN 2.0B หรือบัสแคนขยาย (Extended CAN) โดย CAN 2.0A มีตัวระบุอุปกรณ์หรือ Identifier ขนาด 11 บิต จึงสามารถรองรับข้อความจากอุปกรณ์ที่แตกต่างกันได้สูงสุด 2^{11} หรือ 2,048 อุปกรณ์ ในขณะที่ CAN 2.0B มีขนาด 29 บิต จึงสามารถรองรับข้อความจากอุปกรณ์ที่แตกต่างกันได้สูงสุดมากถึง 2^{29} หรือ กว่า 537 ล้านอุปกรณ์

2.1.7 รูปแบบข้อมูลสำหรับการระบุอุปกรณ์

1) รูปแบบข้อมูลสำหรับการระบุอุปกรณ์สำหรับ CAN2.0A หรือบัสแคนมาตรฐานจะแบ่งออกเป็น 8 ส่วน ดังนี้



ภาพที่ 2.7 รูปแบบข้อมูลสำหรับการระบุอุปกรณ์สำหรับบัสแคนมาตรฐาน

1.1) ส่วนเริ่มต้นเฟรม (1 บิต) ประกอบด้วย

- SOF - Start of Frame คือ บิตเริ่มต้นเฟรม ทำหน้าที่แจ้งการเริ่มต้นของข้อความ ใช้ในการซิงโครไนซ์โนดเมื่อต่อเข้ากับบัส

1.2) ส่วนตีความ (12 บิต) ประกอบด้วย

- Identifier คือ บิตระบุอุปกรณ์มีขนาด 11 บิต ใช้กำหนดนัยสำคัญของข้อความ ค่าของข้อมูลที่ต่ำ หมายถึงมีระดับนัยสำคัญที่สูงกว่า

- RTR - Remote Transmission Request Bit คือ บิตร้องขอการถ่ายทอดข้อมูล มีขนาด 1 บิต จะเป็น "0" เมื่อมีการร้องขอให้ถ่ายทอดข้อมูลจากโนดอื่นโดยปกติทุกโนดจะได้รับการร้องขอและบิตระบุอุปกรณ์จะเป็นตัวกำหนดโนดที่ต้องการติดต่อจากนั้นข้อมูลที่ร้องขอจะได้รับการถ่ายทอดลงบนบัส ทุกโนดจะได้รับข้อความนั้น หากโนดใดสนใจก็สามารถนำข้อมูลนั้นไปใช้ได้

1.3) ส่วนควบคุม (6 บิต) ประกอบด้วย

- IDE - Identifier Extension Bit คือ บิต ขยายข้อมูลระบุอุปกรณ์สำหรับ CAN 2.0A จะไม่ใช้งานบิตนี้ เนื่องจากต้องการบิตระบุอุปกรณ์เพียง 11 บิต กำหนดเป็น "0"

- r0 คือ สำรองไว้กำหนดเป็น "0"

- DLC - Data Length Code Bits คือ บิตกำหนดจำนวนของ ไบต์ข้อมูลที่ต้องการถ่ายทอดข้อมูล มีขนาด 4 บิต

1.4) ส่วนข้อมูล (64 บิต) ประกอบด้วย

- DATA คือ ข้อมูลที่ต้องการถ่ายทอด มีขนาดสูงสุด 8 ไบต์หรือ 64 บิต

1.5) ส่วนตรวจสอบความผิดพลาด (16 บิต) ประกอบด้วย

- CRC - Cyclic Redundancy Check Bits คือ บิตตรวจสอบความผิดพลาดของข้อมูลที่ทำการถ่ายทอด มีขนาด 15 บิต ใช้คำนวณหาค่ารหัสเพื่อตรวจสอบความผิดพลาดที่อาจเกิดขึ้น

- CRC Delimiter คือ บิตขอบเขตของรหัส CRC ปกติกำหนดเป็น "1"

1.6) ส่วนตอบรับ (2 บิต) ประกอบด้วย

- ACK - Acknowledge Bit คือ บิตตอบรับอุปกรณ์ในทุกโนดจะได้รับข้อความที่ถูกถ่ายทอดลงบนบัส โนดปลายทางที่ต้องการข้อความจะตอบสนองและแจ้งกลับไปยังโนดต้นทาง เพื่อยืนยันการติดต่อและถ่ายทอด ข้อมูลให้กันและกัน

- ACK Delimiter – Bit คือ บิตขอบเขตการตอบรับ ปกติกำหนดเป็น "1"

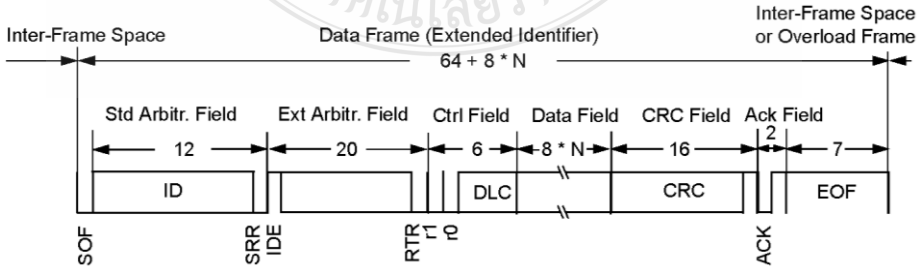
1.7) ส่วนสิ้นสุดเฟรม (7 บิต) ประกอบด้วย

- EOF-End of Frame Bits คือ บิตสิ้นสุดเฟรม มีขนาด 7 บิต กำหนดให้เป็น "1" ทั้งหมด

1.8) ส่วนช่องว่างเฟรม (3 บิต) ประกอบด้วย

- IFS - Inter-Frame Space Bits คือ บิตช่องว่างระหว่างเฟรม มีขนาด 3 บิต กำหนดเป็น "1" ทั้งหมด

2) รูปแบบข้อมูลสำหรับการระบุอุปกรณ์สำหรับ CAN2.0B หรือบัสแคนขยายแบ่งออกเป็น 8 ส่วน ดังนี้



ภาพที่ 2.8 รูปแบบข้อมูลสำหรับการระบุอุปกรณ์สำหรับบัสแคนแบบขยาย

1.1) ส่วนเริ่มต้นเฟรม (1 บิต) ประกอบด้วย

- SOF-Start of Frame คือ บิตเริ่มต้นเฟรม ทำหน้าที่แจ้งการเริ่มต้นของข้อความ ใช้ในการชิงโครไนซ์โนด เมื่อต่อเข้ากับบัส

1.2) ส่วนตีความ (12 บิต) ประกอบด้วย

- 11-Bit Identifier คือ บิตระบุอุปกรณ์ด้านสูง มีขนาด 11 บิต กำหนดเป็นบิต 28 ถึง 18 ใช้กำหนดนัยสำคัญของข้อความ ค่าของข้อมูลที่ต่ำหมายถึงมีระดับนัยสำคัญที่สูงกว่า

- SRR - Substitute Remote Request Bit คือ บิตร้องขอทดแทน มีขนาด 1 บิต ใช้แจ้งการร้องขอข้อมูล จากโนดที่มีข้อมูลที่ต้องการ สำหรับการระบุอุปกรณ์แบบ 29 บิต ต้องกำหนดบิตนี้เป็น "1"

- IDE - Identifier Extension Bit 8 คือ บิตขยายข้อมูลระบุอุปกรณ์ ต้องกำหนดเป็น "1" เพื่อให้ใช้งาน บิตระบุอุปกรณ์ขยายอีก 18 บิต

- 18-bit Identifier คือ บิตระบุอุปกรณ์ด้านต่ำ มีขนาด 18 บิต กำหนดเป็นบิต 17 ถึง 0 ใช้กำหนดนัยสำคัญของข้อความ โดยต้องนำไปรวมกับบิตระบุอุปกรณ์ด้านสูง 11 บิต รวมเป็น 29 บิต ค่าของข้อมูลที่ต่ำหมายถึงมีระดับนัยสำคัญที่สูงกว่า

- RTR - Remote Transmission Request bit คือ บิตร้องขอการถ่ายทอดข้อมูล มีขนาด 1 บิต จะเป็น "0" เมื่อมีการร้องขอให้ถ่ายทอดข้อมูลจากโนดอื่น

1.3) ส่วนควบคุม (6 บิต) ประกอบด้วย

- IDE - Identifier Extension Bit คือ บิต ขยายข้อมูลระบุอุปกรณ์สำหรับ CAN 2.0A จะไม่ใช้งานบิตนี้ เนื่องจากต้องการบิตระบุอุปกรณ์เพียง 11 บิต กำหนดเป็น "0"

- r0 คือ สำรองไว้กำหนดเป็น "0"

- DLC - Data Length Code Bits คือ บิตกำหนดจำนวนของ ไบต์ข้อมูลที่ต้องการถ่ายทอดข้อมูล มีขนาด 4 บิต

1.4) ส่วนข้อมูล (64 บิต) ประกอบด้วย

- DATA คือ ข้อมูลที่ต้องการถ่ายทอด มีขนาดสูงสุด 8 ไบต์หรือ 64 บิต

1.5) ส่วนตรวจสอบความผิดพลาด (16 บิต) ประกอบด้วย

- CRC - Cyclic Redundancy Check Bits คือ บิตตรวจสอบความผิดพลาดของข้อมูลที่ทำการถ่ายทอด มีขนาด 15 บิต ใช้คำนวณหาค่ารหัสเพื่อตรวจสอบความผิดพลาดที่อาจเกิดขึ้น

- CRC Delimiter คือ บิตขอบเขตของรหัส CRC ปกติกำหนดเป็น "1"

1.6) ส่วนตอบรับ (2 บิต) ประกอบด้วย

- ACK - Acknowledge Bit คือ บิตตอบรับ อุปกรณ์ในทุกโนดจะได้รับข้อความที่ถูกถ่ายทอดลงบนบัส โนดปลายทางที่ต้องการข้อความจะตอบสนองและแจ้งกลับไปยังโนดต้นทาง เพื่อยืนยันการติดต่อและถ่ายทอด ข้อมูลให้กันและกัน

- ACK Delimiter คือ บิตขอบเขตการตอบรับ ปกติกำหนดเป็น "1"

1.7) ส่วนสิ้นสุดเฟรม (7 บิต) ประกอบด้วย

- EOF-End of Frame Bits คือ บิตสิ้นสุดเฟรม มีขนาดความยาว 7 บิต กำหนดให้เป็น "1" ทั้งหมด

1.8) ส่วนช่องว่างเฟรม (3 บิต) ประกอบด้วย

- IFS - Inter-Frame Space Bits คือ บิตช่องว่างระหว่างเฟรม มีขนาด 3 บิต กำหนดเป็น "1" ทั้งหมด

2.1.4 หลักการทำงานโดยสรุปของบัสแคน

ในบัสแคนข้อความจะสามารถส่งออกมาจากทุกโนด และทุกโนดสามารถเข้าถึงเพื่อรับข้อมูลนั้นในบัสแคนไม่มีการกำหนดแอดเดรสทั้งโนดส่งและโนดรับ รายละเอียดของข้อความจะถูกกำหนดโดยส่วนระบุอุปกรณ์หรือ Identifier ที่มีลักษณะเฉพาะตัวในเครือข่ายทุกๆ โนดบนบัสจะรับข้อความและคำสั่งการทำงานจากนั้นแต่ละโนดจะตัดสินใจว่า จะรับข้อความนั้นไว้หรือไม่ โดยดูจากบิตระบุอุปกรณ์

ถ้ามีหลายๆ ข้อความได้รับการถ่ายทอดลงบนบัสพร้อมกัน ส่วนควบคุมในแต่ละโนดจะตัดสินใจรับข้อความโดยอาจพิจารณาจากนัยสำคัญที่มีการกำหนดมา และสามารถร้องขอการส่งข้อมูลซ้ำหากรับไม่ทันทำให้แน่ใจว่า จะไม่มีข้อความสูญหายขณะที่ถ่ายทอดเข้ามาบนบัสในเวลาเดียวกัน

2.2 ไมโครคอนโทรลเลอร์ แบบ ARM Cortex-M3

ARM Cortex-M3 [3-4] เป็นชื่อของโปรเซสเซอร์ 32 บิตที่พัฒนาภายใต้สถาปัตยกรรม ARMv7-M ซึ่งต่างจาก LPC2xxx ของ NXP ที่ใช้โปรเซสเซอร์ ARM7TDMI พัฒนาภายใต้สถาปัตยกรรม ARMv4T มีคุณสมบัติโดยสังเขบดังนี้

1) เป็นโปรเซสเซอร์แบบไฮเออราชิคอล (Hierarchical) ที่รวมเอาการทำงานซีพียูคอร์เข้ากับระบบเพอริเฟอรัลขั้นสูง

2) ซีพียูคอร์เป็นสถาปัตยกรรมแบบฮาร์วาร์ด (Harvard architecture) ไปป์ไลน์ 3 ระดับ พร้อมรองรับการกระโดดไปทำงานแบบทันทีทันใด รองรับคำสั่ง Thumb และ Thumb-2

3) หน่วยประมวลผลทางคณิตศาสตร์และลอจิก (ALU) มีโมดูลหารแบบฮาร์ดแวร์ (Hardware divider)

4) สามารถคูณเลข 16 และ 32 บิต ได้อย่างรวดเร็วเพียงหนึ่งไซเกิลของสัญญาณนาฬิกา

5) มีตัวควบคุมการอินเตอร์รัปต์แบบกำหนดค่าได้ รองรับการเกิดอินเตอร์รัปต์ได้ถึง 240 อินเตอร์รัปต์

6) มีตัวควบคุมเวกเตอร์อินเตอร์รัปต์ซ้อน (Nested Vectored Interrupt Controller : NVIC) สามารถกำหนดนัยสำคัญได้ถึง 256 ระดับ

7) ใช้ระบบบัสแบบเมตริกซ์

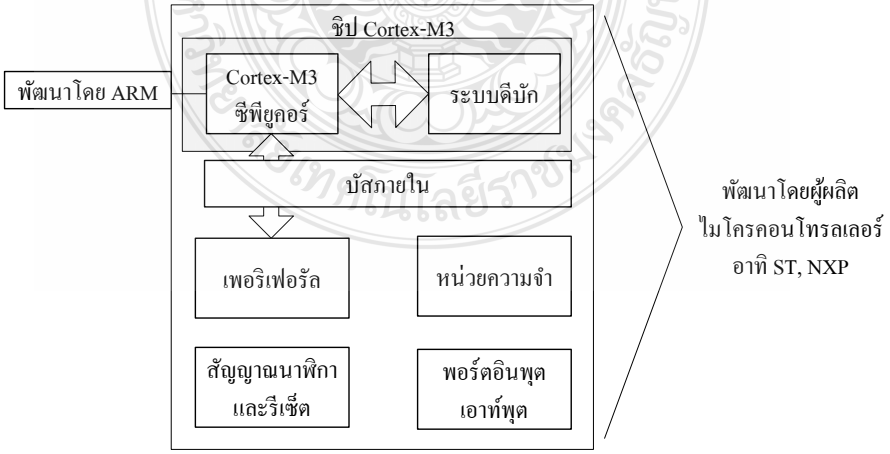
8) มีไทเมอร์ของระบบที่ชื่อ SysTick สำหรับรองรับการทำงานกับระบบปฏิบัติการเวลาจริง หรือ RTOS (Real-Time OS)

9) ใช้จำนวนขาในการดีบักทางฮาร์ดแวร์น้อย จากปรกติ 9 ขาเหลือเพียง 2 ถึง 3 ขา

10)ประสิทธิภาพความเร็วในการประมวลผล 1.2 DMIPS/MHz ซึ่งดีกว่า ARM7TDMI-S 70% เมื่อกระทำคำสั่ง Thumb โปรเซสเซอร์ และดีกว่า 35% เมื่อกระทำคำสั่ง ARM โปรเซสเซอร์

2.2.1 โปรเซสเซอร์ Cortex-M3 กับไมโครคอนโทรลเลอร์ Cortex-M3

โปรเซสเซอร์ Cortex-M3 เป็นซีพียูที่ได้รับการออกแบบและพัฒนาจาก ARM Limited ในประเทศอังกฤษผู้ผลิตชิปไมโครคอนโทรลเลอร์ได้ซื้อลิขสิทธิ์ในส่วนของคอร์ซีพียู Cortex-M3 มาพัฒนาต่อ โดยเพิ่มเติมหน่วยความจำ อุปกรณ์เพอริเฟอรัล ระบบสัญญาณนาฬิกา และโมดูลพอร์ตอินพุตเอาต์พุต ทำให้กลายเป็นไมโครคอนโทรลเลอร์ Cortex-M3 ดังแสดงด้วยบล็อกไดอะแกรมในภาพที่ 2.9



ภาพที่ 2.9 บล็อกไดอะแกรมไมโครคอนโทรลเลอร์แบบ Cortex-M3

2.2.2 พัฒนาการสถาปัตยกรรมของ ARM โปรเซสเซอร์

โปรเซสเซอร์ของ ARM ที่ประสบความสำเร็จอย่างมากในการเปิดตัวในอุปกรณ์สมองกลฝังตัวคือ ARM7TDMI ซึ่งเป็นหนึ่งในอนุกรมของ ARMv4T โปรเซสเซอร์ โดยตัวอักษร T ที่ต่อท้ายมีความหมายว่า ซีพียูเวอร์ชั่นหรือรุ่นนี้รองรับคำสั่ง Thumbโปรเซสเซอร์ด้วย

สถาปัตยกรรมรุ่นถัดมาที่ได้รับการพัฒนาขึ้นคือ ARMv5/v5E ซึ่งเป็นสถาปัตยกรรมที่เริ่มต้นบรรจุความสามารถในการประมวลผลสัญญาณดิจิตอลหรือ DSP (Digital Signal Processing) ทำให้ชิป ARM ได้ถูกนำไปพัฒนาเป็นตัวควบคุมหลักในอุปกรณ์มัลติมีเดีย

จากนั้น มีการเพิ่มเติมความสามารถในการจัดการระบบหน่วยความจำและคำสั่งจัดการข้อมูลจำนวนมากด้วยคำสั่งเพียงคำสั่งเดียวหรือ SIMD (Single Instruction-Multiple Data) เกิดเป็น ARMv6 ขึ้น ซึ่งได้รับการพัฒนามาเป็นไมโครคอนโทรลเลอร์ ARM11

กระทั่งปี 2006 ARM ได้พัฒนาคอร์ของซีพียู 32 บิตใหม่อีกครั้ง ด้วยการรองรับชุดคำสั่ง Thumb-2 ให้แก่ซีพียูคอร์ตัวใหม่ ทำให้สามารถบริหารจัดการคำสั่งได้ดีขึ้น ลดจำนวนอุปกรณ์ลง ลดพลังงานที่ใช้ลงแต่ความเร็วในการทำงานสูงขึ้น ทั้งยังมีการจัดกลุ่มซีพียูคอร์ใหม่เพื่อให้สามารถรองรับกับการประยุกต์ใช้งานที่ตรงจุดมากขึ้น ทั้งนี้เพื่อให้ ซีพียูที่มีความสามารถสูงในราคาที่ต่ำที่สุด นั่นคือที่มาของซีพียูคอร์รุ่นใหม่ ARMv7 โดยมีการแบ่งออกเป็น 3 แบบหลัก ๆ ดังนี้

1) แบบ A หรือ โปรไฟล์ A เป็นซีพียูที่เน้นให้ไปใช้เป็นแอปพลิเคชั่นโปรเซสเซอร์สำหรับที่มีความ ซับซ้อนสูงมาก เช่น ระบบปฏิบัติการ (Operating System : OS) เช่น Symbian, Linux หรือ Windows Embedded โดยซีพียูสำหรับงานลักษณะนี้ต้องมีความสามารถในการประมวลผลข้อมูลสูงสุด ระบบหน่วยความจำเสมือน หรือ Virtual Memory System ต้องสามารถรองรับหน่วยบริหารหน่วยความจำหรือ Memory Management Unit (MMU) ได้เป็นอย่างดี รองรับการทำงานกับภาษาจาวา ตัวอย่างของผลิตภัณฑ์ที่ใช้ซีพียูระดับนี้คือ โทรศัพท์เคลื่อนที่สมาร์ทโฟน เป็นต้น

2) แบบ R หรือโปรไฟล์ R เป็นซีพียูที่รองรับการทำงานแบบรีลไทม์ความเร็วสูง ซีพียูต้องมีเสถียรภาพและความน่าเชื่อถือในการทำงานในระดับสูง ตัวอย่างการพัฒนาชิปไปสู่การใช้งานจริงของซีพียูในรุ่นนี้คือ ชิปควบคุมฮาร์ดดิสก์ เนื่องจากฮาร์ดดิสก์ในปัจจุบันมีความจุสูง ดังนั้นการควบคุมระบบหยุดและเคลื่อนที่ของหัวอ่านต้องมีความเร็วและแม่นยำสูงเพื่อให้สามารถเข้าถึงข้อมูลได้อย่างถูกต้อง รวดเร็ว

3) แบบ M หรือ โปรไฟล์ M เป็นซีพียูที่ออกแบบมาเพื่อต้องการให้นำไปพัฒนาผลิตภัณฑ์ที่ต้องการความสามารถในการประมวลผลสูง งบประมาณต่ำ กินกำลังงานต่ำ มีวัตถุประสงค์ในแต่ละแอปพลิเคชั่นที่เจาะจงอย่างชัดเจน ซึ่งนั่นก็คือ วัตถุประสงค์ในการทำงานของไมโครคอนโทรลเลอร์

นั่นเอง ดังนั้นจากชื่อรุ่นของซีพียู ARMv7M ตัวอักษร M จึงอาจตีความได้ว่า ซีพียูนี้เหมาะสำหรับการนำไปพัฒนาเป็นชิปไมโครคอนโทรลเลอร์ความสามารถสูงต่อไป

Cortex คือ โปรเซสเซอร์รุ่นแรกที่พัฒนาขึ้นภายใต้สถาปัตยกรรม ArMv7-M และดังนั้น Cortex-M3 โปรเซสเซอร์จึงเป็นโปรเซสเซอร์ Cortex ในรุ่นที่ 3 และเป็นที่คาดหมายว่า จะเป็นมาตรฐานใหม่ของการพัฒนาชิปไมโครคอนโทรลเลอร์ 32 บิต ดังเช่นที่ ARM7TDMI เคยประสบความสำเร็จ

2.2.3 ชุดคำสั่ง Thumb-2 คือ จุดเปลี่ยนสำคัญของการพัฒนา Cortex-M3 โปรเซสเซอร์

ARM ได้พัฒนาชุดคำสั่ง Thum-2 ขึ้นมาตั้งแต่ปี 2003 ซึ่งเป็นชุดคำสั่งที่สามารถรองรับการทำงานกับคำสั่ง Thumb ทั้งแบบ 16 และ 32 บิต เนื่องจากปกติแล้วคำสั่งของ Thumb จะเป็นคำสั่ง 16 บิต จุดมุ่งหมายสำคัญของการพัฒนาชุดคำสั่งนี้คือ เป็นคำสั่งที่ใช้งานง่าย มีความต้องการหน่วยความจำน้อย และเป็นคำสั่งที่มีประสิทธิภาพสูง

เหตุผลหลักประการหนึ่งของการพัฒนาชุดคำสั่ง Thumb-2 คือ ต้องการให้ซีพียู ARMv7-M สามารถทำงานกับคำสั่ง 32 บิตได้ โดยใช้หน่วยความจำที่น้อยและคล่องตัวกว่า หากไม่มีการพัฒนาชุดคำสั่ง Thumb-2 การที่โปรเซสเซอร์ ARMv7-M จะทำงานกับคำสั่ง 32 บิตก็ต้องพัฒนาไปใช้คำสั่ง ARM ซึ่งทำให้ต้องการหน่วยความจำในการประมวลผลมาก ต้นทุนในการพัฒนาชิปสูง ทำให้ไม่ตอบโจทย์ที่ต้องการให้ ARMv7-M รองรับแอปพลิเคชั่นราคาประหยัดที่ต้องการประสิทธิภาพสูง

Cortex-M3 โปรเซสเซอร์จึงเป็นโปรเซสเซอร์ที่มีความพร้อมในการรองรับชุดคำสั่ง Thumb-2 ได้อย่างสมบูรณ์ นั่นจึงเป็นสาเหตุสำคัญที่ทำให้ Cortex-M3 โปรเซสเซอร์มีความแตกต่างจาก ARM โปรเซสเซอร์ดั้งเดิม เนื่องจากชุดคำสั่ง Thumb-2 รองรับการทำงานทั้งคำสั่ง 16 และ 32 บิต ทำให้ Cortex-M3 ไม่ต้องสลับการประมวลผลคำสั่งไปมาระหว่างชุดคำสั่ง Thumb มาตรฐาน (16 บิต) กับ ARM (32 บิต) ดังที่พบในรุ่น ARMv4T (ซึ่งก็คือ ARM7TDMI) ส่งผลให้การทำงานเร็วขึ้น มีค่าเวลาหน่วงสำหรับการประมวลผลลดลง สามารถสร้างโปรแกรมที่ซับซ้อนด้วยขนาดของหน่วยความจำที่น้อย

ตารางที่ 2.2 กระบวนการไปป์ไลน์ของ Cortex-M3 โปรเซสเซอร์

ชดคำสั่ง N	อ่านคำสั่ง	ถอดรหัส	กระทำคำสั่ง			
ชุดคำสั่ง N+1		อ่านคำสั่ง	ถอดรหัส	กระทำคำสั่ง		
ชุดคำสั่ง N+2			อ่านคำสั่ง	ถอดรหัส	กระทำคำสั่ง	
ชุดคำสั่ง N+3				อ่านคำสั่ง	ถอดรหัส	กระทำคำสั่ง

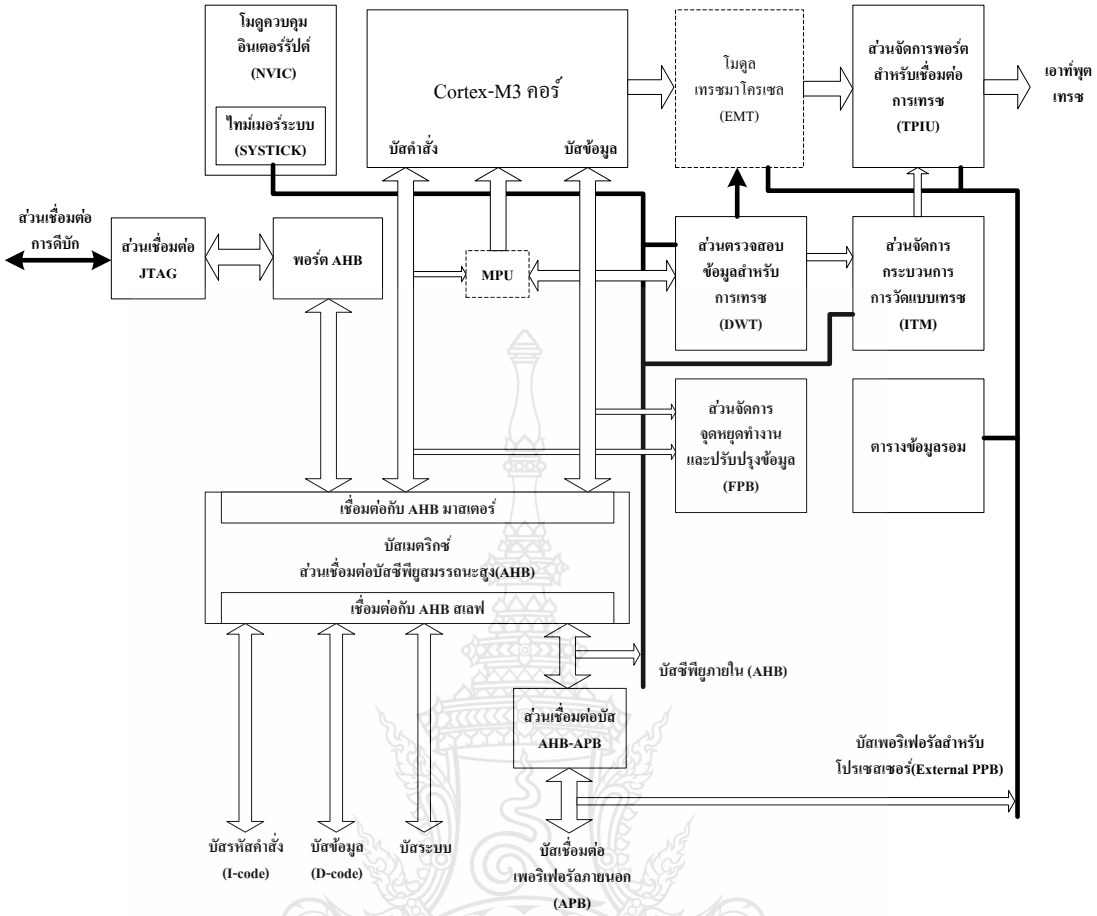
2.2.4 ไปป์ไลน์ของ Cortex-M3 โปรเซสเซอร์

Cortex-M3 โปรเซสเซอร์มีไปป์ไลน์ 3 ช่วง ประกอบด้วย ช่วงอ่านหรือเฟตซ์ (Fetch) คำสั่ง ช่วงถอดรหัสคำสั่ง (Decode) และช่วงเอ็กซิคิวต์ (Execute) หรือกระทำคำสั่ง ดังแสดงในตารางที่ 2.2 จะเห็นได้อย่างชัดเจนว่า ในขณะที่กำลังเอ็กซิคิวต์คำสั่งที่ 1 ซีพียูก็กำลังถอดรหัสคำสั่งที่ 2 พร้อมกันนั้นยังทำการเฟตช์คำสั่งที่ 3 ด้วยดังตารางที่ 2.2

หาก Cortex-M3 โปรเซสเซอร์ทำงานกับคำสั่ง 16 บิต ซีพียูไม่มีความจำเป็นต้องเฟตช์หรืออ่านคำสั่งทุกๆ ไซเกิล เนื่องจากซีพียูสามารถรองรับคำสั่งขนาด 32 บิตได้ ดังนั้นจึงสามารถอ่านคำสั่ง 16 บิต ได้ 2 คำสั่งในคราวเดียว โดยคำสั่ง 16 บิตอีกคำสั่งหนึ่งจะถูกพักไว้ในบัฟเฟอร์เพื่อรอการประมวลผลต่อไป ทำให้ซีพียูสามารถกระทำคำสั่งได้เร็วขึ้น

2.2.5 ไดอะแกรมการทำงานของ Cortex-M3 โปรเซสเซอร์ [1]

ในระบบของ Cortex-M3 โปรเซสเซอร์มิได้มีเพียงโปรเซสเซอร์ ARMv7-M เท่านั้น ยังประกอบด้วยอุปกรณ์ ที่ใช้ในการจัดการระบบจำนวนมากรวมถึงการรองรับกระบวนการดีบักด้วย ดังแสดงไดอะแกรมการทำงานในภาพที่ 2.10 ซึ่งสามารถแบ่งได้ 3 ส่วนหลักๆ คือ ส่วนของระบบโปรเซสเซอร์ ส่วนของระบบดีบัก และบัสเชื่อมต่อ ของระบบ ส่วนประกอบในระบบโปรเซสเซอร์ของ Cortex-M3 โปรเซสเซอร์มีดังนี้



ภาพที่ 2.10 ไดอะแกรมการทำงานของ Cortex-M3 โปรเซสเซอร์

Cortex-M3 คอร์ ประกอบด้วยรีจิสเตอร์ หน่วยคำนวณทางคณิตศาสตร์และลอจิก บัสข้อมูล บัสคำสั่ง และส่วนเชื่อมต่อระบบบัส

ส่วนควบคุมอินเตอร์รัปต์ (Nested Vectored Interrupt Controller: NVIC) เป็นส่วนจัดการอินเตอร์รัปต์ทั้งหมด สามารถเปลี่ยนแปลงได้ในไมโครคอนโทรลเลอร์จากแต่ละผู้ผลิต

ไทเมอร์ระบบ (SYSTICK) เป็นไทเมอร์นับถอยหลังใช้ในการสร้างสัญญาณอินเตอร์รัปต์ตามช่วงเวลาที่กำหนด มีลักษณะคล้ายกับการทำงานของวอตช์ด็อกไทเมอร์ในไมโครคอนโทรลเลอร์ 8 บิต โดยไทเมอร์ระบบ เป็นส่วนหนึ่งของ NVIC

ส่วนป้องกันหน่วยความจำ (Memory Protection Unit : MPU) ใช้ในการป้องกันข้อมูลในหน่วยความจำภายในที่สำคัญของซีพียู เป็นส่วนเสริมที่อาจไม่ได้มีในไมโครคอนโทรลเลอร์ Cortex-M3 ทุกรุ่น ทุกเบอร์

บัสเมตริกซ์ (Bus Matrix) เป็นหัวใจของระบบบัสภายในของ Cortex-M3 โปรเซสเซอร์ เนื่องจากเป็น ส่วนจัดการเชื่อมต่อทั้งหมดของบัสซีพียู (Advanced High-performance Bus : AHB) ในการถ่ายทอดข้อมูลจากส่วนต่างๆ ของโปรเซสเซอร์

ส่วนเชื่อมต่อบัสซีพียูกับบัสเพอริเฟอรัล (AHB-to-APB Bus Bridge) ใช้ในการเชื่อมต่ออุปกรณ์เพอริเฟอรัลเข้ากับบัสซีพียู

2.2.6 ส่วนประกอบในระบบดีบักและเทรซของ Cortex-M3 โปรเซสเซอร์

ส่วนเชื่อมต่อ JTAG เป็นส่วนจัดการสัญญาณเพื่อเชื่อมต่อกับอุปกรณ์ดีบักเกอร์ภายนอก ซึ่ง Cortex-M3 สามารถรองรับการดีบักได้หลายแบบ อาทิ Serial Wire Debug Port (SW-DP), Serial Wire JTAG Debug Port (SWJ-DP)

พอร์ต AHB Access Port (AHB) เป็นช่องทางที่ให้เข้าถึงหน่วยความจำของ Cortex-M3 พอร์ต นี้ได้รับการควบคุมจากส่วนเชื่อมต่อ JTAG ผ่านทางส่วนเชื่อมต่อการดีบักที่เรียกว่า DAP (Debug Access Port)

โมดูลเทรซมาโครเซล (Embedded Trace Macro Cell : ETM) เป็นส่วนแสดงข้อมูลของการเทรซ โดยข้อมูลของการเทรซจะถูกส่งออกไปทางพอร์ตเทรซผ่านส่วนเชื่อมต่อ TPIU ใช้ในการรองรับการเทรซแบบ รีลไทม์ เนื่องจากใน Cortex-M3 ในบางเบอร์นั้นไม่มีความสามารถในการเทรซแบบรีลไทม์ โมดูล ETM จะเข้ามา ช่วยจัดการในส่วนนี้แทน ดังนั้นโมดูล ETM จึงไม่มีในทุกรุ่นทุกเบอร์ของ Cortex-M3 รีจิสเตอร์ที่ใช้ควบคุม ETM จะถูกจัดสรรไว้ในระบบหน่วยความจำของ Cortex-M3 โปรเซสเซอร์และถูกควบคุมด้วยดีบักเกอร์ผ่านทางส่วนเชื่อมต่อการดีบักหรือ DAP

ส่วนตรวจสอบข้อมูลสำหรับการเทรซ (Data Watch Point and Trace : DWT) ใช้ในการตรวจสอบ ข้อมูลที่ปรากฏกับข้อมูลที่กำหนดไว้หากพบว่าตรงกันจะส่งสัญญาณไปกระตุ้นดีบักเกอร์เพื่อสร้างข้อมูลเทรซหรือ กระตุ้นการทำงานของโมดูล ETM

ส่วนจัดการกระบวนการวัดแบบเทรซ (Instrumentation Trace Marceau : ITM) เป็นส่วนประกอบ ที่สำคัญในกระบวนการเทรซของ Cortex-M3 โปรเซสเซอร์ ผู้พัฒนาสามารถเขียนโปรแกรมมายังโมดูลนี้โดยตรงเพื่อส่งข้อมูลเอาต์พุตออกไปยังส่วนเชื่อมต่อ TPIU หรือรับสัญญาณการตรวจสอบข้อมูลจากโมดูล DWT ก็ได้

ส่วนจัดการพอร์ตสำหรับเชื่อมต่อการเทรซ (Trace Port Interface Unit: TPIU) เป็นโมดูล ที่ใช้ในการเชื่อมต่อฮาร์ดแวร์สำหรับการเทรซภายนอก อาทิ เทรซพอร์อะนาไลเซอร์ โดยปกติแล้วข้อมูลของการเทรซภายในโปรเซสเซอร์จะมีรูปแบบเป็นแพ็กเก็ต Advanced Trace Bus (ATB) โมดูล TPIU จะทำการปรับรูปแบบข้อมูลให้สามารถติดต่อกับอุปกรณ์เทรซภายนอกได้

ส่วนจัดการจุดหยุดทำงานและปรับปรุงข้อมูล (Flash Patch and Breakpoint Unit: FPB) เป็นส่วนที่ทำงานเกี่ยวกับการกำหนดจุดหยุดทำงานในกระบวนการดีบักและเทรซ มีประโยชน์อย่างมากในการทดสอบและวิเคราะห์การทำงานของโปรเซสเซอร์

ตารางข้อมูลหน่วยความจำรอม (ROM Table) เป็นส่วนจัดการหน่วยความจำขนาดเล็กที่ใช้ในการ จัดสรรข้อมูลและเปลี่ยนแปลงแอดเดรสของการดีบัก โดยปกติแล้วการจัดสรรหน่วยความจำจะถูกกำหนดมาคงที่ตายตัว แต่สำหรับ Cortex-M3 โปรเซสเซอร์สามารถปรับเปลี่ยนตำแหน่งได้แต่ทั้งนี้ย่อมขึ้นอยู่กับซอฟต์แวร์ ที่ใช้ในการดีบักด้วย

2.2.7 บัสเชื่อมต่อของ Cortex-M3 โปรเซสเซอร์ประกอบด้วย

บัสรหัสคำสั่ง (I-Code Bus) เป็นบัส 32 บิต สำหรับเชื่อมต่อกับหน่วยความจำในช่วง 0x00000000 ถึง 0xlFFFFFFF เพื่อทำการอ่านคำสั่ง

บัสข้อมูล (D-Code Bus) เป็นบัส 32 บิต สำหรับเชื่อมต่อกับหน่วยความจำในช่วง 0x00000000 ถึง 0xlFFFFFFF เพื่อทำการอ่านข้อมูล

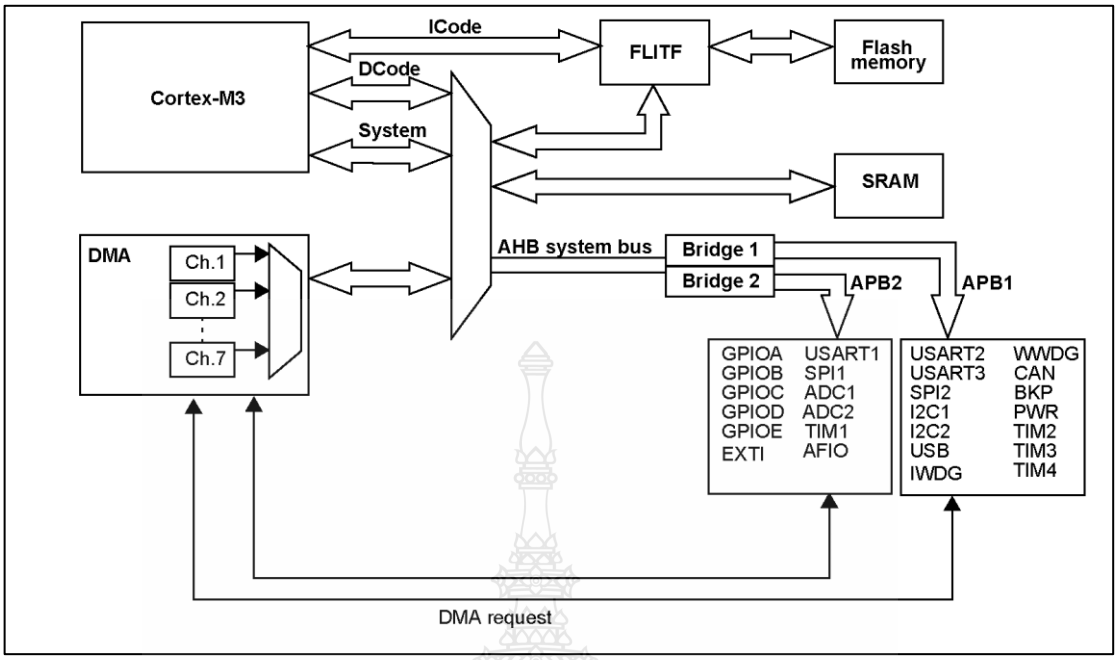
บัสระบบ (System Bus) เป็นบัส 32 บิต สำหรับเชื่อมต่อกับหน่วยความจำในช่วง 0x20000000 ถึง 0xDFFFFFFF และช่วง 0xE0l00000 ถึง 0xFFFFFFFF เพื่อทำการอ่านคำสั่งและข้อมูล

บัสเชื่อมต่ออุปกรณ์เพอริเฟอรัลภายนอก (External Private Peripheral Bus : External PPB) เป็นบัส 32 บิตที่ทำงานด้วยโปรโตคอลของบัส Advanced Peripheral Bus (APB) สำหรับเข้าถึงหน่วยความจำในช่วง 0xE0040000 ถึง 0xE00FFFFF นอกจากนั้นยังมีการใช้งานบัสนี้ร่วมกับโมดูล TPIU, ETM และตารางข้อมูล ROM นอกจากนั้นยังสามารถเพิ่มเติมอุปกรณ์พิเศษเข้าไปบนบัสนี้ได้ โดยกำหนดพื้นที่ไว้ในช่วงแอดเดรส 0xE0042000 ถึง 0xE00FF000 เท่านั้น

บัสดีบัก (Debug Access Port Bus) เป็นบัส 32 บิตที่ใช้เป็นเส้นทางเพื่อเข้าถึงพอร์ตดีบักของ Cortex- M3 โปรเซสเซอร์

2.2.8 ระบบไมโครคอนโทรลเลอร์ Cortex-M3 [2]

ระบบไมโครคอนโทรลเลอร์ที่พัฒนาต่อจาก Cortex-M3 โปรเซสเซอร์ ที่มีการใช้งานจริงโดยมีการต่ออุปกรณ์เพอริเฟอรัลจำนวนมากเข้ากับระบบบัสของ Cortex-M3 โปรเซสเซอร์ หน่วยความจำแฟลชและสแตติกแรมจะถูกต่อเข้ากับบัสซีพียู (AHB) และบัสระบบ ในขณะที่โมดูลเชื่อม ต่ออื่นๆ อาทิ พอร์ตอินพุตเอาต์พุต โมดูลสื่อสารข้อมูลอนุกรม USART หรือไทเมอร์จะถูกต่อเข้ากับบัสเพอริเฟอรัล (APB) และเชื่อมต่อเข้ากับบัสซีพียูผ่านทางวงจรเชื่อมต่อ AHB-APB กับระบบบัสซึ่งระบบไมโครคอนโทรลเลอร์สามารถเขียนไดอะแกรมได้ดังภาพที่ 2.11



ภาพที่ 2.11 ไดอะแกรมระบบไมโครคอนโทรลเลอร์

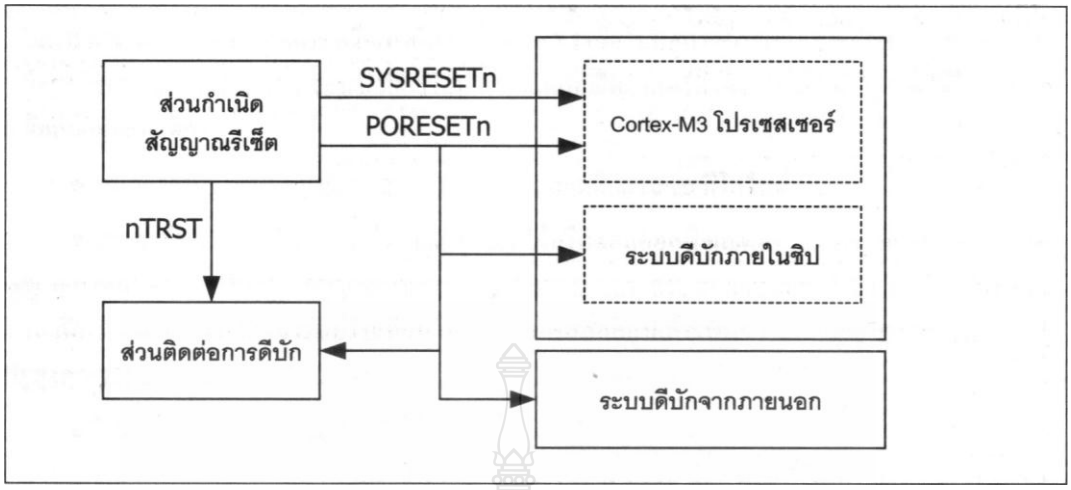
2.2.9 ระบบรีเซตในไมโครคอนโทรลเลอร์ Cortex-M3

ไมโครคอนโทรลเลอร์ Cortex-M3 มีการรีเซต 3 ชนิดหลักๆ ประกอบด้วย

1) การรีเซตจากเพาเวอร์ออนรีเซต (Power on Reset : PORESETn) เป็นการรีเซตที่เกิดขึ้นเมื่อมีการจ่ายไฟเลี้ยงเข้าสู่ระบบ การรีเซตจะเกิดขึ้นพร้อมกันทั้งโปรเซสเซอร์และระบบดีบัก

2) การรีเซตระบบ (System Reset : SYSRESETn) เป็นการรีเซตที่เกิดขึ้นเฉพาะในส่วนของโปรเซสเซอร์

3) การรีเซตทดสอบ (Test Reset : nTRST) เป็นการรีเซตที่เกิดขึ้นที่ระบบดีบักเท่านั้น ในภาพที่ 2.12 แสดงไดอะแกรมของการเกิดรีเซตในไมโครคอนโทรลเลอร์ Cortex-M3



ภาพที่ 2.12 ไดอะแกรมของการเกิดรีเซตในไมโครคอนโทรลเลอร์ Cortex-M3

2.2.10 คุณสมบัติทางเทคนิคที่สำคัญของ STM32F103 [5]

1) ซีพียูคอร์ ARM 32 บิต Cortex-M3 ความถี่สัญญาณนาฬิกาสูงสุด 72 MHz ซึ่งมีประสิทธิภาพของความ เร็วในการทำงาน 1.25 DMIPS/MHz หรือเท่ากับ Dhrystone 2.1 เมื่อไม่มีสภาวะการรอคอยเพื่อเข้าถึงหน่วยความจำ มีตัวหารเลขแบบฮาร์ดแวร์ และหน่วยความประมวลผลคณิตศาสตร์สามารถคูณเลข 32 บิตได้ในเวลา 1 ไซเกิลสัญญาณนาฬิกา

2) หน่วยความจำแบบแฟลช 128-512 กิโลไบต์ และสแตติกแรม 20-64 กิโลไบต์

3) ระบบสัญญาณนาฬิกา 4-16 MHz เมื่อใช้คริสตอลออสซิลเลเตอร์ 8 MHz เมื่อใช้วงจรกำเนิดสัญญาณนาฬิกา RC ภายในชิป มีสัญญาณนาฬิกาภายใน 40 kHz แบบ RC ออสซิลเลเตอร์ความถี่ 32 kHz สำหรับโมดูล เวลานาฬิกาจริงภายในชิป สามารถปรับเทียบได้ มีวงจรเฟสล็อกลูปเพื่อเพิ่มความถี่สัญญาณนาฬิกาหลักแก่ซีพียูสูงสุด 72 MHz

4) ไฟเลี้ยง 2.0-3.6 Vdc

5) ระบบจัดการพลังงาน มีเพาเวอร์-ออนรีเซ็ต (POR) เพาเวอร์-ดาวน์รีเซ็ต (PDR) มีวงจรตรวจจับไฟเลี้ยงแบบโปรแกรมได้ (Programmable Voltage Detector : PVD) รองรับการทำงานในโหมดประหยัดพลังงาน 3 โหมดคือ Sleep, Stop และ Standby

6) มีแหล่งจ่ายไฟ VBAT สำหรับสำหรับโมดูลเวลานาฬิกาจริงภายในชิปและรีจิสเตอร์รักษาค่าเวลา

7) โมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล ความเร็ว 1 ไมโครวินาที ความละเอียด 12 บิต จำนวน 2 ชุด รวม 16 ช่อง รับแรงดันได้ 0 ถึง 3.6Vdc มีวงจรสุ่มและพักข้อมูล (Sample and Hold) 2 ชุด และมีตัวตรวจ จับอุณภูมิอยู่ภายในชิป เพื่อตรวจสอบอุณหภูมิภายในชิปขณะทำงาน

8) DMA มีตัวควบคุมการเข้าถึงหน่วยความจำโดยตรงหรือ DMA Controller จำนวน 7 ช่อง รองรับการทำงาน ของไทเมอร์ โมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล โมดูลเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI โมดูลเชื่อมต่อ ระบบบัส I^2C และโมดูลสื่อสารข้อมูลอนุกรม USART

9) ขาพอร์ตอินพุตเอาต์พุตแบบความเร็วสูง 80 ขา รองรับระดับแรงดัน +5V ได้

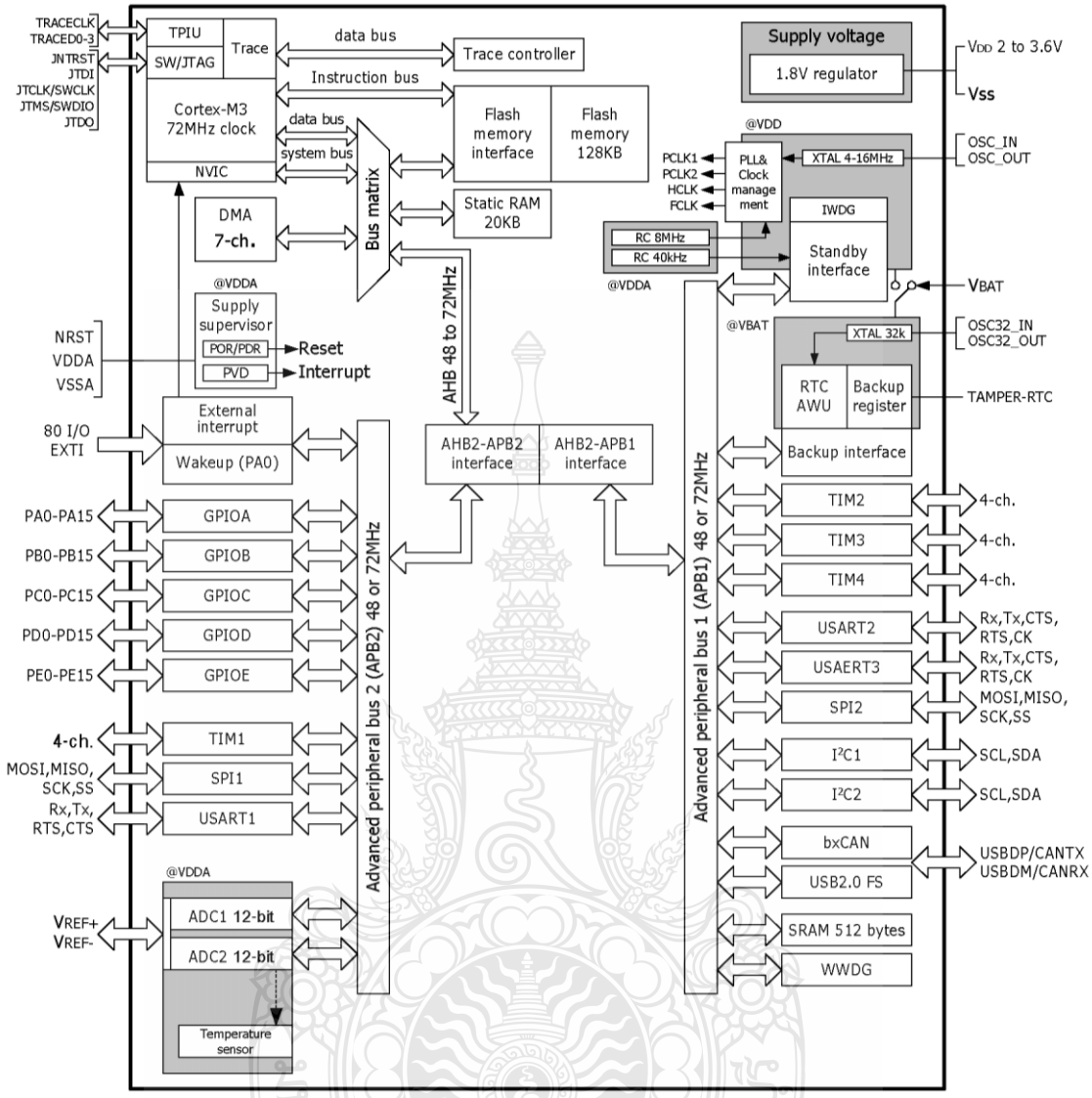
10) มีการรองรับการดีบักแบบอนุกรม (Serial wire debug : SWD) และ JTAG

11) ไทเมอร์ 7 ชุด

- ไทเมอร์ 16 บิตจำนวน 4 ชุด รองรับการทำงานโหมดตรวจจับสัญญาณอินพุต (Input Capture : IC) โหมด เอาต์พุตเปรียบเทียบ (Output Compare : OC) โหมดกำเนิดสัญญาณ PWM และตัวนับพัลส์ สำหรับไทเมอร์ TIM1 สามารถควบคุมโมดูลสัญญาณ PWM 6 ช่อง กำหนดค่าเวลาวิกฤตและการหยุดฉุกเฉินได้
- วอตช์ด็อกไทเมอร์ 2 ชุด แบบอิสระและใช้ฐานเวลาร่วมกับฐานเวลาหลัก
- ไทเมอร์หลักของระบบ (SysTick) เป็นตัวนับแบบถอยหลัง 24 บิต

12) โมดูลสื่อสารข้อมูล 9 แบบ

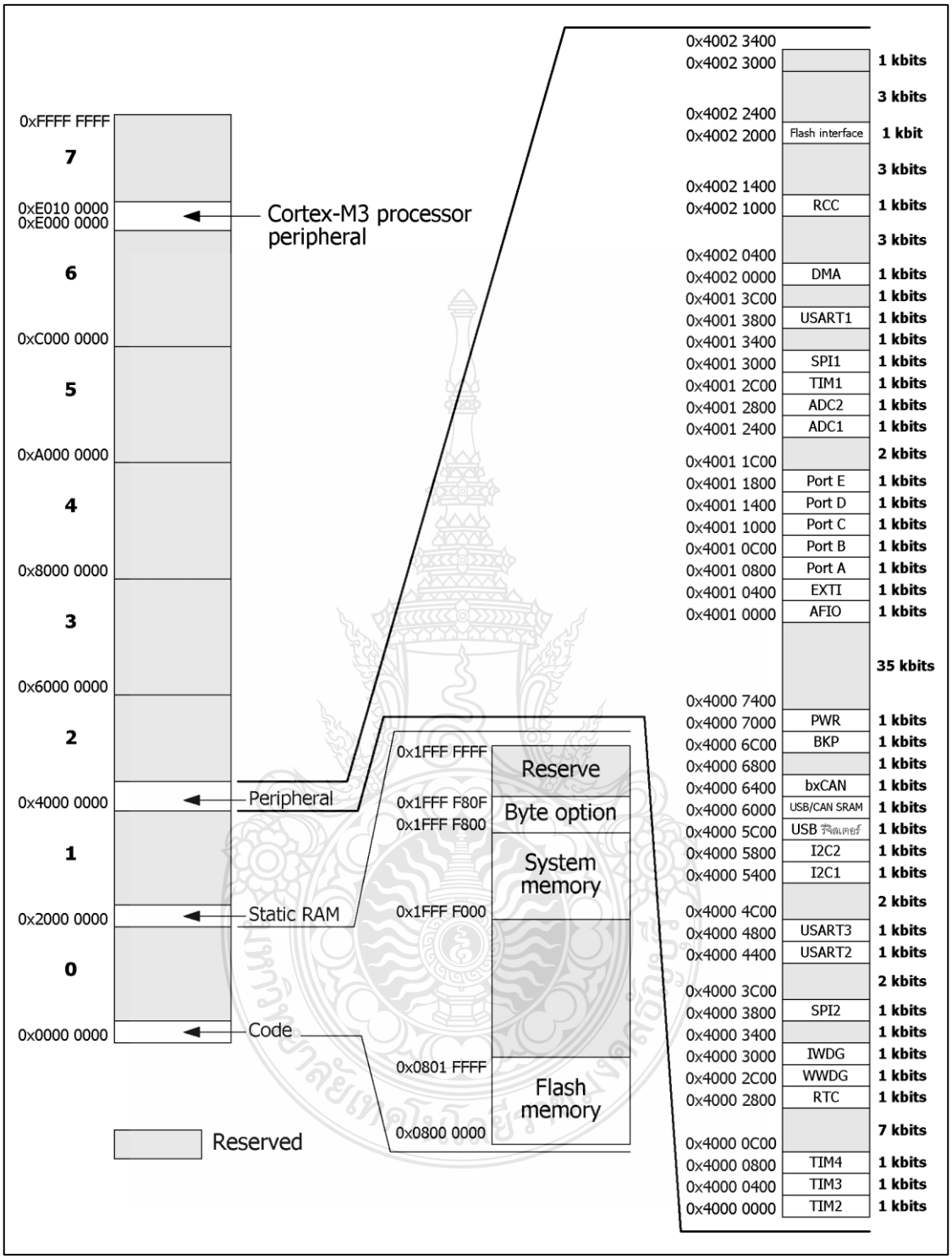
- โมดูลติดต่อระบบบัส I^2C 2 ชุด รองรับ SMBus และ PMBus
- โมดูลสื่อสารข้อมูลแบบอนุกรม รองรับการทำงาน IS07816, LIN และ IrDA
- โมดูลเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI 2ชุด ความเร็ว 18 เมกะบิตต่อวินาที
- โมดูลเชื่อมต่อบัสแคน แบบ 2.0B Active
- โมดูลเชื่อมต่อพอร์ต USB 2.0 แบบความเร็วเต็มที่



ภาพที่ 2.13 โครงสร้างและส่วนประกอบของไมโครคอนโทรลเลอร์ STM32F103

2.2.11 หน่วยความจำ

ใน STM32F103 มีหน่วยความจำแฟลชที่สามารถเขียนและลบได้ความจุตั้งแต่ 128 ถึง 512 กิโลไบต์สำหรับเก็บ ทั้งข้อมูลและโปรแกรมควบคุม มีหน่วยความจำสแตติกแรม 20 กิโลไบต์ ที่สามารถอ่านเขียนได้อย่างรวดเร็ว โดยไม่มีช่วงเวลารอคอย การจัดสรรหน่วยความจำแฟลชของไมโครคอนโทรลเลอร์ STM32F103 แสดงดังในภาพที่ 2.14



ภาพที่ 2.14 การจัดสรรหน่วยความจำแฟลชของไมโครคอนโทรลเลอร์ STM32F103

2.2.12 ตัวควบคุมเวกเตอร์อินเตอร์รัปต์ (NVIC)

ด้วยความสามารถของตัวควบคุมอินเตอร์รัปต์แบบนี้ ทำให้ STM32F103 สามารถรองรับการอินเตอร์รัปต์แบบกำหนดได้มากถึง 43 ช่อง ทั้งนี้ยังไม่รวมกับสายอินเตอร์รัปต์ภายนอกอีก 16 สายของ Cortex-M3 และกำหนดนัยสำคัญได้สูงถึง 16 ระดับ

2.2.13 ตัวควบคุมการอินเตอร์รัปต์จากภายนอกและการเกิดสัญญาณอีเวนต์ (EXTI)

ตัวควบคุมการอินเตอร์รัปต์จากภายนอกและการเกิดสัญญาณอีเวนต์ (External Interrupt/Event Controller) ประกอบด้วยตัวตรวจจับขอบสัญญาณ 16 ชุด เพื่อใช้ในการกำเนิดสัญญาณอินเตอร์รัปต์และสัญญาณอีเวนต์แล้วแต่การร้องขอแต่ละชุดจะทำงานเป็นอิสระต่อกัน สามารถกำหนดเงื่อนไขของการตรวจจับ ได้ทั้งแบบขอบขาลง ขอบขาขึ้น หรือทั้งสองขอบขา โดยสามารถกำหนดให้ตัวตรวจจับทั้ง 16 ชุดต่อกับขา พอร์ตอินพุตเอาต์พุตพอร์ตใดก็ได้ทั้ง 80 ขาของ STM32F103

2.2.14 สัญญาณนาฬิกา

ไมโครคอนโทรลเลอร์ STM32F103 สามารถรองรับสัญญาณนาฬิกาได้ทั้งจากวงจร RC ภายในและ ต่อคริสตอลภายนอก โดยมีวงจรเฟสล็อกลูปเพื่อคูณความถี่ของสัญญาณนาฬิกาได้สูงสุด 16 เท่า แต่ความถี่สัญญาณ นาฬิกาสูงสุดที่สามารถทำงานได้หลังผ่านวงจรเฟสล็อกลูปคือ 72MHz

2.2.15 บูตโหมด (Boot Mode)

เป็นกระบวนการเริ่มต้นการทำงานหรือเรียกว่า การบูต (Boot) ของไมโครคอนโทรลเลอร์ STM32 F103 หลังจากจ่ายไฟเลี้ยงแล้ว มีด้วยกัน 3 วิธีคือ

1) บูตจากหน่วยความจำแฟลช เลือกใช้ในกรณีที่ต้องรันโปรแกรมจากหน่วยความจำแฟลช

2) บูตจากหน่วยความจำของระบบ เลือกใช้ในกรณีที่ต้องการโปรแกรมหน่วยความจำแฟลชใหม่ด้วย บูตโหลดเดอร์ผ่านกระบวนการโปรแกรมในวงจรผ่านทางโมดูลสื่อสารข้อมูลอนุกรม USART 1

3) บูตจากหน่วยความจำสแตติกแรม เลือกใช้ในกรณีที่ต้องการรันโปรแกรมจากหน่วยความจำสแตติกแรมของไมโครคอนโทรลเลอร์ STM32F103

2.2.16 ไฟเลี้ยง (Supply Voltage)

STM32F103 ต้องการไฟเลี้ยงหลัก (V_{DD}) และไฟเลี้ยงส่วนอนาลอก (V_{DDA}) เป็นไฟตรงในช่วง 2 ถึง 3.6 V และไฟเลี้ยงสำหรับระบบสำรองข้อมูลและฐานเวลานาฬิกาจริง (V_{BAT}) ในช่วง 1.8 ถึง 3.6V จากแบตเตอรี่ ในกรณีที่ขาไฟเลี้ยงหลักไม่ได้ต่อกับไฟเลี้ยงวงจร

2.2.17 วงจรจัดการไฟเลี้ยง (Power Supply Supervisor)

STM32F103 มีวงจรเพาเวอร์ออนรีเซ็ต (POR) และเพาเวอร์ดาวน์รีเซ็ต (PDR) ซึ่งทำการแอกตีฟเพื่อทำให้เกิดการรีเซ็ตเมื่อไฟเลี้ยงเกิดการลดต่ำหรือเพิ่มขึ้นเป็น 2V จนกว่าระดับไฟเลี้ยงจะกลับเข้าสู่ระดับปกติคือ 2.6V ขึ้นไป STM32F103 ได้บรรจุตัวตรวจจับแรงดันแบบโปรแกรมได้ (Programmable Voltage Detector : PVD) เพื่อช่วยในการกำหนดเงื่อนไขการเกิดเพาเวอร์ออน และเพาเวอร์ดาวน์รีเซ็ต รวมถึงมีส่วนในการสร้างสัญญาณอินเตอร์รัปต์อันเนื่องมาจากแรงดันไฟเลี้ยงต่ำลงเกินกว่าที่กำหนด

2.2.18 วงจรควบคุมไฟเลี้ยง (Voltage Regulator)

วงจรควบคุมไฟเลี้ยงใน STM32F103 มีโหมดการทำงาน 3 โหมด คือ

1) MR โหมดทำงานปกติหรือโหมดรัน เป็นโหมดการทำงานตามปกติ ที่จะมีการจ่ายไฟเลี้ยง 1.8V ให้แก่วงจรภายในทั้งหมดของโปรเซสเซอร์

2) LPR โหมดพลังงานต่ำ จะเลือกใช้โหมดนี้เมื่อกำหนดให้ซีพียูทำงานในโหมดหยุดทำงาน (Stop mode) ในโหมดนี้วงจรควบคุมไฟเลี้ยงจะทำการจ่ายแรงดัน 1.8V ให้แก่รีจิสเตอร์และสแตติกแรมเท่านั้น

3) Power Down โหมดลดพลังงาน จะเลือกใช้โหมดนี้เมื่อกำหนดให้ซีพียูทำงานในโหมดรอคอย หรือสแตนด์บาย (Standby Mode) ในโหมดนี้วงจรควบคุมไฟเลี้ยงจะหยุดจ่ายแรงดันทั้งหมด ทำให้ข้อมูลใน รีจิสเตอร์และสแตติกแรมภายในตัวชิปสูญหายทั้งหมด ยกเว้นข้อมูลที่ต้องใช้ในการสแตนด์บายเพื่อรอการ เวกอัป รวมถึงข้อมูลในรีจิสเตอร์สำรองข้อมูลและรีลไทม์คล็อก

2.2.19 การทำงานในโหมดพลังงานต่ำ

STM32F103 มีโหมดพลังงานต่ำให้เลือกใช้ 3 โหมดคือ

1) โหมดสลีป (Sleep Mode) ในโหมดนี้ซีพียูจะหยุดทำงาน แต่อุปกรณ์เพอริเฟอรัลทั้งหมดยังทำงานอยู่ และสามารถเวกอัปซีพียูให้กลับมาทำงานได้ด้วยการอินเตอร์รัปต์หรือเกิดสัญญาณอีเวนต์

2) โหมดหยุดทำงาน (Stop Mode) ในโหมดนี้จะหยุดการทำงานของส่วนต่างๆ ในชิปเกือบทั้งหมด เพื่อให้เกิดการใช้พลังงานต่ำที่สุด โดยยังคงยอมให้หน่วยความจำทำงานเพื่อรักษาข้อมูลไว้ วงจรกำเนิดสัญญาณนาฬิกาทั้งหมดหยุดทำงาน เฟสล็อกลูปและออสซิเลเตอร์ถูกดิสเอเบิลหรือยกเลิกการทำงาน การเวกอัปจะเกิดขึ้น เมื่อเกิดการอินเตอร์รัปต์จากสัญญาณภายนอก (EXTI) ขึ้น เกิดภาวะเพาเวอร์ดาวน์รีเซ็ต เกิดการแจ้งปลุก (Alarm) จากรีลไทม์คล็อก หรือจากการเวกอัปของ USB

3) โหมดรอคอยหรือสแตนด์บาย (Standby Mode) การทำงานของไมโครคอนโทรเลอร์ในโหมดนี้ของ STM32F103 จะเป็นการทำงานที่ใช้พลังงานต่ำที่สุด เพราะแม้แต่หน่วยความจำก็จะถูกตัดไฟเลี้ยงด้วย วงจรกำเนิดสัญญาณนาฬิกา เฟสล็อกลูป และออสซิเลเตอร์จะหยุดทำงานลง (เนื่องจากไม่มีการจ่ายไฟเลี้ยง 1.8V ให้) จะเหลือเพียงวงจร ควบคุมการสแตนด์บาย รีลไทม์คล็อก และรีจิสเตอร์สำรองข้อมูลเท่านั้นที่ยังทำงานอยู่ (แต่ต้องมีการต่อไฟเลี้ยง เข้าที่ขา VBAT ด้วย) การเวกอัปเพื่อออกจากโหมดนี้จะเกิดขึ้นเมื่อมีการรีเซ็ตเกิดขึ้นที่ขารีเซ็ตหลัก (NRST) วอตชดอกอิสระ IWDG เกิดการรีเซ็ต เกิดสัญญาณพัลส์ขอบขาขึ้นที่ขา WKUP หรือ เกิดการแจ้งปลุก (Alarm) จากรีลไทม์คล็อก

2.2.20 รีลไทม์คล็อกและรีจิสเตอร์สำรองข้อมูล

1) STM32F103 มีวงจรฐานเวลานาฬิกาจริงหรือรีลไทม์คล็อกในตัว ทำให้สามารถสร้างระบบนาฬิกาที่มี ความแม่นยำสูงได้โดยไม่ต้องอาศัยอุปกรณ์ภายนอก เพียงต่อคริสตอล 32.768 kHz และป้อนแรงดัน ในย่าน 1.8V ถึง 3.6V เข้าที่ขา VBAT และเมื่อชิปทำงานในโหมดพลังงานต่ำ ระบบรีลไทม์คล็อกนี้ก็ยังทำงาน เพื่อรักษาค่าเวลาอยู่ จนกว่าจะมีการปลดแรงดัน VBAT ออกไป

2) รีจิสเตอร์สำรองข้อมูล (Backup register) เป็นรีจิสเตอร์ 16 บิตมีทั้งสิ้น 10 ตัว จะถูกใช้ในการเก็บข้อมูล ที่ผู้ใช้งานต้องการให้รักษาไว้ แม้ว่าจะไม่มีไฟเลี้ยงหลักจ่ายให้แก่ตัวไมโครคอนโทรลเลอร์

2.2.21 วอตช์ด็อกอิสระ (Independent Watchdog: IWDG)

เป็นตัวนับถอยหลัง (หรือนับลง) ขนาด 12 บิต และทำงานร่วมกับปรีสเกลเลอร์ 8 บิต โดยใช้สัญญาณ นาฬิกาความถี่ 40 kHz จากวงจร RC ภายในที่เป็นอิสระจากระบบสัญญาณนาฬิกาหลัก IWDG สามารถทำงานในโหมดหยุดทำงานและโหมดรอคอยได้ เพื่อใช้เป็นกลไกในการสร้างสัญญาณรีเซ็ตแก่ซีพียู เพื่อให้สามารถออกจากการทำงานในโหมดพลังงานต่ำได้ โดยเมื่อ IWDG นับค่าลงถึงศูนย์ หรือเกิดไทม์เอาต์ ก็จะทำให้เกิดสัญญาณรีเซ็ตขึ้น ผู้ใช้งานสามารถเข้าถึงเพื่อกำหนดค่าเวลาไทม์เอาต์ของ IWDG ได้ ตัวนับใน IWDG สามารถหยุดการทำงานได้หากชิปต้องทำงานในโหมดดีบัก

2.2.22 วินโดววอตช์ด็อก (Window Watchdog: WWDG)

เป็นตัวนับถอยหลังหรือนับลงขนาด 7 บิต ที่สามารถตั้งค่าได้ และเมื่อเกิดการไทม์เอาต์ก็จะทำให้เกิดสัญญาณรีเซ็ตซีพียูได้ ความแตกต่างของ WWDG กับ IWDG คือ WWDG ใช้สัญญาณนาฬิกาจากระบบสัญญาณนาฬิกาหลัก ดังนั้นหากซีพียูทำงานในโหมดพลังงานต่ำ WWDG จะหยุดทำงานไปด้วย ดังนั้นในการทำงานตามปกติ ผู้ใช้งานสามารถเลือกใช้งานวอตช์ด็อกได้ 2 ตัวคือทั้ง

IWDG และ WWDG โดยค่าเวลาไทม์เอาต์ของ WWDG จะน้อยกว่า IWDG มาก ดังนั้นหากเอ็นเนเบิลการทำงานของ WWDG โปรแกรมต้องไม่ลืมเข้ามาทำการเคลียร์ค่าตัวนับใน WWDG ให้ทันเวลาด้วย มิเช่นนั้นหาก WWDG นับค่าเวลาครบรอบ จะทำให้เกิดการรีเซ็ตขึ้น อาจส่งผลให้การทำงานไม่ต่อเนื่อง ตัวนับใน WWDG สามารถหยุดการทำงานได้หากชิปต้องทำงานในโหมดดีบัก

2.2.23 ไทเมอร์ระบบ

เป็นไทเมอร์หรือตัวตั้งเวลาหลักของระบบการทำงาน เป็นตัวนับถอยหลังหรือนับลง ขนาด 24 บิต ที่สามารถรีโหลดได้ สามารถสร้างสัญญาณอินเตอร์รัปต์ได้เมื่อค่าการนับเป็นศูนย์ และสามารถโปรแกรมเลือกแหล่งกำเนิดสัญญาณนาฬิกาได้ ตัวนับในไทเมอร์สามารถหยุดการทำงานได้หากชิปต้องทำงานในโหมดดีบัก

2.2.24 ไทเมอร์ใช้งานทั่วไป (TIMx)

ในไมโครคอนโทรลเลอร์ STM32F103 มีไทเมอร์ใช้งานทั่วไปขนาด 16 บิตให้เลือกใช้ 3 ตัว แต่ละตัวมีขาพอร์ตอินพุตเอาต์พุต 4 ช่อง ดังนั้นใน STM32F103 จึงมีขาพอร์ตที่สามารถทำงานกับไทเมอร์ได้มากถึง 12 ขาหรือช่อง สามารถกำหนดให้ทำงานร่วมกัน (คือนับค่าต่อกัน) หรือทำงานให้สัมพันธ์ หรือซิงโครไนซ์กันได้โดยไทเมอร์นี้สามารถกำหนดให้นับขึ้นหรือลงก็ได้สามารถป้อนค่าซ้ำได้มีปรีสเกลเลอร์ 16 บิต สามารถนำมาใช้ในการนับค่าในการทำงานตรวจจับสัญญาณ (Input Capture) หรือสร้างสัญญาณเอาต์พุตเปรียบเทียบ (Output Compare) รวมถึงนำไปใช้การสร้างสัญญาณ PWM ด้วย

นอกจากนั้นไทเมอร์แต่ละตัวยังสามารถทำงานร่วมกับโมดูล DMA ได้อย่างแยกอิสระต่อกัน และตัวนับในไทเมอร์สามารถหยุดการทำงานได้ หากชิปต้องทำงานในโหมดดีบัก

2.2.25 ไทเมอร์ควบคุมพิเศษ (TIM1)

สำหรับ TIM1 มีความพิเศษที่แตกต่างจากไทเมอร์ใช้งานทั่วไปตัวอื่นๆ ตรงที่สามารถกำหนดให้ทำงานเพื่อสร้างสัญญาณ PWM 3 เฟสได้ และทั้ง 4 ช่องสัญญาณของไทเมอร์ TIM1 นี้สามารถกำหนดโหมดทำงานได้ 5 โหมดแยกอิสระต่อกันอันประกอบด้วย

1) โหมดอินพุตตรวจจับสัญญาณ (Input Capture)
2) โหมดเอาต์พุตเปรียบเทียบ (Output Compare)
3) โหมดสร้างสัญญาณ PWM
4) โหมดสร้างสัญญาณพัลส์เดี่ยว (One Pulse Mode Output)
5) โหมดสร้างสัญญาณ PWM เสริม ที่สามารถโปรแกรมการเพิ่มค่าเวลาวิกฤตได้

นอกจากนั้นยังกำหนดให้ TIM1 ทำงานเป็นไทเมอร์แบบ 16 บิตมาตรฐานได้และสามารถทำงานร่วมกับไทเมอร์ TIMx (TIM2, TIM3 และ TIM4) ตัวอื่นๆภายในได้ โดยใช้คุณสมบัติเชื่อมโยงไทเมอร์ หรือ Timer Link

2.2.26 โมดูลเชื่อมต่อระบบบัส I^2C

STM32F103 มีโมดูลติดต่อระบบบัส I^2C จำนวน 2 ชุด สามารถกำหนดให้ทำงานได้ทั้งในแบบ มัลติ-มาสเตอร์หรือสเลฟ รองรับการทำงานแบบความเร็วมาตรฐาน และบัสความเร็วสูง สามารถติดต่อกับ อุปกรณ์บัส I^2C ทั้งแบบค่าแอดเดรส 7 บิตและแบบ 10 บิตได้ สามารถทำงานร่วมกับโมดูล DMA ได้

2.2.27 โมดูลสื่อสารข้อมูลพอร์ตอนุกรม (USART)

มีด้วยกัน 3 ชุด มี 1 ชุดสามารถรองรับอัตราเร็วในการถ่ายทอดข้อมูลสูงถึง 4.5 เมกะบิตต่อวินาที และที่เหลือสามารถกำหนดได้สูงถึง 2.5 เมกะบิตต่อวินาที มีขา RTS และ CTS ด้วย สามารถกำหนดให้ทำงานกับโมดูล IrDA เพื่อรับส่งข้อมูลอนุกรมผ่านแสงอินฟราเรดภายใต้มาตรฐาน IrDA SIR ENDEC และ USART 1 นั้นได้รับการกำหนดให้ใช้ในการโปรแกรมหน่วยความจำแฟลชของ STM32F103 ภายใต้การควบคุมของโหลดเดอร์ภายในไมโครคอนโทรลเลอร์ โมดูล USART ทั้ง 3 ชุด สามารถทำงานร่วมกับโมดูล DMA ได้

2.2.28 โมดูลติดต่ออุปกรณ์อนุกรม (Serial Peripheral Interface: SPI)

ใน STM32F103 มีโมดูล SPI จำนวน 2 ชุด สามารถรับส่งข้อมูลได้ด้วยอัตราเร็วสูงสุด 18 เมกะบิต ต่อวินาที สามารถกำหนดให้งานแบบ 2 ทิศทาง (ฟูลดูเพล็กซ์) หรือแบบทางเดียว (ซิมเพล็กซ์) ก็ได้ เลือกความถี่ของการทำงานในโหมดมาสเตอร์ได้ 8 ค่า สามารถกำหนดเฟรมของข้อมูลได้ตั้งแต่ 8 ถึง 16 บิต รองรับการติดต่อกับ SD/MMC การ์ด สามารถทำงานร่วมกับโมดูล DMA ได้

2.2.29 โมดูลติดต่อระบบบัสแคน

ใน STM32F103 มีโมดูลเชื่อมต่อบัสแคน ที่รองรับการทำงานของ CAN2.0 สามารถกำหนดอัตราเร็วในการถ่ายทอดบิตข้อมูลได้สูงสุด 1 เมกะบิตต่อวินาที สามารถรับและส่งเฟรมข้อมูลมาตรฐานร่วมกับตัวกำหนดรหัสเฉพาะ (Identifier) ขนาด 11 บิต และสามารถขยายไปได้ถึง 29 บิต โมดูลบัสแคนใน STM32F103 มีเมล์บ็อกซ์สำหรับตัวส่ง 3 ชุด ตัวรับข้อมูล FIFO แบบ 3 สเตจ 2 ชุด และมีตัวกรองข้อมูลแบบปรับขนาดได้ 14 ตัว

2.2.30 GPIO พอร์ตอินพุตเอาต์พุตใช้งานทั่วไป

STM32F103 มีขาพอร์ตใช้งานทั่วไปมาถึง 80 ขา โดยแบ่งเป็น 5 กลุ่ม กลุ่มละ 16 ขา คือ PORTA ถึง PORTE กำหนดเป็น PA0…PA15 ไปจนถึง PE0...PE15 สามารถกำหนดลักษณะการ

ทำงานเป็นขาอินพุตแบบ มีหรือไม่มีการต่อพูลอัปหรือพูลดาวน์ภายในขาเอาต์พุตแบบพุชพูลหรือแบบเดรนเปิดและขาพอร์ตฟังก์ชนิดพิเศษ อาทิ ขาอินพุตอนาลอกสำหรับโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล ขาสัญญาณของการเชื่อมต่อกับพอร์ต USB ขาเชื่อมต่อระบบบัส I^2C หรือ CAN หรือ SPI หรือ USART

2.2.31 โมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล

ใน STM32F103 มีโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล ความละเอียด 12 บิต จำนวน 2 ชุด รวม 16 ช่อง สามารถกำหนดให้ทำงานช่องเดียว (Single Shot) หรือสแกน (Scan Mode) ครบทั้ง 16 ช่อง สามารถ ทำงานร่วมกับโมดูล DMA ได้

2.2.32 โมดูล DMA

ความพิเศษของไมโครคอนโทรลเลอร์ Cortex-M3 คือ โมดูล DMA (Direct Memory Access) ใน STM32F103 มีโมดูล DMA 7 ช่อง ที่สามารถจัดการถ่ายทอดข้อมูลระหว่างหน่วยความจำต่อหน่วยความจำเพอริเฟอรัลต่อหน่วยความจำ และหน่วยความจำต่อเพอริเฟอรัลแต่ละช่องของ DMA สามารถต่อเข้ากับโมดูลฟังก์ชั่นพิเศษได้หลายตัว ประกอบด้วย SPI, I^2C, UART, ไทเมอร์ และโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล

2.2.33 โมดูลเชื่อมต่อบัส USB

รองรับการทำงานของพอร์ต USB 2.0 แบบความเร็วเต็มที่ มีอัตราการถ่ายทอดข้อมูลเท่ากับ 12 เมกะบิตต่อวินาที โดยทำงานเป็นอุปกรณ์สเลฟ

2.2.34 ตัวตรวจจับอุณหภูมิ

ตัวตรวจจับอุณหภูมิใน STM32F103 จะให้ผลการทำงานเป็นแรงดันไฟตรง มีย่านของแรงดันเอาต์พุตเท่ากับ 2 V ถึง 3.6 V ดังนั้นในการอ่านค่าจึงเหมือนกับการอ่านค่าอินพุต อนาลอกตามปกติ

2.2.35 พอร์ตดีบัก (SWJ-DP)

STM32F103 รองรับการดีบักทั้งจาก JTAG และแบบอนุกรมแต่จะต้องเลือกวิธีใดวิธีหนึ่งเท่านั้น เนื่องจากมีขาทำงานที่ทับซ้อนกัน

2.3 เครือข่ายไร้สายที่ใช้ในงานวิจัย

เครือข่ายไร้สาย (Wireless Local Area Network: WLAN) [6] ใช้การรับ-ส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์กับเครื่องคอมพิวเตอร์ หรือระหว่างเครื่องคอมพิวเตอร์กับระบบเครือข่ายอื่นๆ ผ่านทางอากาศ ทะลุกำแพง เพดานหรือสิ่งก่อสร้างอื่นๆ โดยปราศจากความต้องการของการเดินสาย

นอกจากนี้ระบบเครือข่ายไร้สายจะมีคุณสมบัติครอบคลุมทุกอย่างเหมือนกับระบบแบบใช้สาย และที่สำคัญคือการที่ไม่ต้องใช้สายทำให้การเคลื่อนย้ายการใช้งานทำได้โดยสะดวก ไม่เหมือนระบบแบบใช้สายที่ต้องใช้เวลาและการลงทุนในการปรับเปลี่ยนตำแหน่งการใช้งานหรือเพื่อเชื่อมต่อกับอุปกรณ์ในรูปแบบต่างๆ

2.3.1 ประโยชน์ของเครือข่ายไร้สาย

1) ระบบมีความคล่องตัวสูง ไม่ว่าจะเคลื่อนที่ไปที่ไหน หรือเคลื่อนย้ายคอมพิวเตอร์ไปตำแหน่งใด ก็ยังมีการเชื่อมต่อกับเครือข่ายตลอดเวลา ตราบใดที่ยังอยู่ในระยะการส่งข้อมูล

2) ระบบสามารถติดตั้งได้ง่ายและรวดเร็ว เพราะไม่ต้องเสียเวลาติดตั้งสายเคเบิล

3) ระบบสามารถขยายระบบเครือข่ายได้ง่าย เพราะเพียงแค่มี พีซีการ์ดมาต่อเข้ากับโน้ตบุ๊คหรือพีซี ก็ทำให้สามารถเข้าสู่เครือข่ายได้ทันที

4) ระบบลดค่าใช้จ่ายโดยรวมที่ผู้ลงทุนต้องลงทุน ซึ่งมีราคาสูงเพราะถ้าพิจารณาในระยะยาวระบบเครือข่ายไร้สายไม่จำเป็นต้องเสียค่าบำรุงรักษามาก

5) ระบบทำให้องค์กรสามารถปรับขนาดและความเหมาะสมได้ง่ายไม่ยุ่งยาก เพราะสามารถโยกย้ายตำแหน่งการใช้งาน โดยเฉพาะระบบที่มีการเชื่อมระหว่างจุดต่อจุด

2.3.2 รูปแบบการเชื่อมต่อของเครือข่ายไร้สาย

1) รูปแบบการเชื่อมต่อของระบบเครือข่ายไร้สายแบบ Peer to Peer เป็นลักษณะการเชื่อมต่อแบบโครงข่ายโดยตรงระหว่างเครื่องคอมพิวเตอร์ จำนวน 2 เครื่องหรือมากกว่า เป็นการใช้งานร่วมกันของ Wireless Adapter Card โดยไม่ได้มีการเชื่อมต่อกับเครือข่ายแบบใช้สาย

2) ระบบเครือข่ายไร้สายแบบ Client/Server เป็นลักษณะการรับส่งข้อมูลโดยอาศัยตัวกระจายสัญญาณ หน้าที่เป็นสะพานเชื่อมต่อระหว่างระบบเครือข่ายแบบใช้สายกับเครื่องคอมพิวเตอร์ลูกข่าย (Client) โดยจะกระจายสัญญาณคลื่นวิทยุเพื่อรับส่งข้อมูลเป็นรัศมีโดยรอบ เครื่องคอมพิวเตอร์ที่อยู่ในรัศมีของตัวกระจายสัญญาณจะกลายเป็นเครือข่ายกลุ่มเดียวกันทันที โดยเครื่องคอมพิวเตอร์ จะสามารถติดต่อกัน หรือติดต่อกับ Server เพื่อแลกเปลี่ยนและค้นหาข้อมูลได้ โดยต้องติดต่อผ่านตัวกระจายสัญญาณเท่านั้น

3) ระบบเครือข่ายแบบ Multiple Access Points and Roaming เป็นการเชื่อมต่อในสถานที่ที่ติดตั้งมีขนาดกว้างมากๆ โดยจะมีการเพิ่มจุดการติดตั้งตัวกระจายสัญญาณให้มากขึ้น เพื่อให้การรับส่งสัญญาณในบริเวณของเครือข่ายขนาดใหญ่เป็นไปอย่างครอบคลุมทั่วถึง

4) ระบบเครือข่ายแบบ Use of an Extension Point กรณีที่โครงสร้างของสถานที่ติดตั้งเครือข่ายแบบใช้สายมีปัญหา ผู้ออกแบบระบบอาจจะใช้ Extension Points ที่มีคุณสมบัติเหมือนกับตัวกระจายสัญญาณ แต่ไม่ต้องผูกติดไว้กับเครือข่ายไร้สายเป็นส่วนที่ใช้เพิ่มเติมในการรับส่งสัญญาณ

5) ระบบเครือข่ายแบบ The Use of Directional Antennas ใช้เสาอากาศในการรับส่งสัญญาณระหว่างจุดที่อยู่ห่างกันโดยการติดตั้งเสาอากาศที่แต่ละจุด เพื่อส่งและรับสัญญาณระหว่างกัน

2.3.3 ช่องความถี่ในระบบเครือข่ายแบบไร้สาย

ช่องความถี่ไร้สายถูกกำหนดให้มี 14 ช่อง ตามมาตรฐาน IEEE 802.11g 2.4 GHz แต่ละช่องมีความถี่ต่างกัน โดยมีช่องความถี่ต่าง ๆ ที่แสดงในตารางที่ 2.3 ดังนี้

ตารางที่ 2.3 การเปรียบเทียบความถี่ที่ใช้ในช่องสัญญาณต่างๆ ตามมาตรฐาน IEEE 802.11g ที่ความถี่ 2.4 GHz

Chanel Identifier	**Center Frequency (MHz)**	**Regulatory Domains**							
		Americas(-A)		**EMEA(-E)**		**Israel(-I)**		**Japan(-J)**	
		CCK	**OFDM**	**CCK**	**OFDM**	**CCK**	**OFDM**	**CCK**	**OFDM**
1	2412	X	X	X	X	-	-	X	X
2	2417	X	X	X	X	-	-	X	X
3	2422	X	X	X	X	-	-	X	X
4	2427	X	X	X	X	-	-	X	X
5	2432	X	X	X	X	X	X	X	X
6	2437	X	X	X	X	X	X	X	X
7	2442	X	X	X	X	X	X	X	X
8	2447	X	X	X	X	X	X	X	X
9	2452	X	X	X	X	X	-	X	X
10	2457	X	X	X	X	X	-	X	X
11	2462	X	X	X	X	X	-	X	X
12	2467	-	-	X	X	X	-	-	-
13	2472	-	-	X	X	X	-	-	-
14	2484	-	-	-	-	-	-	-	-

2.3.4 คลื่นความถี่วิทยุ 2.4GHz ที่ใช้ในงานวิจัย

RF 2.4 GHz เครือข่ายไร้สาย เป็นระบบการสื่อสารข้อมูลโดยใช้การส่งคลื่นความถี่วิทยุในย่าน RF และคลื่นอินฟาเรด ช่องความถี่ไร้สายถูกกำหนดให้มี 14 ช่อง ตามมาตรฐาน IEEE 802.11g 2.4 GHz โดยย่านความถี่ที่ศึกษานั้นใช้ย่านความถี่ 2400-2483.5 MHz อยู่ในช่วงการทำงานของ ISM/SRD และใช้รูปแบบในการมอดูเลตแบบ Gaussian Frequency Shift Keying (GFSK) ซึ่งขนาดของคลื่นพาห์จะไม่เปลี่ยนแปลง ลักษณะของสัญญาณมอดูเลตนั้น เมื่อค่าของบิตของสัญญาณข้อมูลดิจิตอลมีค่าเป็น 1 ความถี่ของคลื่นพาห์จะสูงกว่าปกติ และเมื่อบิตมีค่าเป็น 0 ความถี่ของคลื่นพาห์ก็จะต่ำกว่าปกติ

2.4 งานวิจัยที่เกี่ยวข้อง

2.4.1 จากงานวิจัยของ Harish Chincholi [7] ได้นำเสนอ “Wireless Controller Area Network Based Cross Channel Data Link” เป็นแนวคิดในการออกแบบโดยใช้ไมโครคอนโทรลเลอร์แบบ 16 บิต และโมดูล RF ที่ทำงานที่ย่านความถี่ 433MHz. โดยแยกเป็นอุปกรณ์ตัวรับและตัวส่ง โดยนำไมโครคอนโทรลเลอร์แบบ 16 บิต มาใช้ติดต่อกับชิปแคน เพื่อรับข้อมูลส่งให้กับภาครับและแสดงผลข้อมูลที่ได้รับออกจอ LDC อย่างไรก็ตามในงานวิจัยนี้ได้นำเสนอรูปแบบการออกแบบเท่านั้น ไม่ได้นำมาทดสอบเพื่อหาผลลัพธ์ว่า สามารถทดแทนการทำงานของระบบแคนได้หรือไม่ที่อัตราการถ่ายทอดข้อมูลของแคนเท่ากับ 1 เมกะบิตต่อวินาที และความสามารถของโมดูลไร้สายที่ใช้นั้นสามารถรับส่งข้อมูลได้ที่ 500 กิโลบิตต่อวินาทีเท่านั้น

2.4.2 ในงานวิจัยของ Lin Hongju, Wang Haifang, Xiao Nianxin, Liu Chunxia, Chen Panfeng [8] ได้นำเสนอ “Research on Coal Mine Personnel Positioning System Based on Zigbee and CAN” โดยใช้ ไมโครคอนโทรลเลอร์และ Zigbee ในการติดต่อสื่อสารข้อมูลแคน มีผลลัพธ์เพื่อใช้ในการตรวจสอบข้อมูลแคน ในระบบ โดยใช้ Zigbee เป็นตัวส่งข้อมูลออกมาให้กับตัวรับเพื่อความสะดวกในการเก็บข้อมูล โดยที่ไม่ต้องนำสายสัญญาณไปเชื่อมต่อโดยตรง ข้อมูลที่ได้รับนั้นจะเป็นค่าของข้อมูลที่อยู่ในชุดข้อมูลแคนเท่านั้น ไม่รวมข้อมูลส่วนระบุอุปกรณ์อื่นๆ ซึ่งงานวิจัยนี้มีจุดมุ่งหมายในการเก็บข้อมูลโดยใช้ Zigbee เป็นสื่อกลางในการสื่อสารเท่านั้น

2.4.3 ในงานวิจัยของ Mathias Johanson, Lennart Karlsson, Tore Risch [9] ได้นำเสนอ “Relaying Controller Area Network Frames over Wireless Internetworks for Automotive Testing Applications” โดยใช้อุปกรณ์ Wireless internet มาเชื่อมต่อกับระบบแคน เพื่อใช้ในการสื่อสารกับระบบ โดยผลที่ได้นั้นสามารถติดต่อกับระบบแคนโดยใช้อินเตอร์เน็ตได้ โดยมีแนวทางในการ

นำมาใช้งานเพื่อใช้ติดต่อกับระบบแคน โดยใช้ Wireless เป็นสื่อกลางในการเชื่อมต่อ แต่ในการนำมาใช้งานกับระบบจำเป็นที่จะต้องมีอุปกรณ์คอมพิวเตอร์และ Wireless ในการสื่อสารข้อมูล

2.4.4 ในงานวิจัยของ Luiz Alberto Castro de Almeida and Carlos Alberto dos Reis Filho [10] ได้นำเสนอ“Latency Evaluation in A Bluetooth-CAN Dual Media Sensor Network” โดยใช้ Bluetooth เป็นแนวคิดในการออกแบบและพัฒนาระบบ ผลลัพธ์ที่ได้นั้น สามารถสรุปได้ว่าสามารถพัฒนาการสื่อสารข้อมูลในรูปแบบไร้สายได้ โดยใช้โมดูล Bluetooth ได้

2.4.5 จากงานวิจัยของ “Ai Chunli, Zhang Fengdeng,Liu Rongpeng [11] ได้นำเสนอแนวทางและรูปแบบการพัฒนาการสื่อสารในรูปแบบแคน ให้อยู่ในรูปแบบไร้สาย ซึ่งเป็นการจำลองการทดสอบเพื่อหาผลลัพธ์หรือวิธีการที่สามารถพัฒนารูปแบบการสื่อสารนี้ ให้ตรงตามวัตถุประสงค์เพื่อหาวิธีการเท่านั้น

จากงานวิจัยที่กล่างมาข้างต้น มีจุดที่สามารถพัฒนาได้โดยเพิ่มความสามารถในการประมวลผลของไมโครคอนโทรลเลอร์จากเดิมที่มีขนาด 16 บิต ซึ่งมีความเร็วในการประมวลผลน้อยและไม่สามารถตอบสนองการรับ-ส่งข้อมูลที่มีปริมาณมากได้ แต่การพัฒนางานวิจัยโดยใช้ไมโครคอนโทรลเลอร์ขนาด 16 บิต นั้นสามารถทำการรับส่งข้อมูลแคนได้ที่ 1 เมกะบิตต่อวินาที เช่นกันแต่ค่าความผิดพลาดและค่าหน่วงเวลาอาจจะมากขึ้น และถ้านำมาประยุกต์ใช้งานแบบมัลติโนดหรือแบบหลายโนดร่วมกัน ก็อาจจะเกิดข้อผิดพลาดและเวลาที่สูญเสียมากขึ้นได้ ดังนั้นในงานวิจัยนี้จึงมีแนวคิดในการพัฒนาความสามารถของไมโครคอนโทรลเลอร์จากเดิมที่มีขนาด 16 บิต เป็นไมโครคอนโทรลเลอร์แบบ ARM Cortex-M3 ขนาด 32 บิต ที่มีความเร็วในการประมวลผลคำสั่งมากกว่าเดิมและขนาดของหน่วยความจำมากขึ้น เพื่อเพิ่มศักยภาพในการประมวลผล รวมทั้งรองรับการพัฒนาซอฟแวร์ให้มีความสามารถในการลดความผิดพลาดในการรับ-ส่งข้อมูลระหว่างกัน พัฒนาความสามารถในการรับส่งข้อมูลแคน ให้สามารถรับ-ส่งข้อมูลได้ที่ 1 เมกะบิตต่อวินาที ซึ่งเป็นอัตราการถ่ายทอดข้อมูลที่แคน สามารถทำงานได้สูงสุด โดยปรับปรุงอุปกรณ์และพัฒนารูปแบบการทำงานจากโมดูลไร้สายที่แยกภาครับและภาคส่งออกจากกัน เป็นโมดูลไร้สายที่สามารถรับ-ส่งข้อมูลได้ภายในตัวเดียวกันทำให้ลดขนาดวงจรรวมให้มีขนาดที่เล็กลง รวมทั้งพัฒนาเพื่อรองรับการทำงานแบบมัลติโนด และสามารถนำไปใช้ทดแทนการเชื่อมต่อแบบเดิมบนบัสการทำงานได้ ไม่ว่าจะเป็นการเชื่อมต่อที่จุดใด เนื่องจากไม่ต้องทำการปรับปรุงระบบ และลดความซับซ้อนในการนำมาใช้งาน

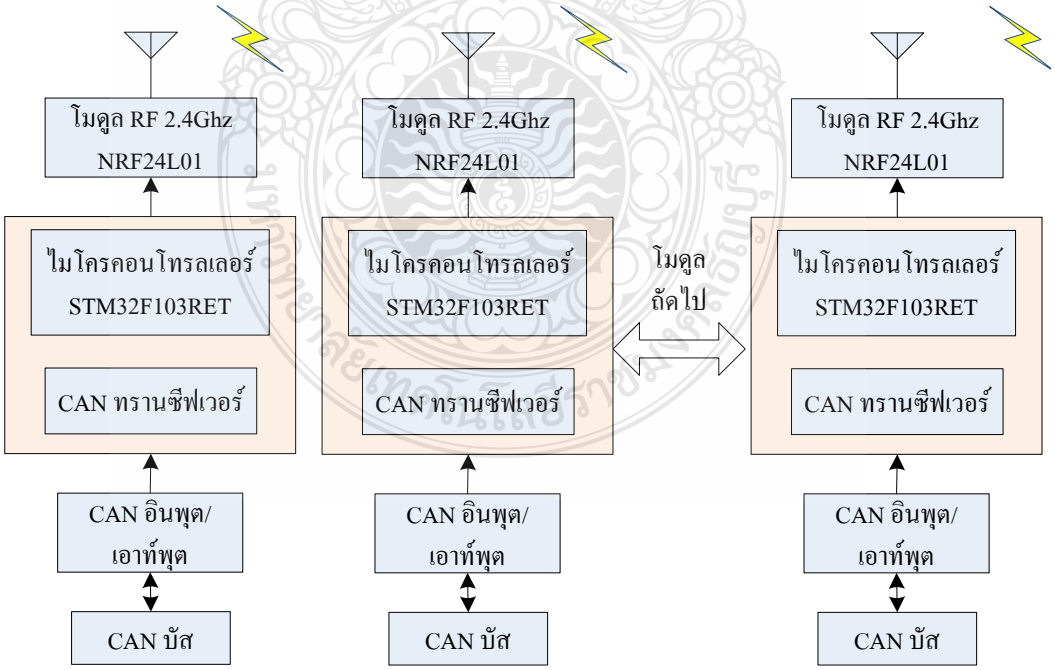
ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงมีแนวคิดที่จะพัฒนาและประยุกต์ใช้ไมโครคอนโทรลเลอร์ ARM Cortex-M3 สำหรับระบบแคนไร้สาย เพื่อพัฒนาความสามารถของระบบให้มีประสิทธิภาพมากขึ้นและสามารถนำมาประยุกต์ใช้งานตามวัตถุประสงค์ได้

บทที่ 3
วิธีดำเนินการวิจัย

3.1 บทนำ

จากข้อมูลที่ศึกษาไมโครคอนโทรลเลอร์ STM32F103RET ที่มีขนาด 32 บิต นั้นมีความเร็วในการประมวลผลมากกว่าไมโครคอนโทรลเลอร์ขนาด 16 บิต และสามารถออกแบบโปรแกรมให้นำข้อมูลที่รับส่งเข้าถึงรีจิสเตอร์ได้โดยตรง เพื่อรับ-ส่งข้อมูลกับโมดูล RF 2.4GHz เบอร์ NRF24L01 ซึ่งเป็นโมดูลสำเร็จรูปที่สามารถรับ-ส่งข้อมูลได้ภายในโมดูล นำมาใช้งานร่วมกับไมโครคอนโทรลเลอร์ โดยทำการออกแบบและประกอบวงจรเพื่อใช้การทดสอบจำนวน 4 ชุด เพื่อทดสอบการรับ-ส่งข้อมูลระหว่างโมดูลแต่ละตัวที่ระยะทางต่างกันว่าสามารถทำการรับ-ส่งข้อมูลได้สมบูรณ์ตามที่กำหนดไว้หรือไม่ ซึ่งในขั้นตอนการทดสอบจะใช้อุปกรณ์ทดสอบสัญญาณแคน หรือ CAN USB มาใช้ในการทดสอบส่งสัญญาณแคน ให้กับโมดูลและรับข้อมูลแคน กลับไปแสดงผลในโปรแกรม

3.2 หลักการทำงานของระบบ



ภาพที่ 3.1 หลักการทำงานของระบบ

จากภาพที่ 3.1 หลักการทำงานโดยรวมนั้นเมื่อนำอุปกรณ์ทดสอบสัญญาณแคน มาทำการเชื่อมต่อกับวงจรและส่งข้อมูลให้กับโมดูลชุดแรก ข้อมูลแคนที่ได้รับจะถูกเปลี่ยนแปลงให้อยู่ในรูปแบบที่ไมโครคอนโทรลเลอร์ สามารถนำมาประมวลผลได้ โดยใช้ชิปแคนทรานซีฟเวอร์ เบอร์ SN65HVD230 ของ TI ทำหน้าที่ในการแปลงสัญญาณ CAN High และ CAN Low ให้มาอยู่ในรูปแบบ CAN RX และ CAN TX เพื่อส่งข้อมูลแคน ให้กับไมโครคอนโทรลเลอร์

หลังจากที่ไมโครคอนโทรลเลอร์ ได้รับข้อมูลมาแล้วก็จะประมวลผลข้อมูลแคน ที่ได้รับเข้ามาให้อยู่ในรูปแบบที่สามารถติดต่อกับโมดูล RF โดยนำข้อมูลแคน และข้อมูลตรวจสอบความถูกต้องที่ได้จากการประมวลผล นำมารวมไว้เป็นหนึ่งแพ็คเก็คข้อมูล เพื่อนำแพ็คเก็คข้อมูลที่ประมวลผลแล้วส่งให้กับโมดูล RF ด้วยการสื่อสารแบบ SPI เมื่อโมดูล RF ได้รับข้อมูลที่ถูกส่งมาจากไมโครคอนโทรลเลอร์ โมดูล RF จะทำการแปลงข้อมูลให้อยู่ในรูปแบบของความถี่เพื่อที่จะส่งสัญญาณออกไปให้กับโมดูล RF ตัวรับทั้งหมดพร้อมกัน เปรียบเสมือนการส่งสัญญาญบัสแคนผ่านทางอากาศ เมื่อโมดูล RF ตัวรับได้รับข้อมูล ข้อมูลก็จะถูกส่งต่อไปให้กับไมโครคอนโทรลเลอร์ผ่านการสื่อสารแบบ SPI ด้านตัวรับเช่นเดียวกัน เพื่อจะประมวลผลกลับไปให้อยู่ในรูปแบบของข้อมูลแคน โดยทำการตรวจสอบค่าความถูกต้องของข้อมูล ก่อนที่จะส่งข้อมูลออกไปให้กับอุปกรณ์ทดสอบสัญญาณแคนภาครับ ในขณะเดียวกันทางด้านรับเมื่อมีการรับข้อมูลที่จะถูกส่งกลับก็จะดำเนินการแบบเดียวกัน

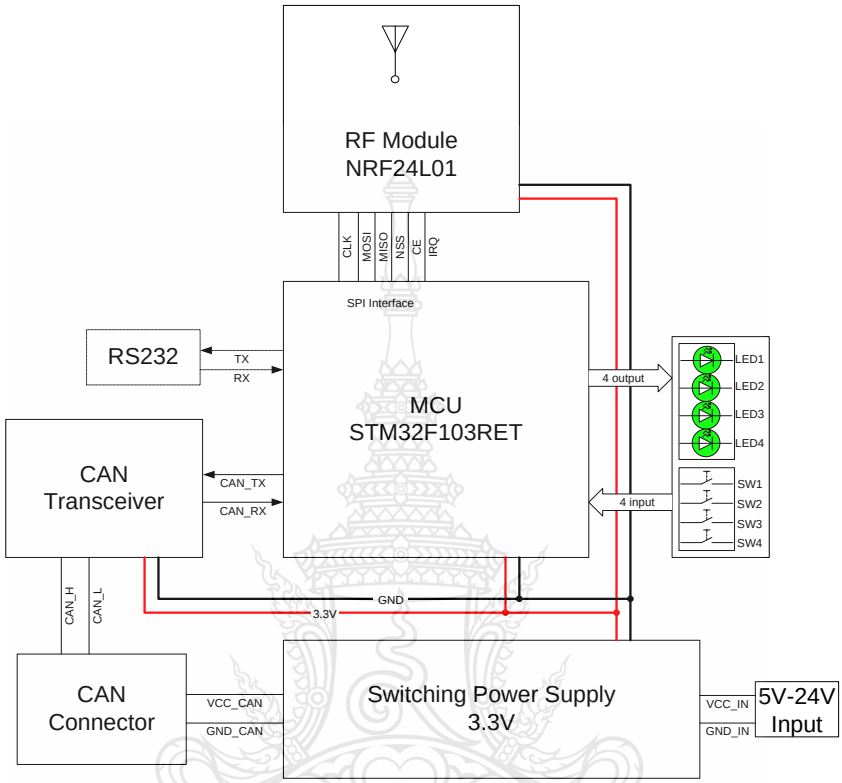
3.3 แนวทางการออกแบบและพัฒนาระบบ

การออกแบบและพัฒนาระบบนั้นจะแยกออกเป็นสองส่วนหลักคือ ส่วนของวงจรหรือฮาร์ดแวร์ และส่วนของโปรแกรมไมโครคอนโทรลเลอร์หรือซอฟแวร์ โดยทำการออกแบบในส่วนของฮาร์ดแวร์และทดสอบการทำงานเบื้องต้นก่อน เพื่อที่จะได้นำซอฟแวร์ที่ได้ออกแบบมาทดสอบการทำงานทีละขั้นตอนหรือการดีบัก (Debug) โปรแกรมให้กับไมโครคอนโทรลเลอร์ ซึ่งการออกแบบระบบการทำงานและการทดสอบนั้นได้แบ่งเป็นสองส่วนด้วยกันคือส่วนของฮาร์ดแวร์และซอฟแวร์ดังนี้

3.4 การออกแบบฮาร์ดแวร์

ในการออกแบบวงจรนี้มีส่วนประกอบอยู่ 4 ส่วนหลักด้วยกันโดยแยกการทำงานของแต่ละส่วนโดยออกแบบตามลักษณะความต้องการของระบบ จากการทำงานโดยรวมของระบบทั้งหมด เพื่อ

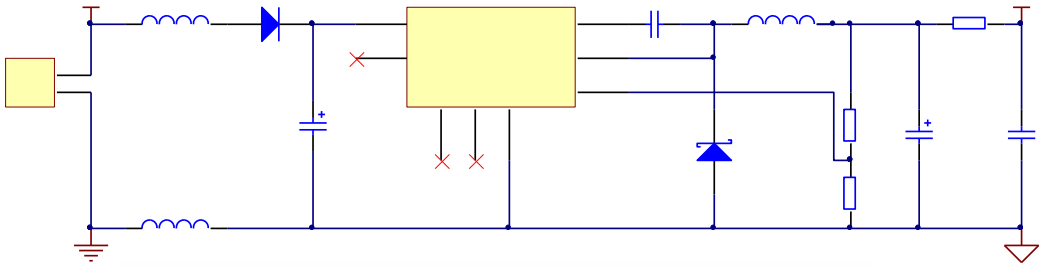
นำมาใช้ในการทดสอบ ซึ่งการออกแบบในส่วนของฮาร์ดแวร์นั้น ได้ทำการออกแบบวงจรตามภาพที่ 3.2 ซึ่งเป็นผังงานการออกแบบฮาร์ดแวร์โดยรวม



ภาพที่ 3.2 ผังงานการออกแบบฮาร์ดแวร์

3.4.1 การออกแบบวงจรภาคจ่ายไฟ

จากผังงานการออกแบบฮาร์ดแวร์แสดงให้เห็นว่าไมโครคอนโทรลเลอร์ STM32F103RET และอุปกรณ์ทั้งหมดนั้นต้องการแรงดันไฟฟ้ากระแสตรง 3.3V สำหรับใช้งานในระบบและสื่อสารข้อมูลระหว่างกัน แต่แหล่งจ่ายไฟโดยทั่วไปนั้นมักจะนิยมใช้งานตั้งแต่ 5V-24V เป็นส่วนมาก ซึ่งความต้องการของระบบนั้นต้องการแรงดันไฟฟ้าคงที่ที่ 3.3V จึงได้ออกแบบวงจรภาคจ่ายไฟที่สามารถปรับแรงดันได้ตั้งแต่ 5V-24V มาเป็น 3.3V คงที่โดยที่ไม่เกิดความร้อนจากการปรับแรงดันไฟฟ้าจนเกิดผลกระทบต่อระบบ จากข้อมูลที่ได้ศึกษาพบว่า การออกแบบภาคจ่ายไฟแบบสวิตชิ่งเพาเวอร์ซัพพลาย (Switching Power Supply) นั้นสามารถปรับแรงดันได้คงที่และเกิดความร้อนจากการปรับแรงดันน้อยมาก ซึ่งเหมาะสมต่อการนำมาประยุกต์ใช้งาน



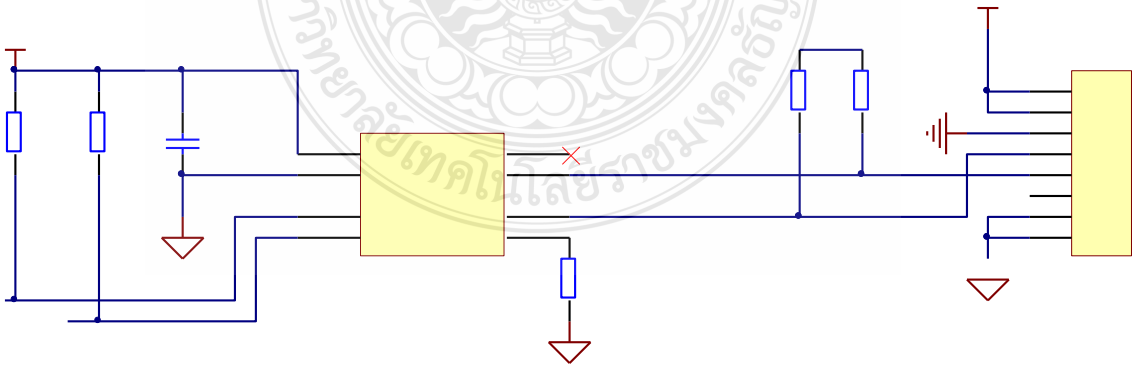
ภาพที่ 3.3 วงจรการทำงานของภาคจ่ายไฟ 3.3 V

จากภาพที่ 3.3 แสดงการออกแบบวงจรภาคจ่ายไฟแบบสวิตชิ่งเพาเวอร์ซัพพลาย เพื่อปรับแรงดันตั้งแต่ 5V-24V มาเป็น 3.3V ให้กับระบบ โดยใช้ไอซีสวิตชิ่งเพาเวอร์ซัพพลาย เบอร์ TPS5420 ของ TI ที่สามารถปรับแรงดันเอาท์พุตได้โดยการปรับค่า Vref ซึ่งจากข้อมูลของผู้ผลิตระบุการคำนวณแรงดันเอาท์พุตจากสมการ

$$R2 = \frac{R1 \times 1.221}{Vout - 1.221} \qquad (3.1)$$

ค่าที่ได้จากการกำหนดค่า R1 = 8.06K และ Vout = 3.3V จะได้ค่า R2 = 4.7K ทำให้ไอซีสามารถจ่ายแรงดันได้ที่ 3.3V ตามความต้องการของระบบ

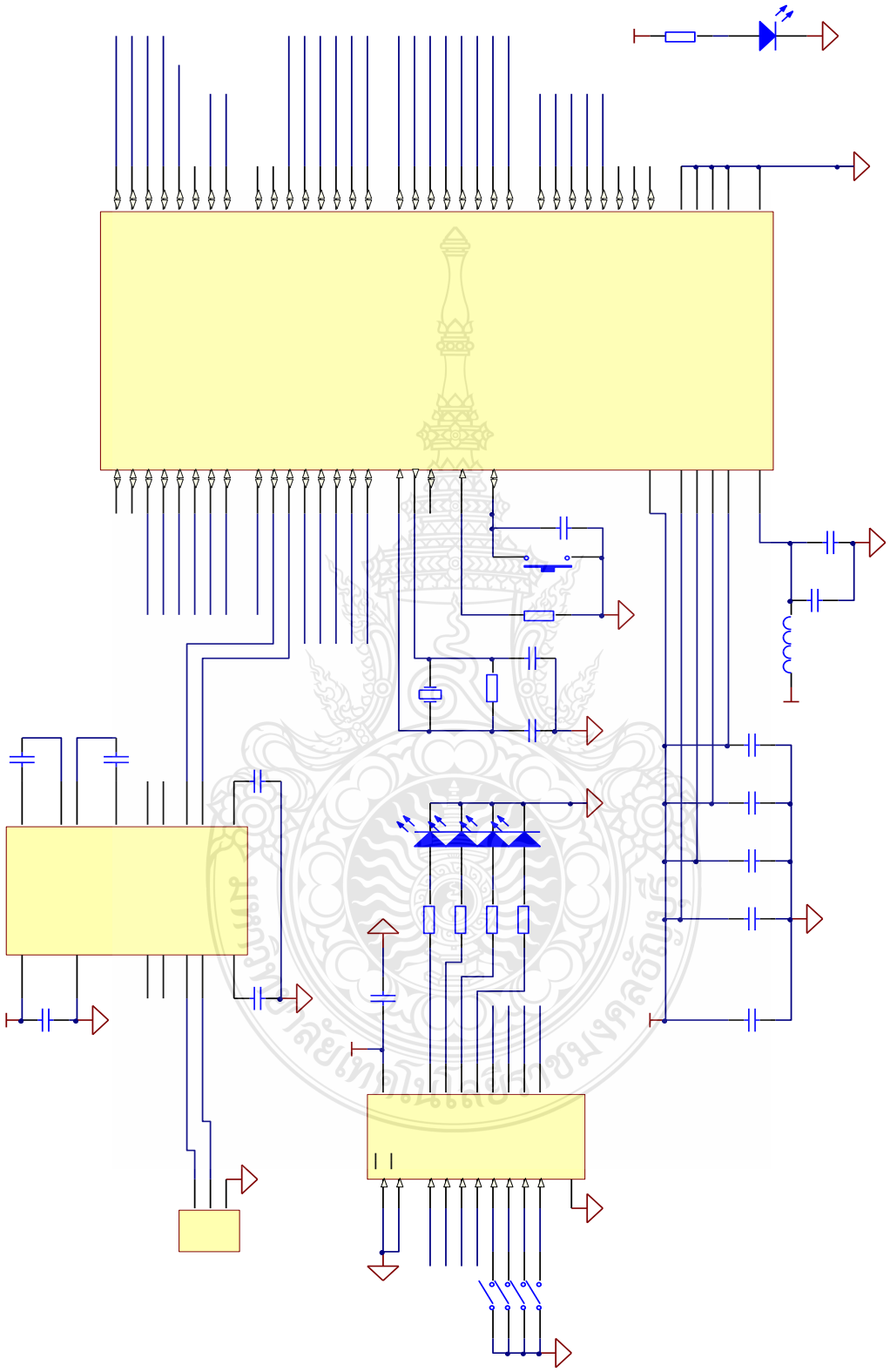
3.4.2 การออกแบบวงจรภาคแคน



ภาพที่ 3.4 วงจรการทำงานของแคนทรานซีฟเวอร์

การออกแบบส่วนที่สองนั้น เป็นการออกแบบวงจรในส่วนของการแปลงรูปแบบสัญญาณแคน ให้สามารถสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ ซึ่งเลือกใช้ชิปแคนทรานซีฟเวอร์ เบอร์ SN65HVD230 ของ TI เป็นชิปที่มีการทำงานที่แรงดันไฟ 3.3V สามารถต่อเข้ากับพอร์ท CAN TX และ CAN RX ของไมโครคอนโทรลเลอร์เพื่อใช้ในการสื่อสารข้อมูลแคนในระบบได้ จากคุณสมบัติของชิปแคนทรานซีฟเวอร์นั้น จะรับสัญญาณ CAN High และ CAN Low แล้วนำมาเปลี่ยนเป็นสัญญาณดิจิตอล เพื่อส่งข้อมูลเข้าที่ขา CAN RX ของไมโครคอนโทรลเลอร์ และรับข้อมูลจากขา CAN TX ของไมโครคอนโทรลเลอร์เพื่อส่งข้อมูลออกไปบนบัสแคน ดังภาพที่ 3.4

3.4.3 การออกแบบวงจรภาคไมโครคอนโทรลเลอร์



ภาพที่ 3.5 วงจรการทำงานของไมโครคอนโทรลเลอร์

จากภาพที่ 3.5 แสดงการออกแบบวงจรควบคุมการทำงานโดยใช้ไมโครคอนโทรลเลอร์เบอร์ STM32F103RET เป็นหน่วยประมวลผลหลัก ซึ่งจากคุณสมบัติของชิปไมโครคอนโทรลเลอร์เบอร์ STM32F103RET สามารถระบุความต้องการในกาออกแบบวงจรไมโครคอนโทรลเลอร์ได้ดังนี้

1) ทำงานที่แรงดันไฟ 3.3V

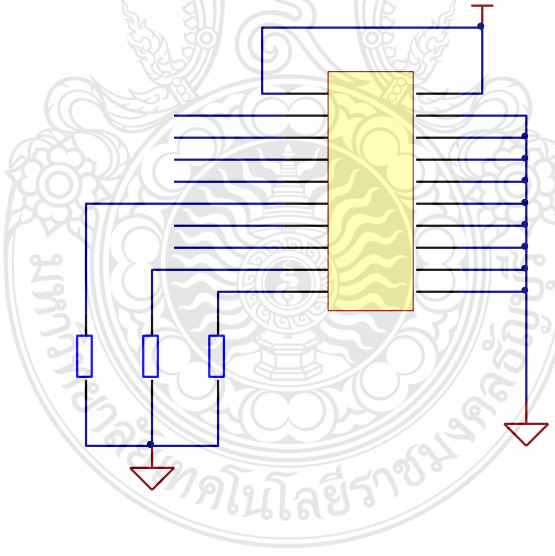
2) สื่อสารกับแคนทรานซีฟเวอร์ และ RF โมดูลโดยใช้แหล่งจ่ายไฟเท่ากัน

3) ใช้สัญญาณนาฬิกาจากคริสตอลออสซิลเลเตอร์หรืออุปกรณ์สร้างสัญญาณนาฬิกาคงที่ (Crystal Oscillator) ขนาด 8 MHz.

4) สื่อสารกับแคนทรานซีฟเวอร์ด้วย ขา 44 (CAN RX) และขา 45 (CAN TX)

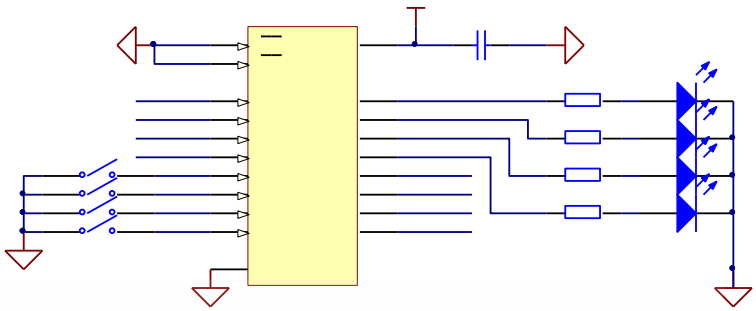
5) สื่อสารกับ RF โมดูล NRF24L01 แบบ SPI อินเตอร์เฟสด้วยขา SPI2_SCK, SPI2_NSS, SPI2_MOSI, SPI2_MISO โดยกำหนดให้ขา PB10 เป็นขาควบคุมการทำงาน (Chip Enable) และ PB11 เป็น IRQ

6) เชื่อมต่อ JTAG ของไมโครคอนโทรลเลอร์เพื่อใช้ในการบรรจุและทดสอบโปรแกรมลงในชิป ตามที่ระบุไว้ในเอกสารคุณสมบัติของไมโครคอนโทรลเลอร์ ดังภาพที่ 3.6



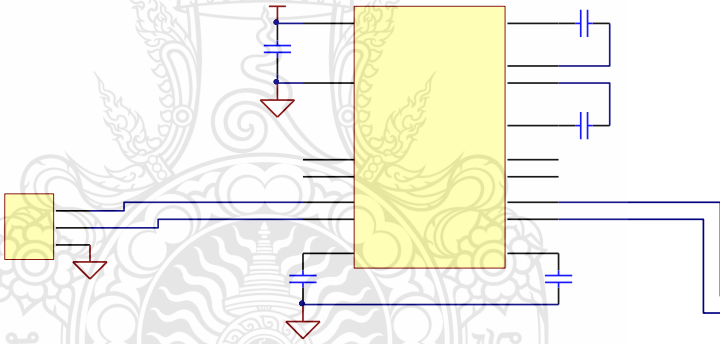
ภาพที่ 3.6 คอนเนคเตอร์ JTAG ของไมโครคอนโทรลเลอร์

7) ออกแบบวงจรเพื่อใช้ทดสอบและแสดงผลอินพุต-เอาท์พุต โดยใช้ชิปบัฟเฟอร์(Buffer) เป็นตัวกลางในการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับ LED และสวิตซ์แสดงในภาพที่ 3.7



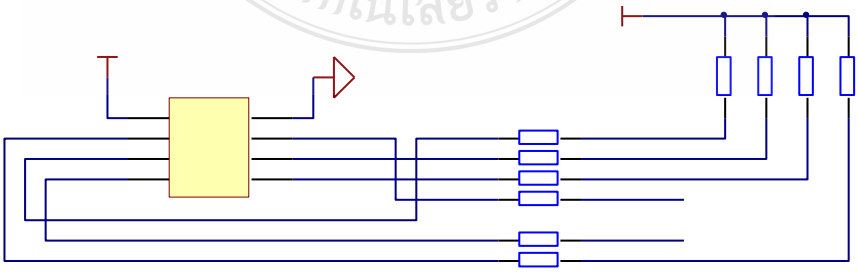
ภาพที่ 3.7 วงจรอินพุต-เอาท์พุตของไมโครคอนโทรลเลอร์

8) เพิ่มเติมในส่วนของการแสดงผลและส่วนทดสอบระบบโดยใช้ การเชื่อมต่อแบบ RS232 เพื่อใช้ในการติดต่อกับคอมพิวเตอร์ในการแสดงผลการรับ-ส่งข้อมูล ของขั้นตอนการทดสอบโปรแกรมเบื้องต้นดังภาพที่ 3.8



ภาพที่ 3.8 วงจรติดต่อสื่อสารแบบ RS232 ของไมโครคอนโทรลเลอร์

3.4.3 การออกแบบวงจรภาคติดต่อ RF โมดูล



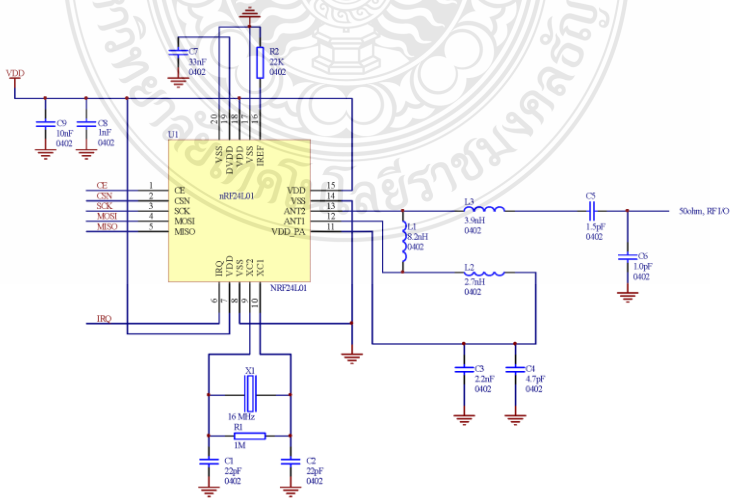
ภาพที่ 3.9 วงจรการติดต่อโมดูล RF แบบ SPI

จากภาพที่ 3.9 เป็นส่วนติดต่อโมดูลรับส่ง RF 2.4 GHz โดยเลือกใช้โมดูลไร้สายเบอร์ NRF24L01 มีการเชื่อมต่อกับไมโครคอนโทรลเลอร์ด้วยการสื่อสารแบบ SPI อินเตอร์เฟสโดยรับข้อมูลที่ถูกประมวลผลจากไมโครคอนโทรลเลอร์เพื่อส่งไปยังโมดูลตัวรับ ซึ่งโมดูลไร้สายเบอร์ NRF24L01 นั้นมีคอนเน็คเตอร์สำหรับต่อใช้งานดังตารางที่ 3.1

ตารางที่ 3.1 ขาคอนเน็คเตอร์ของโมดูลไร้สาย

ขาที่	ชื่อ	คุณสมบัติการทำงาน
1	Ground	กราวด์ของระบบ
2	VCC	ไฟเลี้ยงของโมดูล (3.3V)
3	CE	Chip Enable ใช้สำหรับอีนาเบิล (เปิด) การทำงานของโมดูล
4	NSS	Chip Select ใช้ควบคุมการสื่อสารในรูปแบบของ SPI
5	SCK	Clock ใช้รับสัญญาณนาฬิกา จากไมโครคอนโทรลเลอร์
6	MOSI	Data Input ใช้รับข้อมูลจากไมโครคอนโทรลเลอร์
7	MISO	Data Output ใช้ส่งข้อมูลให้กับไมโครคอนโทรลเลอร์
8	IRQ	Interrupt ใช้ส่งสัญญาณอินเตอร์รัพท์ให้กับไมโครคอนโทรลเลอร์

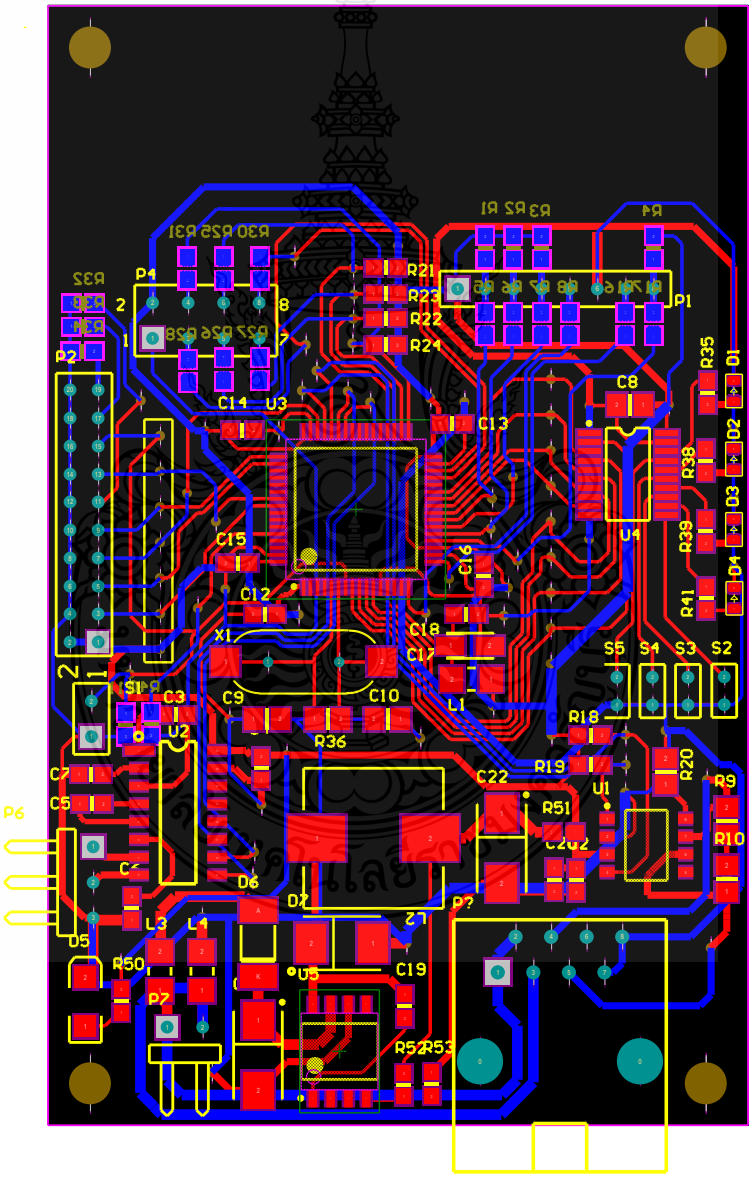
โดยขาคอนเน็คเตอร์ของโมดูลไร้สาย RF นั้นจะเชื่อมต่อกับชิป NRF42L01 โดยตรง ซึ่งไมโครคอนโทรลเลอร์จะสามารถติดต่อกับไอซีของโมดูลได้ถ้ามีการต่อวงจรการทำงานที่ถูกต้อง โดยที่โมดูลไร้สายมีวงจรภายในดังภาพที่ 3.10



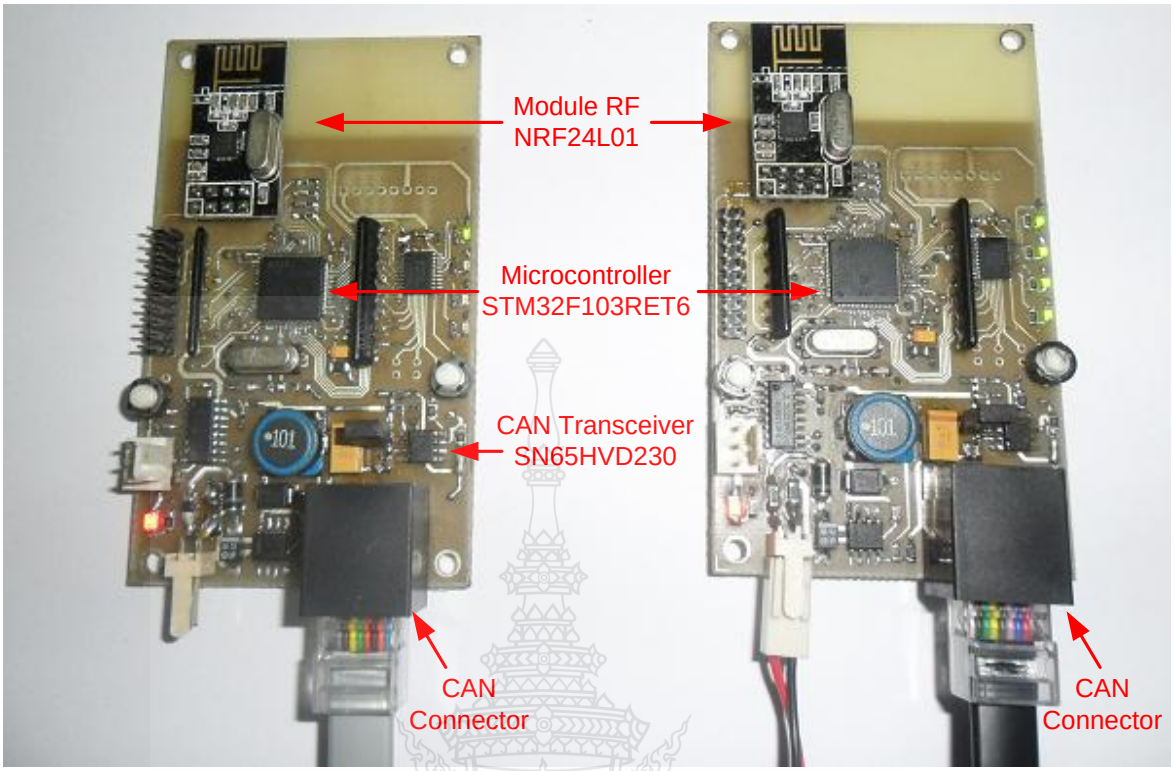
ภาพที่ 3.10 วงจรภายในโมดูล NRF24L01

3.4.5 การออกแบบบอร์ดโมดูล

จากส่วนประกอบของวงจรทั้งหมดสามารถนำมาประกอบวงจรรวมและออกแบบลายวงจรเพื่อใช้ทดสอบได้ โดยกำหนดให้บอร์ดนั้นรับไฟเลี้ยงวงจรได้ทั้งสองแบบคือ จากดีซีคอนเน็คเตอร์ (DC Connector) ของภาคจ่ายไฟโดยตรงและจากแคนคอนเน็คเตอร์ เพื่อที่จะสามารถทดสอบการทำงานที่ระยะทางไกลจากแหล่งจ่ายไปโดยที่ไม่ต่องเพิ่มสายได้ รวมทั้งจัดวางอุปกรณ์ตามวงจรที่ได้ออกแบบไว้ในแต่ละส่วนให้เป็นระบบ เพื่อให้ขนาดบอร์ดมีขนาดที่พอเหมาะแก่การทดสอบดังแสดงในภาพที่ 3.11



ภาพที่ 3.11 ลายวงจรของบอร์ดโมดูลที่ใช้ทดสอบ



ภาพที่ 3.12 บอร์ดโมดูลสำเร็จสำหรับทดสอบการทำงานของระบบ

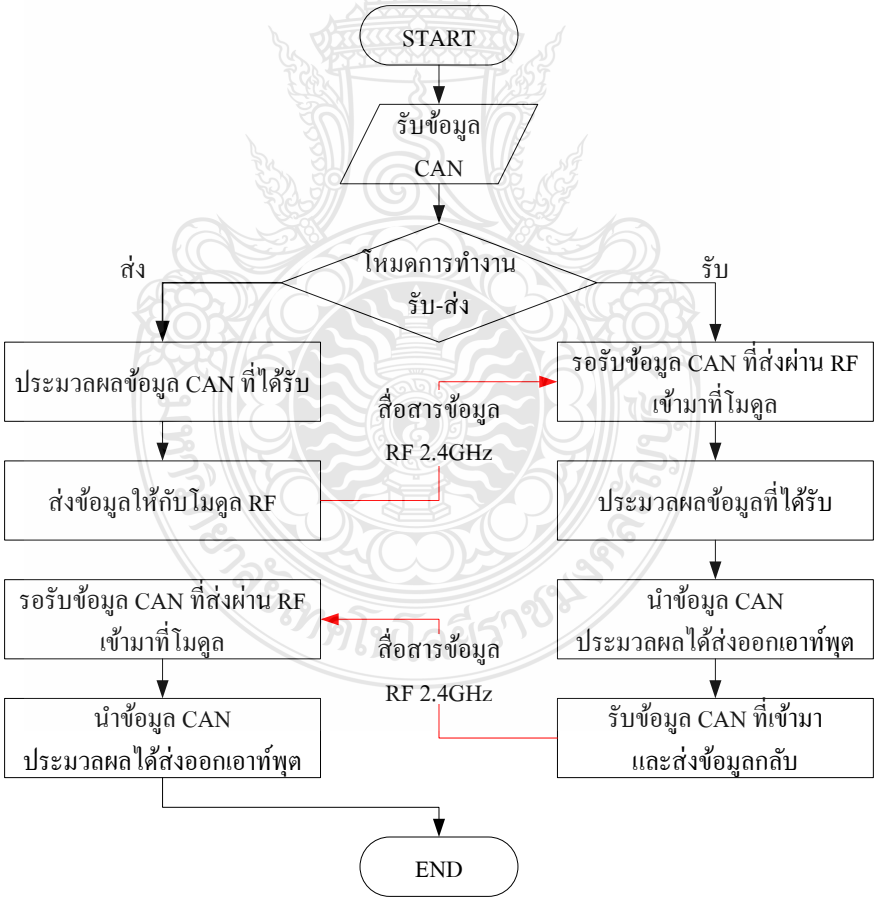
จากภาพที่ 3.12 แสดงบอร์ดที่ประกอบอุปกรณ์ทั้งหมดสำเร็จและประกอบโมดูล RF พร้อมทั้งทดสอบการทำงานของระบบในเบื้องต้น โดยการจ่ายไฟให้กับบอร์ดเพื่อตรวจสอบการทำงานของภาคจ่ายไฟว่ามีแรงดันที่ใช้ในระบบเท่ากับ 3.3V ซึ่งโมดูลที่นำมาใช้ทดสอบนั้นสามารถทำงานได้ รวมทั้งทดสอบว่าไมโครคอนโทรลเลอร์สามารถโปรแกรมซอร์ซโค้ดหรือติดต่อกับ JTAG โดยการเชื่อมต่อกับยูเอสบี ARM JTAG ผ่าน JTAG คอนเน็คเตอร์ และทดสอบด้วยซอฟแวร์ Rowley Crosswork ARM 1.7 ว่าสามารถเชื่อมต่อได้หรือไม่ในการเตรียมพร้อมก่อนที่จะนำโมดูลไปทดสอบกับโปรแกรมที่ออกแบบไว้ต่อไป

3.5 การออกแบบซอฟแวร์

ในการออกแบบซอฟแวร์นั้นจะมีการทำงานของระบบแยกออกมาเป็นสองส่วนหลักด้วยกันคือส่วนของภาคส่ง-ภาครับ และส่วนประมวลผลข้อมูล ซึ่งการทดสอบในงานวิจัยนี้ได้ใช้ภาษาซี ในการพัฒนาแอพพลิเคชั่น เนื่องจากภาษาซีมีความสามารถในการพัฒนาการทำงานของโปรแกรมไมโครคอนโทรลเลอร์มากกว่าภาษาแอสเซมบลี ถึงแม้ว่าภาษาแอสเซมบลีจะมี

ความสามารถในการเขียนโปรแกรมเพื่อใช้งานไมโครคอนโทรลเลอร์หรือเข้าถึงรีจิสเตอร์ได้โดยตรง แต่การนำมาประยุกต์ใช้งานร่วมกับอุปกรณ์หรือการสื่อสารระหว่างโมดูลอื่นๆนั้น จำเป็นจะต้องเขียนโปรแกรมเพื่อกำหนดการเชื่อมต่อหรือไดร์เวอร์ (Driver) ในการสื่อสารกับอุปกรณ์ต่างๆ ซึ่งเป็นข้อจำกัดเพราะผู้ผลิตอุปกรณ์หรือชิปบางแห่ง ไม่เปิดเผยการเข้าถึงข้อมูลโครงสร้างภายในหรือ ไม่ได้มีเอกสารระบุการใช้งานร่วมกับภาษาแอสเซมบลีไว้ให้ ดังนั้นในงานวิจัยนี้จึงได้เลือกภาษาซีในการพัฒนาโปรแกรมเพื่อใช้งานร่วมกับไมโครคอนโทรลเลอร์ โดยพัฒนาภาษาซีด้วยโปรแกรม Rowley Crosswork ARM 1.7 และใช้ ARM JTAG สำหรับส่วนบันทึกโปรแกรมลงในไมโครคอนโทรลเลอร์ และ ดีบักโปรแกรมบนไมโครคอนโทรลเลอร์ เพื่อทดสอบการทำงานของระบบว่ามีการทำงานตามขั้นตอนได้อย่างถูกต้องตามที่ได้ออกแบบไว้ โดยการออกแบบซอฟแวร์นั้นจะมีหลักการทำงานโดยรวมดังนี้

3.5.1 ภาพรวมการทำงานของระบบ



ภาพที่ 3.13 ผังการออกแบบซอฟแวร์

จากภาพที่ 3.13 แสดงผังการออกแบบซอฟแวร์สำหรับใช้ทดสอบการทำงานของโมดูลไมโครคอนโทรลเลอร์ที่ได้ออกแบบไว้ โดยมีลำดับขั้นตอนการทำงานดังนี้

1) เมื่อโมดูลได้รับข้อความแคน ที่ถูกส่งเข้ามาและอยู่ในสภาวะว่างหรือโหมดรอรับข้อมูลและไม่มีการรับข้อมูลอยู่ โมดูลก็จะเปลี่ยนสถานะเป็นโหมดส่งข้อมูล แต่ถ้าไม่ว่างในขณะนั้นจะทำการเก็บข้อมูลแคน ไว้ในเมล์บ็อกซ์ก่อน

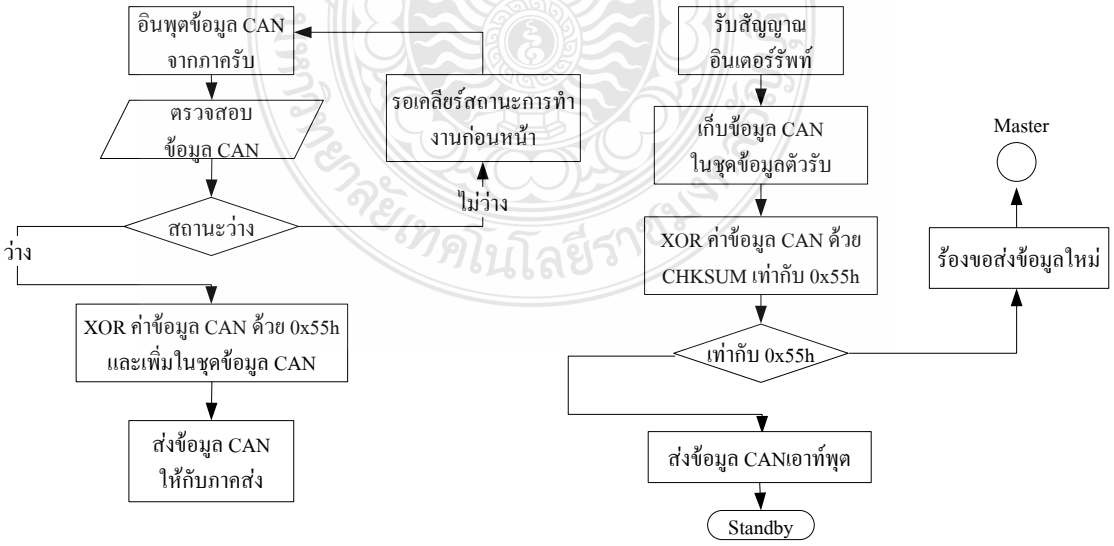
2) เมื่อถูกกำหนดการทำงานมีการระบุการทำงานว่าทำหน้าที่ภาคส่งแล้ว จะนำข้อมูล แคนที่ได้รับมาทำการแบ่งเฟรมข้อความออกเป็นส่วนๆ ไมโครคอนโทรลเลอร์จะทำการแยกข้อความออกจากกันเพื่อเก็บค่าไว้ในชุดข้อมูลที่จะทำการส่ง

3) โดยกำหนดให้ในชุดข้อมูลที่ทำการส่งนั้น มีข้อมูลบิตที่ใช้ในการตรวจสอบค่าความถูกต้องของข้อมูลรวมอยู่ด้วยเพื่อใช้ในการตรวจสอบความผิดพลาดหลังจากที่ได้ส่งข้อมูลและได้รับข้อมูลแล้ว

4) เมื่อทำการระบุชุดข้อมูลที่ทำการส่งเรียบร้อยก็จะส่งข้อมูลให้กับโมดูล RF ทำหน้าที่ส่งข้อมูลออกไปและรอรับข้อมูลตอบกลับจนสิ้นสุดกระบวนการ

5) ส่วนภาครับทำหน้าที่รอข้อมูลที่ถูกส่งมาเมื่อมีการรับข้อมูลเข้ามาโมดูลนั้นจะทำหน้าที่รับข้อมูลแล้วส่งให้ไมโครคอนโทรลเลอร์ประมวลผลข้อมูลที่ได้รับออกมาเป็นข้อมูลแคน เพื่อส่งข้อมูลออกให้กับอุปกรณ์ปลายทาง และรอรับข้อมูลแคน ที่ถูกส่งกลับไปที่ภาคส่งเพื่อยืนยันว่ามีการติดต่อสื่อสารกันอย่างสมบูรณ์แล้ว

3.5.2 การออกแบบซอฟแวร์ภาคประมวลผลข้อมูล



ภาพที่ 3.14 ผังการออกแบบซอฟแวร์ภาคประมวลผลข้อมูล

จากภาพที่ 3.14 แสดงผังการทำงานของภาคประมวลผลข้อมูล โดยมีหน้าที่หลักในการควบคุมการทำงานของระบบทั้งหมดและทำหน้าที่ประมวลผลข้อมูลแคน เพื่อรับ-ส่งระหว่างโมดูลทั้งหมด มีการทำงานตามผังงานดังนี้

1) ตรวจสอบข้อมูลแคน ว่าถ้ามีการรับข้อมูลเข้ามาและยังอยู่ในขั้นตอนการรับ-ส่งให้รอจนสิ้นสุดกระบวนการรับส่ง และเก็บข้อมูลไว้ในเมล์บ็อกซ์

2) ประมวลผลชุดข้อมูลแคน (CAN Packet) และเก็บค่าไว้ชุดเก็บข้อมูลแคน

3) นำชุดข้อมูลแคน ที่ได้มาวนลูปบวกค่าข้อมูลระดับบิต (เอ็กซ์คลูซีฟออร์ : XOR) ค่าภายในทั้งหมดด้วย 0x55h และเก็บค่าผลลัพธ์ที่ได้ไว้ส่วนท้ายของชุดเก็บข้อมูลเพื่อใช้ไว้เป็นค่าตรวจสอบความผิดพลาด

4) ส่งข้อมูลที่ประมวลผลแล้วให้กับภาคส่งข้อมูลเพื่อส่งข้อมูลให้กับโมดูลไร้สาย

5) รับสัญญาณอินเตอร์รัปต์ของโมดูลไร้สายเพื่อรับข้อมูลจากโมดูลอื่นที่ส่งเข้ามา

6) นำข้อมูลจากภาครับเข้ามาเก็บไว้ในชุดข้อมูลแคนตัวรับ

7) นำชุดข้อมูลแคน ตัวรับที่ได้มาวนลูปบวกค่าข้อมูลระดับบิตค่าภายในทั้งหมด ด้วยข้อมูลตรวจสอบความผิดพลาดที่ถูกส่งมาด้วย

8) ถ้าผลลัพธ์ไม่เท่ากับ 0x55h ให้ส่งข้อมูลร้องขอให้โมดูลตัวส่ง ส่งข้อมูลกลับมาอีกครั้ง

9) ตรวจสอบค่าความถูกต้องและประมวลผลเพื่อส่งข้อมูลแคนออกสู่ระบบ

10) เปลี่ยนสถานะมาอยู่ในโหมดแสตนด์บายเพื่อรับข้อมูล

3.5.3 การออกแบบซอฟแวร์ภาครับ-ส่งข้อมูลไร้สาย

การทำงานของภาครับส่งข้อมูลไร้สายโดยที่จะแยกการทำงานเป็นภาคส่งข้อมูล และภาครับข้อมูล ดังภาพที่ 3.15 โดยการควบคุมการทำงานนั้นจะถูกกำหนดหรือถูกควบคุมโดยภาคประมวลผลข้อมูล มีหลักการทำงานดังนี้

1) เริ่มต้นถ้ามีข้อมูลแคนเข้ามาในระบบแล้วโมดูลจะถูกกำหนดว่าเป็นตัวส่ง

2) ตรวจสอบว่าอยู่ในสถานะพร้อมและเคลียร์หน่วยเก็บข้อมูลชั่วคราวหรือบัฟเฟอร์ให้ว่างเพื่อรอรับข้อมูล

3) รับข้อมูลที่ผ่านการประมวลผลแล้วเก็บในบัฟเฟอร์

4) ส่งข้อมูลในบัฟเฟอร์ให้กับโมดูลไร้สายเพื่อส่งข้อมูลให้กับโมดูลไร้สายที่รอรับข้อมูล

5) เคลียร์ข้อมูลในบัฟเฟอร์และเปลี่ยนสถานะจากโหมดส่งข้อมูลเป็นโหมดรอรับข้อมูล

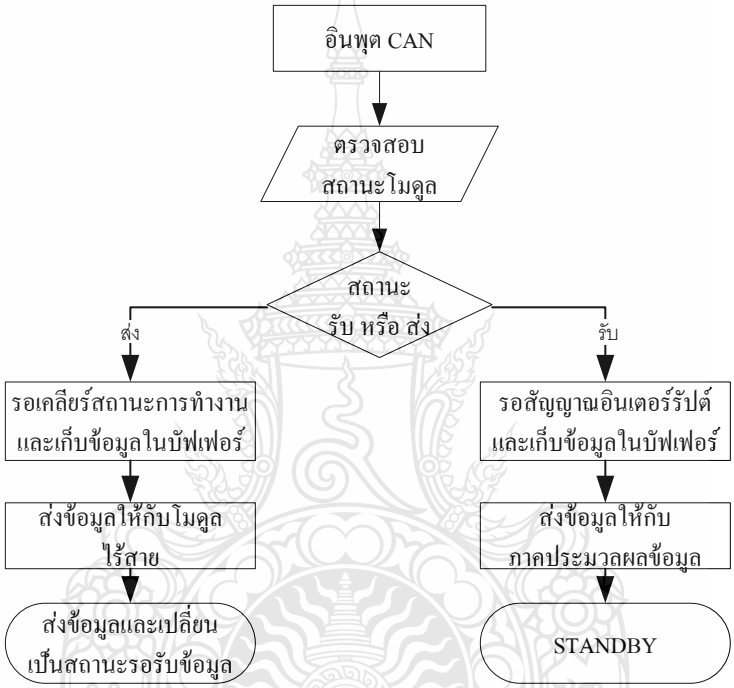
6) เมื่ออยู่ในสถานะรอรับข้อมูลระบบจะรอสัญญาณอินเตอร์รัปต์ เพื่อรับข้อมูล

7) เมื่อสัญญาณอินเตอร์รัปต์ถูกส่งมาจากโมดูลไร้สาย ระบบจะรับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์ตัวรับ

8) ส่งข้อมูลจากบัฟเฟอร์ตัวรับให้กับภาคประมวลผลข้อมูล

9) เคลียร์ข้อมูลในบัฟเฟอร์และรอการส่งข้อมูลร้องขอกลับไปที่โมดูลตัวส่งในกรณีที่ข้อมูลไม่สมบูรณ์

10) เปลี่ยนสถานะมาอยู่ในโหมดแสตนด์บาย



ภาพที่ 3.15 ผังการทำงานของภาครับส่งข้อมูลไร้สาย

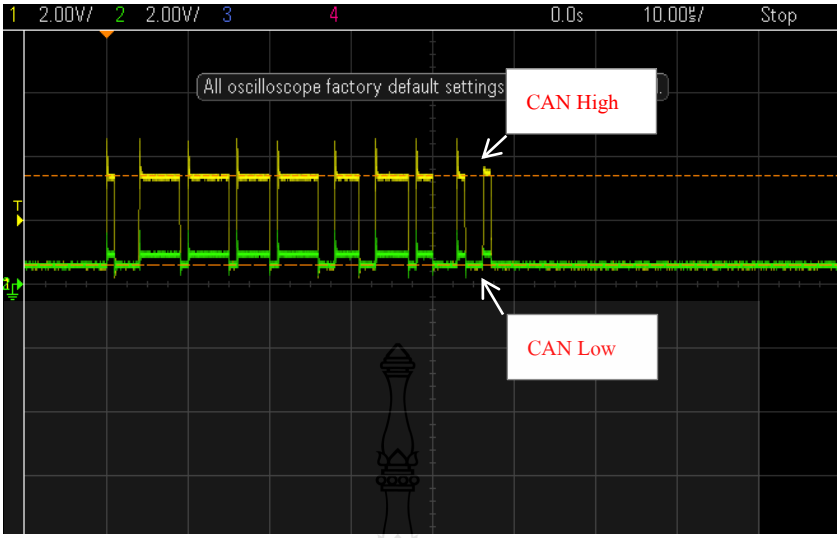
บทที่ 4
ผลการวิจัย

การทดสอบงานวิจัยนั้นจะทำการทดสอบโดยการนำสัญญาณอินพุตของแคน ป้อนให้กับวงจรโดยใช้โปรแกรมและอุปกรณ์กำเนิดสัญญาณแคน ซึ่งโมดูลที่ทำการทดสอบมีทั้งหมด 4 ชุด โดยที่โมดูลที่ใช้ทดสอบทุกตัวสามารถทำงานได้ทั้งภาคส่งและภาครับข้อมูล ทำการทดสอบที่สภาพแวดล้อมแบบปิดไม่มีสัญญาณรบกวนโดยนำอุปกรณ์ที่ทำการทดสอบวางไว้ห่างจากอุปกรณ์ตัวส่งเป็นระยะทางที่กำหนดในการทดสอบแต่ละครั้ง ก่อนเพิ่มระยะห่าง จนถึงจุดที่สามารถรับสัญญาณได้ไกลสุด เริ่มทำการทดสอบที่ภาคส่งโดยทำการกำหนดค่าข้อความโดยโปรแกรม PCAN-View และส่งค่าออกมาผ่านทาง PCAN-USB ให้กับโมดูลที่ใช้ทดสอบในระบบ ซึ่งในโหมดการทดสอบนั้นจะกำหนดให้โมดูลที่รับสัญญาณทำการบวกค่าของข้อมูลที่ได้รับ ด้วย 0x01h เพื่อเป็นการระบุว่าข้อมูลที่ได้รับนั้นผ่านการส่งไปให้กับโมดูลตัวรับโดยรูปแบบไร้สายไม่ใช่สัญญาณย้อนกลับหรือ Loopback ดังนั้นในขั้นตอนการทดสอบจะแบ่งเป็นการทดสอบเบื้องต้นระหว่างสองโมดูลเพื่อทดสอบการทำงานของฮาร์ดแวร์ว่า มีความสมบูรณ์หรือไม่โดยทำการวัดสัญญาณระหว่างไมโครคอนโทรลเลอร์และโมดูล RF เพื่อพิสูจน์การทำงานระหว่างกันและเป็นการตรวจสอบข้อมูลที่รับส่งระหว่างกันด้วย หลังจากทราบว่า ฮาร์ดแวร์โมดูลสามารถทำงานได้อย่างสมบูรณ์แล้ว ก็จะดำเนินการทดสอบการทำงานของโมดูลทั้งหมดพร้อมกัน

4.1 การทดสอบการทำงานของอุปกรณ์หลักบนบอร์ดโมดูล

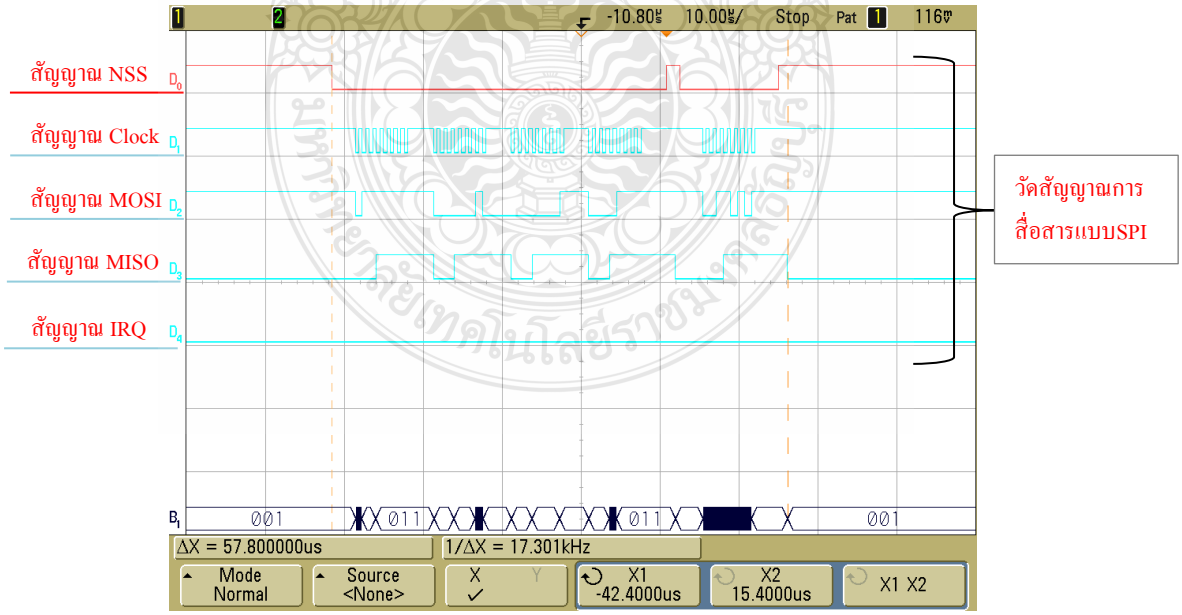
การทดสอบการทำงานของอุปรณ์หลักบนบอร์ดนั้น ประกอบด้วยการทดสอบการทำงานภาคแคน และภาคติดต่อกับโมดูลไร้สาย NRF24L01 โดยทำการวัดสัญญาณอุปกรณ์หลักบนบอร์ดเพื่อพิสูจน์ว่าอุปกรณ์หรือวงจรที่ออกแบบไว้สามารถทำงานร่วมกับไมโครคอนโทรลเลอร์ได้ ซึ่งได้ทำการทดสอบดังนี้

4.1.1 วัดสัญญาณ CAN High และ CAN Low ของภาคแคนที่ไอซี SN65HVD230 ดังภาพที่ 4.1 ได้แสดงสัญญาณที่เกิดความต่างศักย์กัน (Differential) ระหว่าง CAN High และ CAN Low ซึ่งทำการวัดสัญญาณขณะที่กำลังทำงานที่ 3.3 V



ภาพที่ 4.1 วัดสัญญาณ CAN High และ CAN Low

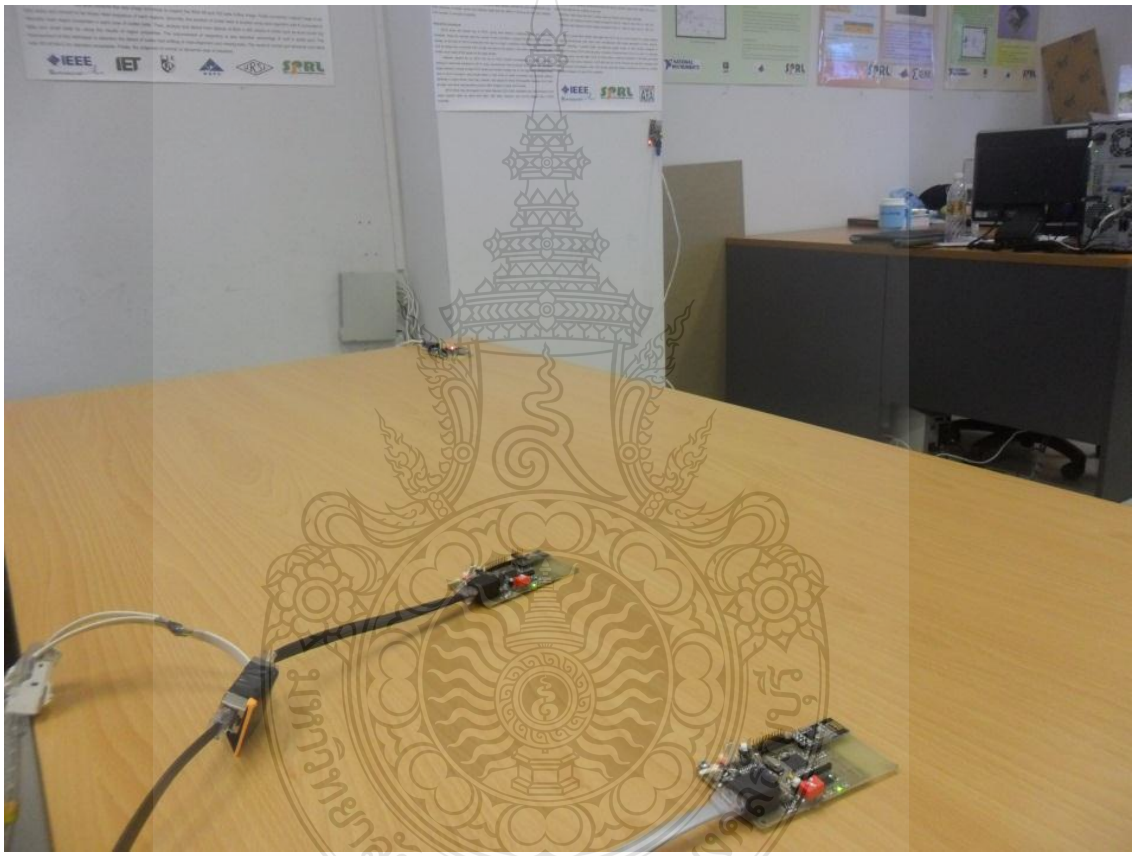
4.1.2 วัดสัญญาณ SPI ระหว่างไมโครคอนโทรลเลอร์และโมดูลไร้สาย NRF24L01 ดังภาพที่ 4.2 ได้แสดงการสื่อสารข้อมูลที่ไมโครคอนโทรลเลอร์ทำการอ่านค่าข้อมูลภายในโมดูลไร้สาย NRF24L01 และยังเป็นการทดสอบว่าไมโครคอนโทรลเลอร์สามารถติดต่อกับโมดูลไร้สาย NRF24L01 ได้อย่างสมบูรณ์ด้วย



ภาพที่ 4.2 วัดสัญญาณ SPI ระหว่างไมโครคอนโทรลเลอร์กับโมดูลไร้สาย

4.2 สภาพแวดล้อมในการทดสอบ

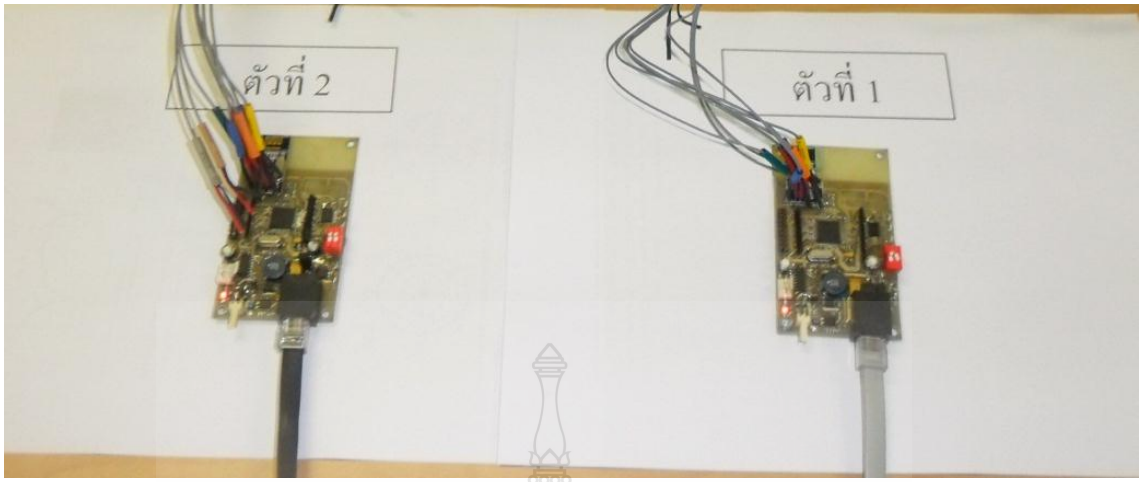
อุปกรณ์ที่นำมาใช้ในการทดสอบคืออุปกรณ์ในการสร้างและรับสัญญาณหรือข้อมูลในรูปแบบแคน (PCAN USB) โปรแกรม PCAN-View และบอร์ดวงจรไมโครคอนโทรลเลอร์ STM32F103RET ที่ออกแบบ นำมาต่อร่วมกับโมดูล RF 2.4GHz (NRF24L01) จำนวน 4 บอร์ด จากภาพที่ 4.3 แสดงการทดสอบโมดูลที่ระยะทางแตกต่างกัน โดยทำการส่งสัญญาณที่โมดูลตัวแรก และรับข้อมูลด้วยโมดูลทดสอบทั้งหมด เพื่อนำข้อมูลมาเปรียบเทียบหาผลลัพธ์ที่เกิดขึ้น



ภาพที่ 4.3 สภาพแวดล้อมการทดสอบของระบบที่ระยะทางแตกต่างกัน

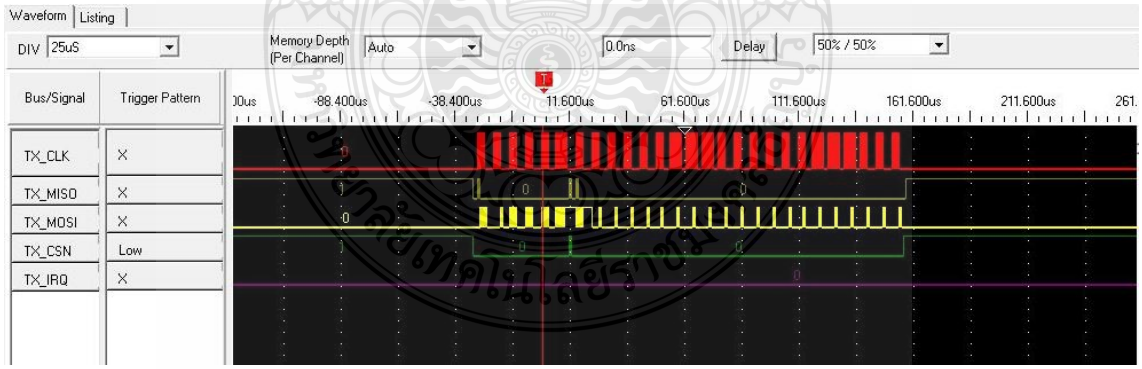
4.3 การทดสอบการทำงานของระบบเบื้องต้น

การทดสอบระบบเบื้องต้นนั้น เป็นการทดสอบฮาร์ดแวร์ ของโมดูลที่ได้ออกแบบไว้ว่ามีการทำงานถูกต้องสมบูรณ์หรือไม่ โดยนำโมดูลที่ใช้ทดสอบจำนวนสองชุดมาทดสอบการรับ-ส่งข้อมูล ซึ่งทำการกำหนดให้โมดูลตัวที่ 1 เป็นตัวส่งและโมดูลตัวที่ 2 เป็นตัวรับ ดังภาพที่ 4.4



ภาพที่ 4.4 การทดสอบโมดูลเบื้องต้น

นำสัญญาณอินพุตของแคนป้อนให้กับวงจรโดยใช้ PCAN-USB ต่อเข้ากับแคนคอนเน็คเตอร์ทั้งสองชุด และใช้โปรแกรม PCAN-View ส่งค่าออกมาผ่านทาง PCAN-USB ให้กับโมดูลที่ใช้ทดสอบทั้งตัวรับและตัวส่ง ซึ่งโมดูลที่ถูกกำหนดให้เป็นตัวส่งนั้นจะทำการประมวลผลข้อมูลแล้วส่งข้อมูลให้กับโมดูล RF ผ่านรูปแบบการสื่อสารแบบ SPI อินเตอร์เฟส และทำการวัดสัญญาณ SPI ระหว่างไมโครคอนโทรลเลอร์และโมดูล RF โดยทำการวัดสัญญาณที่ภาคส่งเพื่อตรวจสอบว่าข้อมูลแคน ที่กำหนดจากโปรแกรม PCAN-View และข้อมูลที่ถูกส่งให้กับโมดูล RF เป็นค่าที่ถูกต้อง

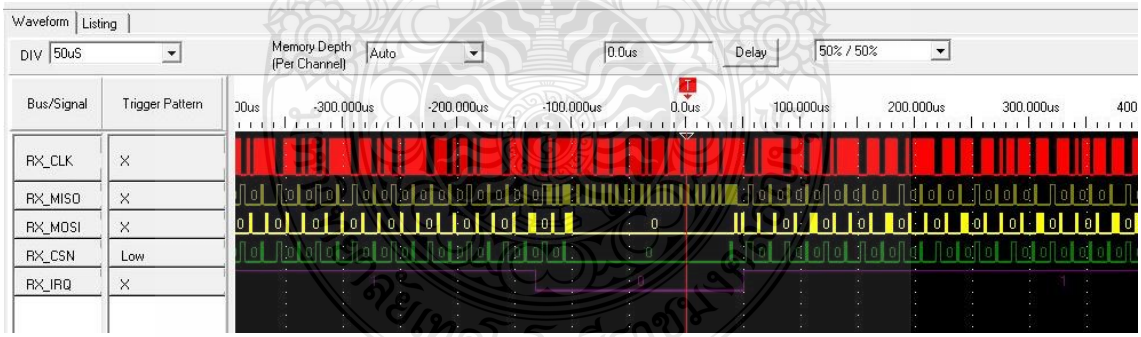


ภาพที่ 4.5 วัดสัญญาณระหว่างไมโครคอนโทรลเลอร์กับโมดูล NRF24L01 (ส่งข้อมูล)

จากภาพที่ 4.5 แสดงสัญญาณระหว่างการเชื่อมต่อของไมโครคอนโทรลเลอร์ กับ โมดูล RF โดยจับสัญญาณ ของ SPI ระหว่างไมโครคอนโทรลเลอร์กับโมดูล RF เพื่อทดสอบว่ามีการส่งข้อมูลระหว่างกันในขณะส่งข้อมูล ซึ่งสัญญาณที่ทำการวัดนั้นมีคำอธิบายดังนี้

1) TX_CLK คือสัญญาณนาฬิกาของ SPI ที่ถูกส่งออกมาโดยไมโครคอนโทรลเลอร์
2) TX_MISO คือสัญญาณ MISO ของ SPI มีค่าข้อมูลเป็น 0 เพราะอยู่ในโหมดส่งข้อมูล
3) TX_MOSI คือสัญญาณ MOSI ของ SPI ที่ส่งข้อมูลจากไมโครคอนโทรลเลอร์ให้กับโมดูล RF ซึ่งเป็นข้อมูลที่ถูกประมวลผลแล้ว
4) TX_CS คือสัญญาณขาการเลือก (Chip Select : CS) ของสัญญาณ SPI ที่ถูกส่งออกมาโดยไมโครคอนโทรลเลอร์เพื่อกำหนดการติดต่อระหว่างไมโครคอนโทรลเลอร์กับโมดูล RF โดยการเปลี่ยนสภาวะจาก 1 (High) เป็น 0 (Low)
5) TX_IRQ คือสัญญาณอินเตอร์รัปต์ของ RF โมดูล ที่ไมโครคอนโทรลเลอร์จะใช้งานในโหมดรับเท่านั้น

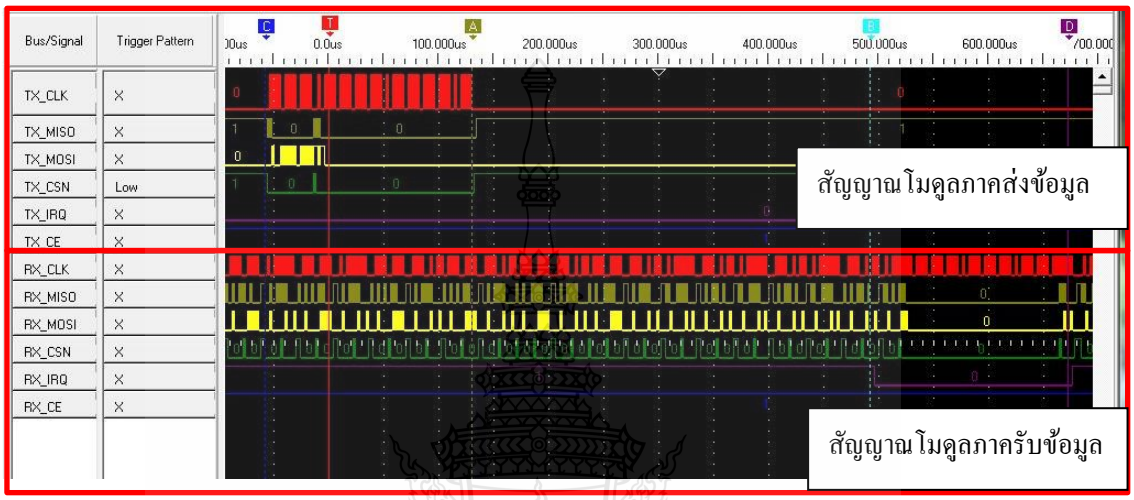
สัญญาณที่วัดได้นั้นแสดงให้เห็นว่า ค่าข้อมูลแคน ที่ถูกส่งออกมาให้กับโมดูล RF มีค่าเป็น 0x01h ทั้งหมด จากการกำหนดด้วยโปรแกรม PCAN-View ให้ข้อมูลแคน ที่ใช้ในการส่งนั้นเป็น 0x01h แสดงให้เห็นการเปรียบเทียบข้อมูลในโปรแกรม PCAN-View และข้อมูลของ SPI ดังภาพที่ 4.6



ภาพที่ 4.6 วัดสัญญาณระหว่างไมโครคอนโทรลเลอร์กับโมดูล NRF24L01 (รับข้อมูล)

รับข้อมูลแคนที่ถูกส่งกลับโดยใช้โมดูลตัวรับ ทำการวัดสัญญาณระหว่างการเชื่อมต่อของไมโครคอนโทรลเลอร์กับโมดูล RF โดยวัดสัญญาณ ของ SPI ทางด้านรับว่ามีการวนรับข้อมูลที่ถูกส่งมาแล้วทำการส่งต่อข้อมูลให้กับไมโครคอนโทรลเลอร์

ทดสอบช่วงเวลารับส่งระหว่างโมดูล โดยวัดสัญญาณขณะทำการส่งและรับข้อมูลระหว่างโมดูล RF และไมโครคอนโทรลเลอร์ พบว่าหลังจากทำการส่งข้อมูลออกไปแล้วจะมีช่วงเวลาประมาณ 360ไมโครวินาที ที่โมดูลตัวรับ รับข้อมูลเข้ามาประมวลผลจากภาพที่ 4.7 แสดงผลการวัดสัญญาณ



ภาพที่ 4.7 การรับ-ส่งข้อมูลระหว่างสองโมดูล

4.4 การทดสอบการทำงานของระบบ

ในการทดสอบการทำงานของระบบแบบสมบูรณ์นั้น จะทำการตรวจสอบข้อมูลที่ใช้ส่งและรับโดยอุปกรณ์กำเนิดสัญญาณแคนเพื่อเทียบข้อมูลระหว่างต้นทางและปลายทางว่ามีข้อมูลที่ถูกต้องตรงกัน แต่ต้องทำการปรับค่าตัวกำหนดรหัสเฉพาะหรือ ID ของข้อความแคนฝั่งรับด้วยโปรแกรมไมโครคอนโทรลเลอร์เพื่อให้แตกต่างและพิสูจน์ได้ว่าเป็นข้อมูลที่ได้รับมาจากโมดูล RF แตกต่างกันและโมดูลที่ใช้ทดสอบมีการทำงานที่สมบูรณ์ ในงานวิจัยนี้ได้ทำการการทดสอบการทำงานของระบบ 4 โมดูลพร้อมกันที่จุดเดียวและแบบหลายระยะทางดังนี้

1) ทำการทดสอบโมดูล 4 ชุด แบ่งเป็นภาคส่งข้อมูลที่ตัวที่ 1 และภาครับเป็นตัวที่ 2, 3 และ 4 ตามลำดับ โดยทำการทดสอบที่สภาพแวดล้อมแบบปิดไม่มีสัญญาณรบกวนและวางโมดูลไว้จุดเดียวกันดังภาพที่ 4.8

ภาพที่ 4.8 การทดสอบโมดูล 4 ชุดในระยะใกล้กัน

2) ทดสอบส่งข้อมูลแคน ให้กับโมดูลด้วยโปรแกรม PCAN-View และส่งค่าออกมาผ่านทาง PCAN-USB ให้กับโมดูลที่ใช้ทดสอบที่ตัวส่งดังแสดงในภาพที่ 4.9

Transmit

Message	Length	Data	Period	Count	Trigger
000h	8	00 00 00 00 00 00 00 00	Wait	1	Manual
001h	8	01 01 01 01 01 01 01 01	Wait	1	Manual
002h	8	02 02 02 02 02 02 02 02	Wait	1	Manual
003h	8	03 03 03 03 03 03 03 03	Wait	1	Manual

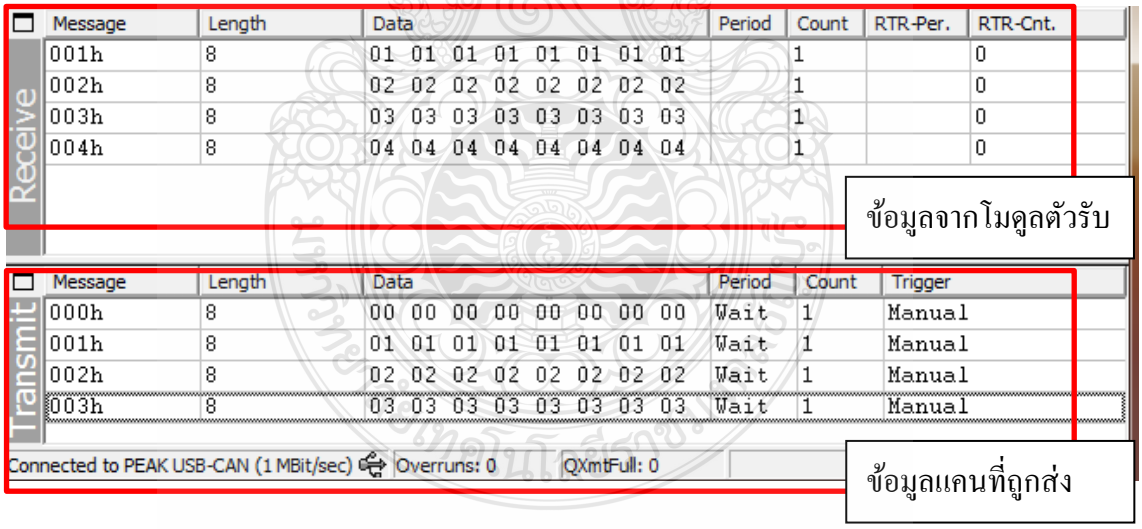
Connected to PEAK USB-CAN (1 MBit/sec) Overruns: 0 QXmtFull: 0

ภาพที่ 4.9 ทดสอบส่งข้อความแคน ให้กับโมดูล

3) วัดสัญญาณ SPI ระหว่างโมดูลว่ามีข้อมูลที่ถูกส่งออกมาและรับข้อมูลนั้นมีความถูกต้องตรงกัน ดังแสดงในภาพที่ 4.10

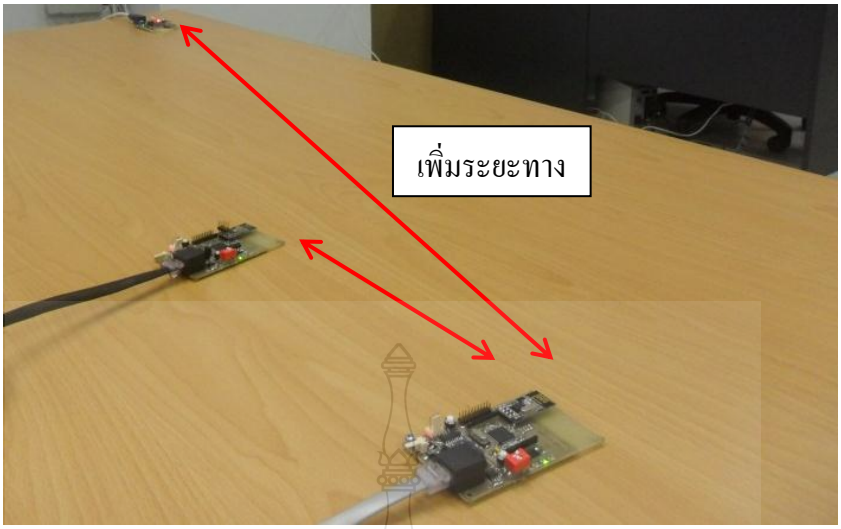
ภาพที่ 4.10 วัดสัญญาณ SPI ระหว่างการรับ-ส่งข้อมูลไร้สายทั้งโมดูลตัวส่งและโมดูลตัวรับ

4) อ่านค่าข้อมูลแคน ทั้งรับและส่งจากโปรแกรม จากภาพที่ 4.11 แสดงให้เห็นข้อมูลแคนที่ผ่านการรับ-ส่งแล้ว ซึ่งในขั้นตอนทดสอบในงานวิจัยนี้ได้กำหนดให้โมดูลที่รับข้อมูลต้องบวกค่าผลลัพธ์ให้มีค่าแตกต่างกันเพื่อเป็นการป้องกันและพิสูจน์ได้ว่ามีการรับข้อมูลได้จริง



ภาพที่ 4.11 ข้อความแคนที่ผ่านการรับ-ส่งแล้ว

5) นำอุปกรณ์ที่ทำการทดสอบวางไว้ห่างจากอุปกรณ์ตัวส่งเป็นระยะทางที่กำหนดในการทดสอบแต่ละครั้ง ก่อนเพิ่มระยะห่างระหว่างโมดูล จนถึงจุดที่สามารถรับสัญญาณได้ไกลสุดดังภาพที่ 4.12



ภาพที่ 4.12 ทดสอบโมดูลโดยเพิ่มระยะทางที่แตกต่างกัน

6) ทดสอบส่งข้อมูลและรับข้อมูลแคนที่มีระทางในการทดสอบต่างกัน เพื่อพิสูจน์ว่าโมดูลสามารถทำงานได้สมบูรณ์ที่ระยะทางแตกต่างกันหรือไม่ ดังแสดงผลในภาพที่ 4.13

Receive

Message	Length	Data	Period	Count	RTR-Per.	RTR-Cnt.
001h	8	01 01 01 01 01 01 01 01		1		0
002h	8	01 01 01 01 01 01 01 01		1		0
003h	8	01 01 01 01 01 01 01 01		1		0
004h	8	01 01 01 01 01 01 01 01		1		0

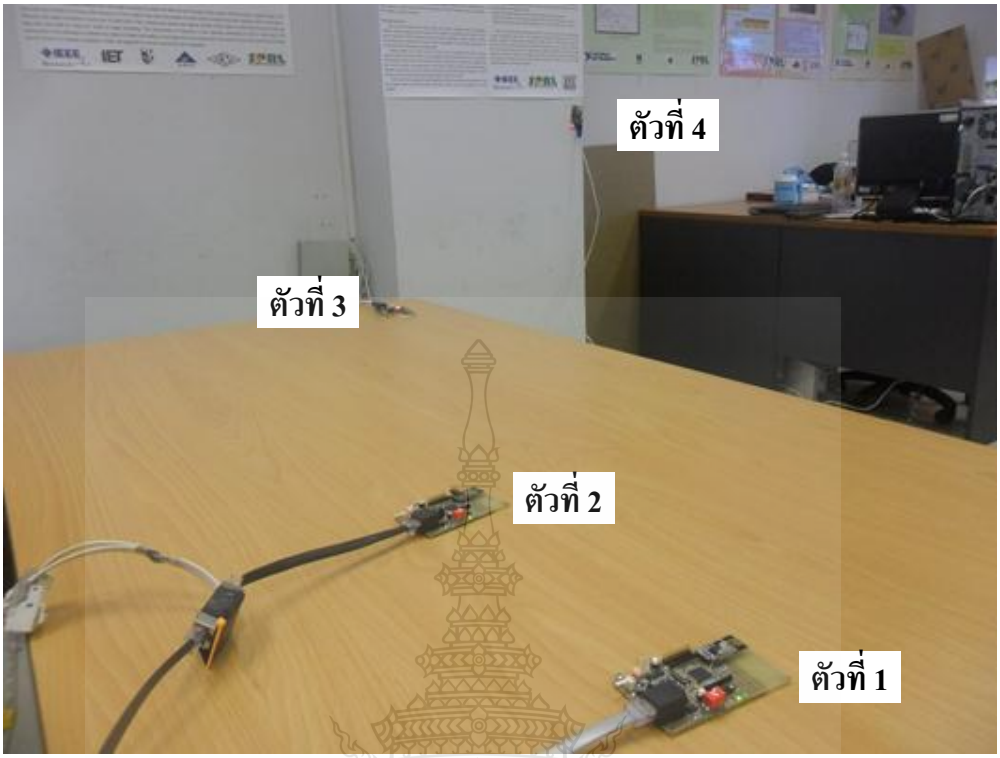
Transmit

Message	Length	Data	Period	Count	Trigger
000h	8	00 00 00 00 00 00 00 00	Wait	1	Manual
001h	8	00 00 00 00 00 00 00 00	Wait	1	Manual
002h	8	00 00 00 00 00 00 00 00	Wait	1	Manual
003h	8	00 00 00 00 00 00 00 00	Wait	1	Manual

Connected to PEAK USB-CAN (1 MBit/sec) Overruns: 0 QXmtFull: 0

ภาพที่ 4.13 ข้อความแคน ที่ผ่านการรับ-ส่งแล้วแบบระยะทางต่างกัน

7) กำหนดระยะทดสอบแบบคงที่เพื่อเปรียบเทียบข้อมูล และระยะทางที่โมดูลสามารถรับ-ส่งข้อมูลได้ เพื่อทดสอบกำลังการส่ง และรับข้อมูลระหว่างกันในการสื่อสารข้อมูลดังภาพที่ 4.14



ภาพที่ 4.14 ทดสอบโมดูลโดยกำหนดระยะทางที่แตกต่างกัน

8) ทดสอบส่งข้อมูลและรับข้อมูลแคนระหว่างโมดูลที่ถูกกำหนดระยะทาง เพื่อเปรียบเทียบและทดสอบว่าโมดูลสามารถทำงานได้สมบูรณ์ที่ระยะทางแตกต่างกัน ดังแสดงผลในภาพที่ 4.15

Receive

Message	Length	Data	Period	Count	RTR-Per.	RTR-Cnt.
001h	8	01 01 01 01 01 01 01 01		1		0
002h	8	02 02 02 02 02 02 02 02		1		0
003h	8	03 03 03 03 03 03 03 03		1		0
004h	8	04 04 04 04 04 04 04 04		1		0

Transmit

Message	Length	Data	Period	Count	Trigger
000h	8	00 00 00 00 00 00 00 00	Wait	1	Manual
001h	8	01 01 01 01 01 01 01 01	Wait	1	Manual
002h	8	02 02 02 02 02 02 02 02	Wait	1	Manual
003h	8	03 03 03 03 03 03 03 03	Wait	1	Manual

Connected to PEAK USB-CAN (1 MBit/sec) Overruns: 0 QXmtFull: 0

ภาพที่ 4.15 ข้อความแคนที่สามารถรับ-ส่งได้แบบระยะทางต่างกัน

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลงานวิจัย

5.1.1 การพัฒนาโดยใช้ไมโครคอนโทรลเลอร์ขนาด 32 บิต

จากการเปรียบเทียบชุดคำสั่งและทดสอบการทำงานระหว่างไมโครคอนโทรลเลอร์ขนาด 16 บิต และไมโครคอนโทรลเลอร์ขนาด 32 บิต สามารถสรุปผลการทดสอบได้ว่า ในการกระทำคำสั่งของโปรแกรมไมโครคอนโทรลเลอร์ขนาด 16 บิต ในการทำการประมวลผลข้อมูลนั้นช้ากว่าการประมวลผลของไมโครคอนโทรลเลอร์ขนาด 32 บิต ใน 1 สัญญาณนาฬิกาหรือไซเคิล (Cycle) เช่น คำสั่งให้ไมโครคอนโทรลเลอร์ขนาด 16 บิต บวกค่าข้อมูลขนาด 32 บิต นั้นต้องใช้ 2 สัญญาณนาฬิกา แต่ไมโครคอนโทรลเลอร์ขนาด 32 บิตนั้นสามารถทำการบวกค่าข้อมูลขนาด 32 บิต ได้ใน 1 สัญญาณนาฬิกา ซึ่งเวลาที่ใช้นั้นน้อยกว่าการทำงานของไมโครคอนโทรลเลอร์ขนาด 16 บิต 1 เท่า ทำให้การประมวลผลของไมโครคอนโทรลเลอร์มีความเร็วมากขึ้นกว่าเดิม

จากการพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์เพื่อใช้ในการทดสอบนั้น สามารถพัฒนาโปรแกรมการทำงานให้ไมโครคอนโทรลเลอร์สามารถทำงานด้วยคำสั่ง 32 บิตได้ ซึ่งทำให้การทำงานหรือการสื่อสารข้อมูลระหว่างโมดูลต่างๆ สามารถทำงานได้รวดเร็วและมีประสิทธิภาพมากขึ้นด้วย

5.1.2 การพัฒนาการรับ-ส่งข้อมูลแคนในรูปแบบไร้สาย

จากการทดสอบการรับ-ส่งข้อมูลแคนในรูบแบบไร้สาย ด้วยระยะทางที่แตกต่างกันพบว่าเมื่อเพิ่มการส่งข้อมูล และระยะทางในการรับ-ส่ง มากขึ้นก็จะเกิดปัญหาในเรื่องของความถูกต้องของข้อมูล และการเชื่อมต่อระหว่างโมดูล เหตุผลอันเนื่องมาจากขีดจำกัดของการรับ-ส่งข้อมูล ของโมดูล RF ที่มีกำลังในการส่งที่ 0dBm เมื่อมีระยะทางไกลขึ้นหรือมีสิ่งกีดขวางที่ทำให้ไม่สามารถติดต่อสื่อสารระหว่างกันได้ จะทำให้ไม่สามารถรับ-ส่งข้อมูลในขณะนั้นได้ หรืออาจจะทำให้ความถูกต้องของข้อมูลลดลงและเวลาที่ใช้ในการรับ-ส่งจนครบก็เพิ่มขึ้นจากการวนรอบรับข้อมูลเพื่อตรวจสอบความถูกต้อง ก่อนส่งข้อมูลออกมาที่แคนบัส ดังแสดงผลของของการเปรียบเทียบที่ระยะทางต่างกัน ดังตารางที่ 5.1 ดังนี้

ตารางที่ 5.1 เปรียบเทียบระยะทางและผลการรับ-ส่งข้อมูล

ระยะทาง เมตร	การสื่อสารระหว่างโมดูล	จำนวนการส่งข้อมูล/รอบ	จำนวนการรับข้อมูล/รอบ	ค่าความถูกต้องของข้อมูล	ค่าเวลาคลาดเคลื่อน
1	สมบูรณ์	100	100	100%	100%
2	สมบูรณ์	100	100	100%	100%
3	สมบูรณ์	100	100	100%	100%
4	สมบูรณ์	100	100	100%	100%
5	สมบูรณ์	100	100	100%	100%
6	สมบูรณ์	100	100	100%	100%
7	ขาดการสื่อสารบางครั้ง-น้อย	100	95	100%	95%
8	ขาดการสื่อสารบางครั้ง-น้อย	100	90	100%	90%
9	ขาดการสื่อสารบางครั้ง-น้อย	100	82	100%	82%
10	ขาดการสื่อสารบ่อยครั้ง	100	65	100%	65%

จากตารางที่ 5.1 สามารถสรุปได้ว่า ระยะทางของการรับ-ส่งข้อมูลนั้นสามารถรับส่งได้สูงสุดที่ 10 เมตร และยังคงความถูกต้องของข้อมูล 100% แต่ค่าเวลาคลาดเคลื่อนมีค่าสูงมาก ซึ่งจากผลการทดสอบ การรับ-ส่งข้อมูลระหว่างโมดูลจะเริ่มมีปัญหาในเรื่องการสื่อสารและรับ-ส่งข้อมูล เมื่อมีระยะห่างระหว่างโมดูลมากกว่า 6 เมตรขึ้นไป เนื่องจากกำลังส่งของโมดูล RF นั้นมีกำลังส่งน้อย เมื่อระยะทางมากขึ้นอาจเกิดปัญหาขณะสื่อสาร อาจทำให้การรอรับข้อมูลที่ถูกส่งมาขาดการสื่อสารในบางครั้ง และโมดูลต้องรอจนกว่าจะรับข้อมูลได้อีกครั้ง จึงทำให้เกิดค่าเวลาคลาดเคลื่อนเมื่อมีระยะทางในการรับ-ส่งข้อมูลมากขึ้น

5.2 แนวทางในการพัฒนาและปรับปรุงงานวิจัย

แนวทางในการพัฒนาและปรับปรุงงานวิจัย สามารถพัฒนาและปรับปรุงรูปแบบการสื่อสารและข้อผิดพลาดที่เกิดขึ้นในการรับ-ส่งสัญญาณ โดยเพิ่มความสามารถของเสาอากาศในการส่งสัญญาณ หรือเพิ่มกำลังส่งสัญญาณ RF ของโมดูลที่ 2.4GHz ให้มีกำลังส่งมากขึ้น เนื่องจากโมดูลไร้สายที่นำมาประยุกต์ใช้ มีกำลังส่งสัญญาณที่ 0dBm ซึ่งมีระยะส่งสัญญาณได้ไม่ไกล มาพัฒนาเสาอากาศหรือเพิ่มกำลังส่งสัญญาณ เพื่อช่วยให้มีระยะทางในการรับ-ส่งข้อมูลเพิ่มมากขึ้น รวมทั้งปรับปรุงระบบในการทำงานจากไมโครคอนโทรลเลอร์เป็น เอฟพีจีเอ หรือ อุปกรณ์ลอจิกแบบโปรแกรมได้ (FPGA) เพื่อให้มีประสิทธิภาพการทำงานที่มากขึ้น เนื่องจากเอฟพีจีเอมีความรวดเร็วในการทำงานสูง และช่วยให้การพัฒนาการสื่อสารแคนในรูปแบบไร้สายสามารถทำงานได้อย่างสมบูรณ์

รายการอ้างอิง

[1] นคร ภักดีชาติ,ชัยวัฒน์ ลิ้มพรจิตรวิไล "**ปฏิบัติการไมโครคอนโทรลเลอร์ ARM Cortex-M3 กับ STM32**" หน้า 7-25, 459-530.

[2] Reference Manual datasheet and application note "**RM0008 Reference manual STM32F101x and STM32F103xx advanced ARM-based 32-bit MCUs**" 2008 STMicroelectronics. pp 274-316

[3] Shyam Sadasivan "**An Introduction to the ARM Cortex-M3 Processor**" White paper ARM Limited. October 2006 pp 1-17.

[4] Mark Collier "**Running ARM7TDMI® Processor Software on the Cortex™-M3 Processor**" White paper ARM Limited. November 2006. pp 1-10.

[5] STM32F103 datasheet "**STM32F103x6 STM32F103x8 STM32F103xB Datasheet**" 2008 STMicroelectronics.

[6] ดร.วิรินทร์ เมฆประดิษฐ์สิน. **คำภีร์ตรวจซ่อมระบบเครือข่าย.**กรุงเทพฯ: ซีเอ็ดยูเคชั่น, 2550

[7] IEEE Harish Chincholi "Wireless Controller Area Network Based Cross Channel Data Link" Electronics & Communication Department, **Application of Information and Communication Technologies**, 2009, pp 1-5.

[8] IEEE Lin Hongju, Wang Haifang, Xiao Nianxin, Liu Chunxia, Chen Panfeng "Research on Coal Mine Personnel Positioning System Based on Zigbee and CAN" **New Trends in Information and Service Science,** 2009, pp 749 – 753.

[9] IEEE Mathias Johanson, Lennart Karlsson, Tore Risch "Relaying Controller Area Network Frames over Wireless Internetworks for Automotive Testing Applications" **Systems and Networks Communications**, 2009, pp 1-5.

[10] IEEE Luiz Alberto Castro de Almeida and Carlos Alberto dos Reis Filho "Latency Evaluation in a Bluetooth-CAN Dual Media Sensor Network" **Industrial Technology (ICIT)**, 2010, pp 247 – 251.

[11] IEEE Ai Chunli, Zhang Fengdeng, Liu Rongpeng "**Research on Wireless Backup for CAN in Process Control System**" RFID Eurasia, 2007 pp 1-6.

ภาคผนวก

ภาคผนวก ก

ซอร์ซโค้ด

1. main.c

```
#include "stm32f10x_rcc.h"
#include "stm32f10x_gpio.h"
#include "hardware_config.h"
#include "nRF24l01P.h"
#include "stm32f10x_can.h"
#include "stm32f10x.h"

/* Private variables ---------------------------------------------------------*/
u32        GlobalTime_ms;
//ErrorStatus HSEStartUpStatus;
u32        LifeSignTimeStamp;
char txBuffer[4];
char rxBuffer[4];
//CanTxMsg    TxCANData;
//CanRxMsg    RxCANData;

vu8 rcv_flag=0;        // Keep status receive data flag
CanRxMsg RxMessage;     // Keep CAN receive message
CanTxMsg TxMessage;     // Keep CAN transmit message
unsigned char Buf[2]= {0};

void GPIO_Configuration(void);
void CAN_Configuration(void);
void Peripheral_Configuration(void);
void OnInit(void);
void SysTick_Configuration(void);

int main(void)
{
   OnInit();

       nRF24L01_Set_RX_Address(0x01,0x23,0x45,0x67,0x89);
```

```
        nRF24L01_Config(0,P0dBm,R1mbps);
        RX_Mode();
    /*
     if(GPIO_ReadInputDataBit(GPIOC, SW1_Pin)==0)
       {
        RF_TestMode();
        Test_RF1();
       }
       //else {Test_CAN(); }
       RX_Mode();
     */

   while(1)
       {
       if(GPIO_ReadInputDataBit(GPIOC, SW2_Pin)==0)
        {
         if(nRF24L01_RxPacket(Buf))
         {
          if( Buf[0]==0x12 )
          {
            GPIO_SetBits(GPIOC, LED2_Pin);
            //delay(10);
          }
          else
          {
            GPIO_ResetBits(GPIOC, LED2_Pin);
          }
         }
        }
       }
}

void USB_LP_CAN_RX0_IRQHandler(void)
{
```

```
 unsigned char buf[20] = {0};
 unsigned char i1 = 0;
 unsigned char i2 = 0;
 typedef struct
{
      u32 StdId1;
      u32 ExtId1;
      u8 IDE1;
      u8 RTR1;
      u8 DLC1;
      u8 Data1[8];
      u8 FMI1;
    }RxMsgU;
RxMsgU  RxMessage1;
typedef union
 {
   unsigned char   buf2[20];
   RxMsgU  RxMessage1;

} U_can;
U_can        tmp2;
        //RxMsgU RxMessage1;
        RxMessage1.StdId1 =0x00;
        RxMessage1.ExtId1 = 0x00;
        RxMessage1.IDE1 = 0x00;
        RxMessage1.DLC1 = 0x00;
        RxMessage1.FMI1 = 0x00;
        RxMessage1.Data1[0] = 0x00;
        RxMessage1.Data1[1] = 0x00;
        RxMessage1.Data1[2] = 0x00;
        RxMessage1.Data1[3] = 0x00;
        RxMessage1.Data1[4] = 0x00;
        RxMessage1.Data1[5] = 0x00;
        RxMessage1.Data1[6] = 0x00;
```

```
    RxMessage1.Data1[7] = 0x00;
    CanRxMsg RxMessage;
    RxMessage.StdId=0x00;
    RxMessage.ExtId=0x00;
    RxMessage.IDE=0;
    RxMessage.DLC=0;
    RxMessage.FMI=0;
    RxMessage.Data[0]=0x00;
    RxMessage.Data[1]=0x00;
    RxMessage.Data[2]=0x00;
    RxMessage.Data[3]=0x00;
    RxMessage.Data[4]=0x00;
    RxMessage.Data[5]=0x00;
    RxMessage.Data[6]=0x00;
    RxMessage.Data[7]=0x00;
    CAN_Receive(CAN_FIFO0, &RxMessage);
    tmp2.RxMessage1.StdId1 = RxMessage.StdId;
    tmp2.RxMessage1.ExtId1 = RxMessage.ExtId;
    tmp2.RxMessage1.IDE1 = RxMessage.IDE;
    tmp2.RxMessage1.DLC1 = RxMessage.DLC;
    tmp2.RxMessage1.FMI1 = RxMessage.FMI;
    tmp2.RxMessage1.Data1[0] = RxMessage.Data[0];
    tmp2.RxMessage1.Data1[1] = RxMessage.Data[1];
    tmp2.RxMessage1.Data1[2] = RxMessage.Data[2];
    tmp2.RxMessage1.Data1[3] = RxMessage.Data[3];
    tmp2.RxMessage1.Data1[4] = RxMessage.Data[4];
    tmp2.RxMessage1.Data1[5] = RxMessage.Data[5];
    tmp2.RxMessage1.Data1[6] = RxMessage.Data[6];
    tmp2.RxMessage1.Data1[7] = RxMessage.Data[7];

if(RxMessage.StdId==0x11)
{

    nRF24L01_Set_RX_Address(0x01,0x23,0x45,0x67,0x89);
```

```
//nRF24L01_Set_TX_Address(0x01,0x23,0x45,0x67,0x89);
nRF24L01_Config(0,P0dBm,R1mbps);
TX_Mode();
for( unsigned char i4 = 0; i4 < 19; i4++ )
{
  tmp2.can.XORChkSum = buf[i4]^0x55;
}

for( unsigned char j = 0; j < 19; j++ )
{
   Val = buf[j]^tmp.can.XORChkSum;
}
if( Val == 0x55 )
buf[0] =  RxMessage.StdId;
buf[1] =  RxMessage.ExtId;
buf[2] =  RxMessage.IDE;
buf[3] =  RxMessage.DLC;
//buf[4] =  RxMessage.FMI;
RxMessage.Data[0]=0x00;
RxMessage.Data[1]=0x00;
RxMessage.Data[2]=0x00;
RxMessage.Data[3]=0x00;
RxMessage.Data[4]=0x00;
RxMessage.Data[5]=0x00;
RxMessage.Data[6]=0x00;
RxMessage.Data[7]=0x00;
        i2 = buf[3]+5;
 for(i1=0;i1<i2;i1++)
       buf[i1] = RxMessage.Data[i1];

i2 = RxMessage.DLC;

 for(i1=0;i1<i2;i1++)
       buf[i1] = RxMessage.Data[i1];
```

```
        nRF24L01_TxPacket(tmp2.buf2);
        RX_Mode();
        //delay_ms(100);
      }

}

void OnInit(void)
{
 RCC_Configuration();
 //SysTick_Configuration();

 Peripheral_Configuration();
 NVIC_Configuration();
}
void Peripheral_Configuration(void)
{
 nRF24L01_HW_Init();
 GPIO_Configuration();
 DMA_Configuration();
 //UART_ClkConfiguration();
 //UART_Configuration();
 CAN_Configuration();
}
/*
void SysTick_Configuration(void)
{
 SysTick_CounterCmd( SysTick_Counter_Disable );
 SysTick_CLKSourceConfig( SysTick_CLKSource_HCLK_Div8 );
 SysTick_CounterCmd( SysTick_Counter_Clear );
 SysTick_SetReload(TimerTickCycle_ms*9000);
 SysTick_ITConfig(ENABLE);
 SysTick_CounterCmd( SysTick_Counter_Enable );
}
```

```
*/
void delay_ms2(unsigned char x)
{
   //unsigned int x;
   unsigned int i4,j4;
   i4=0;
   for(i4=0;i4<x;i4++)
   {
     j4=108;
     while(j4--);
   }
}
```

2. nRF24L01.c

```
#include "stm32f10x_rcc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_spi.h"
#include "nRF24l01P.h"
//Define the commands for operate the nRF24L01P
#define READ_nRF_REG    0x00       // Command for read register
#define WRITE_nRF_REG   0x20       // Command for read register
#define RD_RX_PLOAD     0x61       // Command for read Rx payload
#define WR_TX_PLOAD     0xA0       // Command for write Tx payload
#define FLUSH_TX        0xE1       // Command for flush Tx FIFO
#define FLUSH_RX        0xE2       // Command for flush Rx FIFO
#define REUSE_TX_PL     0xE3       // Command for reuse Tx payload
#define NOP             0xFF    // Reserve
//Define the register address for nRF24L01P
#define CONFIG          0x00  //  Configurate the status of transceiver, mode of CRC and the replay of transceiver
status
#define EN_AA           0x01  //  Enable the atuo-ack in all channels
#define EN_RXADDR       0x02  //  Enable Rx Address
#define SETUP_AW        0x03  // Configurate the address width
#define SETUP_RETR      0x04  //  setup the retransmit
#define RF_CH           0x05  // Configurate the RF frequency
#define RF_SETUP        0x06  // Setup the rate of data, and transmit power
#define NRFRegSTATUS    0x07  //
#define OBSERVE_TX      0x08  //
#define CD              0x09  //    //Carrier detect
#define RX_ADDR_P0      0x0A  //  Receive address of channel 0
#define RX_ADDR_P1      0x0B  // Receive address of channel 1
#define RX_ADDR_P2      0x0C  // Receive address of channel 2
#define RX_ADDR_P3      0x0D  // Receive address of channel 3
#define RX_ADDR_P4      0x0E  // Receive address of channel 4
#define RX_ADDR_P5      0x0F  // Receive address of channel 5
#define TX_ADDR         0x10  //       Transmit address
```

```
#define RX_PW_P0       0x11  //  Size of receive data in channel 0
#define RX_PW_P1       0x12  //  Size of receive data in channel 1
#define RX_PW_P2       0x13  //  Size of receive data in channel 2
#define RX_PW_P3       0x14  //  Size of receive data in channel 3
#define RX_PW_P4       0x15  // Size of receive data in channel 4
#define RX_PW_P5       0x16  //  Size of receive data in channel 5
#define FIFO_STATUS    0x17  // FIFO Status
///*************************************************************************************
unsigned char TxBuf[Buffer_Size] = {0};
unsigned char RxBuf[Buffer_Size] = {0};
unsigned char nRF24L01_Freq = 0;
unsigned char nRF24L01_power_rate = 0;
//define the initial Address
unsigned char  TX_ADDRESS[ADR_WIDTH]= {0xE7,0xE7,0xE7,0xE7,0xE7};
unsigned char  RX_ADDRESS[ADR_WIDTH]= {0xE7,0xE7,0xE7,0xE7,0xE7};
unsigned char nRF24L01_SPI_Send_Byte(unsigned char dat);
void nRF24L01_HW_Init(void);
void nRF24L01_SPI_NSS_L(void);
void nRF24L01_SPI_NSS_H(void);
void nRF24L01_SPI_CE_L(void);
void nRF24L01_SPI_CE_H(void);
unsigned char SPI_WR_Reg(unsigned char reg, unsigned char value);
unsigned char SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char Len);
unsigned char SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char Len);
unsigned char SPI_RD_Reg(unsigned char reg);
void nRF24L01_Delay_us(unsigned long n);
void nRF24L01_Set_TX_Address(      unsigned char A,unsigned char B,unsigned char C,unsigned char
D,unsigned char E)
{
            TX_ADDRESS[0] = A;
            TX_ADDRESS[1] = B;
            TX_ADDRESS[2] = C;
            TX_ADDRESS[3] = D;
            TX_ADDRESS[4] = E;
```

```
}
void nRF24L01_Set_RX_Address(     unsigned char A,unsigned char B,unsigned char C,unsigned char
D,unsigned char E)
{
        RX_ADDRESS[0] = A;
        RX_ADDRESS[1] = B;
        RX_ADDRESS[2] = C;
        RX_ADDRESS[3] = D;
        RX_ADDRESS[4] = E;
}
unsigned char nRF24L01_Config(unsigned char freq, unsigned char power, unsigned char Rate)
{
      nRF24L01_Freq = 0;
        nRF24L01_power_rate = 0;
        if(freq>125)
                return 0;
        else
                nRF24L01_Freq = freq;
        if (P0dBm == power)
                nRF24L01_power_rate|=0x06;
        else if (Pm6dBm == power)
                nRF24L01_power_rate|=0x04;
        else if (Pm12dBm == power)
                nRF24L01_power_rate|=0x02;
        else if (Pm18dBm == power)
                nRF24L01_power_rate|=0x00;
        else
                return 0;
        if (R2mbps == Rate)
                {nRF24L01_power_rate|=0x08;}
        else if (Rate == R1mbps)
                {nRF24L01_power_rate|=0x00;}
        else if (Rate == R250kbps)
                nRF24L01_power_rate|=0x20;
```

```
        else
                return 0;
        return 1;
}

void RX_Mode(void)
{unsigned char buf[5]={0};
    nRF24L01_SPI_CE_L();
        SPI_Read_Buf(TX_ADDR, buf, ADR_WIDTH);
        SPI_Write_Buf(WRITE_nRF_REG + RX_ADDR_P0, RX_ADDRESS, ADR_WIDTH);
        SPI_WR_Reg(WRITE_nRF_REG + EN_AA, 0);
        SPI_WR_Reg(WRITE_nRF_REG + EN_RXADDR, 0x01);
        SPI_WR_Reg(WRITE_nRF_REG + SETUP_RETR, 0x1a);
        SPI_WR_Reg(WRITE_nRF_REG + RF_CH,nRF24L01_Freq);
        SPI_WR_Reg(WRITE_nRF_REG + RX_PW_P0, RX_PLOAD_WIDTH);
        SPI_WR_Reg(WRITE_nRF_REG + RF_SETUP, nRF24L01_power_rate);
        SPI_WR_Reg(WRITE_nRF_REG + CONFIG, 0x0f);
        nRF24L01_SPI_CE_H();
}
void TX_Mode(void)
{
    nRF24L01_SPI_CE_L();
    SPI_Write_Buf(WRITE_nRF_REG + TX_ADDR, TX_ADDRESS, ADR_WIDTH);
        SPI_Write_Buf(WRITE_nRF_REG + RX_ADDR_P0, RX_ADDRESS, ADR_WIDTH);
        SPI_WR_Reg(WRITE_nRF_REG + EN_AA, 0);
        SPI_WR_Reg(WRITE_nRF_REG + EN_RXADDR, 0x01);
        SPI_WR_Reg(WRITE_nRF_REG + SETUP_RETR, 0x1a);
        SPI_WR_Reg(WRITE_nRF_REG + RF_CH,nRF24L01_Freq);
        //SPI_WR_Reg(WRITE_nRF_REG + RF_CH,40);
        SPI_WR_Reg(WRITE_nRF_REG + RF_SETUP,  nRF24L01_power_rate);
        //SPI_WR_Reg(WRITE_nRF_REG + RF_SETUP,  0x0f);
        SPI_WR_Reg(WRITE_nRF_REG + CONFIG, 0x0e);
    nRF24L01_SPI_CE_H();
}
```

```
void nRF24L01_TxPacket(unsigned char * tx_buf)
{
        SPI_Write_Buf(WRITE_nRF_REG + RX_ADDR_P0, TX_ADDRESS, ADR_WIDTH);
        SPI_Write_Buf(WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH);
}
unsigned char nRF24L01_RxPacket(unsigned char* rx_buf)
{unsigned char flag=0;
 unsigned char status;
        status=SPI_RD_Reg(NRFRegSTATUS);
        if(status & 0x40)
        {
                SPI_Read_Buf(RD_RX_PLOAD,rx_buf,TX_PLOAD_WIDTH);
                flag =1;
        }
        SPI_WR_Reg(WRITE_nRF_REG+NRFRegSTATUS, status);
        return flag;
}
unsigned char SPI_RD_Reg(unsigned char reg)
{
        unsigned char reg_val;
        nRF24L01_SPI_NSS_L();
        nRF24L01_SPI_Send_Byte(reg);
        reg_val = nRF24L01_SPI_Send_Byte(0);
        nRF24L01_SPI_NSS_H();
        return(reg_val);
}
unsigned char SPI_WR_Reg(unsigned char reg, unsigned char value)
{
        unsigned char status;
      nRF24L01_SPI_NSS_L();
        status = nRF24L01_SPI_Send_Byte(reg);
        nRF24L01_SPI_Send_Byte(value);
        nRF24L01_SPI_NSS_H();
        return(status);
```

```
}
unsigned char SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char Len)
{
        unsigned int status,i;
        nRF24L01_SPI_NSS_L();
        status = nRF24L01_SPI_Send_Byte(reg);
 for(i=0;i<Len;i++)
 {
   pBuf[i] = nRF24L01_SPI_Send_Byte(0);
 }
        nRF24L01_SPI_NSS_H();
        return(status);
}

unsigned char SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char Len)
{
        unsigned int status,i;
        nRF24L01_SPI_NSS_L();
        status = nRF24L01_SPI_Send_Byte(reg);
        for(i=0; i<Len; i++) //
        {
                nRF24L01_SPI_Send_Byte(*pBuf);
                 pBuf ++;
        }
        nRF24L01_SPI_NSS_H();
        return(status);
}
unsigned char nRF24L01_SPI_Send_Byte(unsigned char dat)
{
 while(SPI_I2S_GetFlagStatus(SPI2, SPI_I2S_FLAG_TXE) == RESET);
 SPI_I2S_SendData(SPI2, dat);
 while(SPI_I2S_GetFlagStatus(SPI2, SPI_I2S_FLAG_RXNE) == RESET);
 return SPI_I2S_ReceiveData(SPI2);
}
```

```
void nRF24L01_SPI_NSS_H(void)
{
	GPIO_SetBits(GPIOB,GPIO_Pin_12);
}

void nRF24L01_SPI_NSS_L(void)
{
	GPIO_ResetBits(GPIOB,GPIO_Pin_12);
}

void nRF24L01_SPI_CE_H(void)
{
	GPIO_SetBits(GPIOB,GPIO_Pin_10);
}
void nRF24L01_SPI_CE_L(void)
{
	GPIO_ResetBits(GPIOB,GPIO_Pin_10);
}
void nRF24L01_HW_Init(void)
{
	SPI_InitTypeDef  SPI_InitStructure;
	GPIO_InitTypeDef GPIO_InitStructure;
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_SPI2, ENABLE);
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
	/* Configure SPI2 pins: NSS, SCK, MISO and MOSI */
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
	GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;		//NRF24L01  MODE-CE
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;		//NRF24L01  IRQ
```

```
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_SetBits(GPIOB, GPIO_Pin_11);

        SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
        SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
        SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
        SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;ดอ
        SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge;
        SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
        SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_16;
        SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
        SPI_InitStructure.SPI_CRCPolynomial = 7;
        SPI_Init(SPI2, &SPI_InitStructure);
        //Config the NSS pin
        SPI_SSOutputCmd(SPI2, ENABLE);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_Init(GPIOB, &GPIO_InitStructure);
        /* Enable SPI2  */
        SPI_Cmd(SPI2, ENABLE);
    nRF24L01_SPI_NSS_H();
    nRF24L01_SPI_CE_H();
}
void nRF24L01_Delay_us(unsigned long n)
{
        unsigned long i;
        while(n--)
        {
          i=2;
          while(i--);
  }
}
```

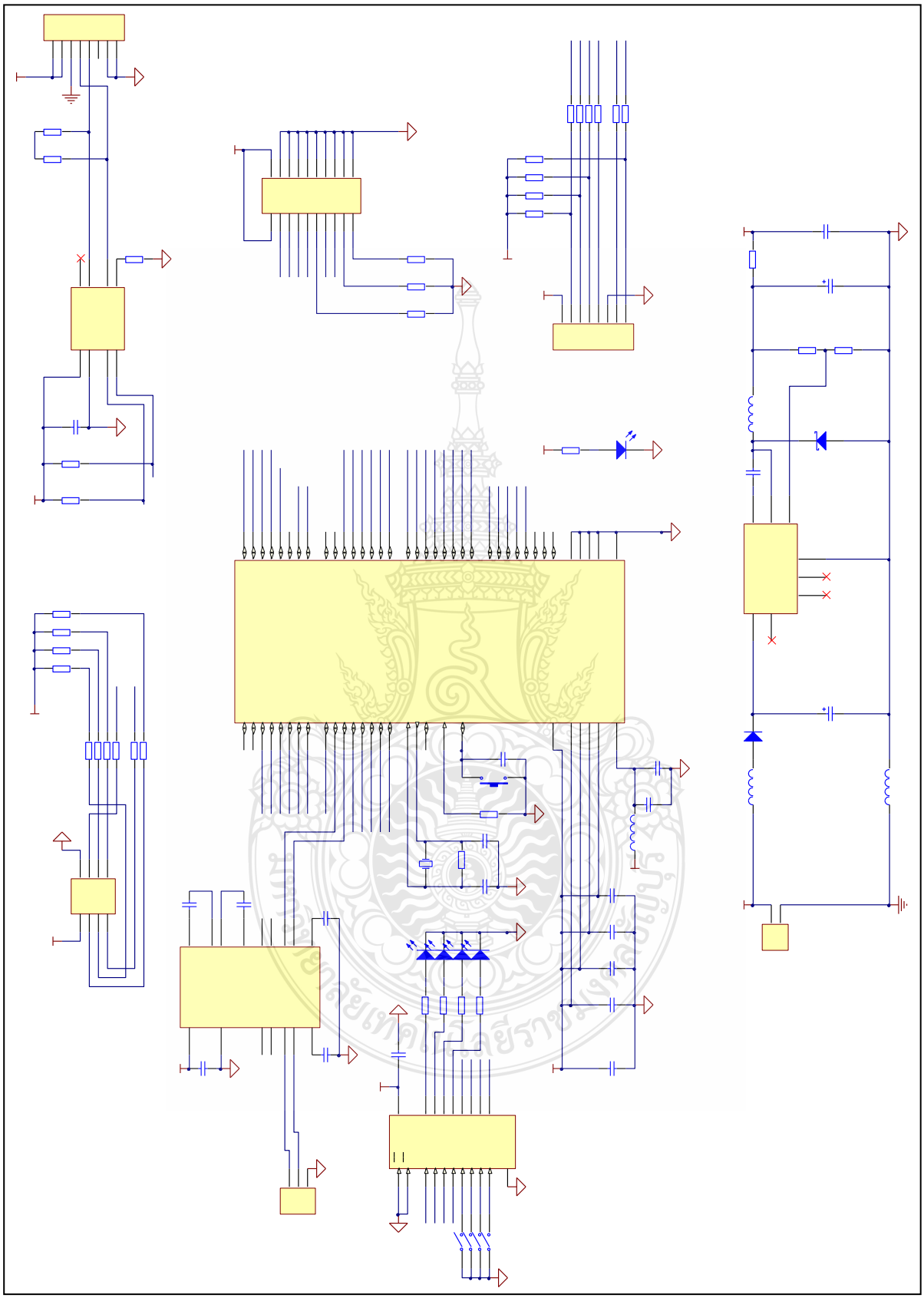
3. can_stm.c

```
#include "stm32f10x.h"
#include "stm32f10x_lib.h"
#include "stm32f10x_can.h"
void CAN_Configuration(void)
{
 CAN_InitTypeDef       CAN_InitStructure;
 CAN_FilterInitTypeDef  CAN_FilterInitStructure;
 /* CAN register init */
 CAN_DeInit();
 CAN_StructInit(&CAN_InitStructure);
 /* CAN cell init */
  CAN_InitStructure.CAN_TTCM=DISABLE;
  CAN_InitStructure.CAN_ABOM=DISABLE;
  CAN_InitStructure.CAN_AWUM=DISABLE;
  CAN_InitStructure.CAN_NART=DISABLE;
  CAN_InitStructure.CAN_RFLM=DISABLE;
  CAN_InitStructure.CAN_TXFP=DISABLE;
  //CAN_InitStructure.CAN_Mode=CAN_Mode_LoopBack;
  CAN_InitStructure.CAN_Mode=CAN_Mode_Normal;
  /*
  CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
  CAN_InitStructure.CAN_BS1=CAN_BS1_8tq;
  CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
  CAN_InitStructure.CAN_Prescaler=1;
  */

  //CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
  //CAN_InitStructure.CAN_BS1=CAN_BS1_8tq;
  //CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
  //CAN_InitStructure.CAN_Prescaler=5;       //72MHz/2=36MHz=PCLK1 / 5 => 7200KHz / (1+8+7) =>
450KHz
  /*
```

```
  CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
  CAN_InitStructure.CAN_BS1=CAN_BS1_10tq;
  CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
  CAN_InitStructure.CAN_Prescaler=4;        //72MHz/2=36MHz=PCLK1 / 4 => 9000KHz / (1+10+7) =>
500KHz
  */
  CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
  CAN_InitStructure.CAN_BS1=CAN_BS1_10tq;
  CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
  CAN_InitStructure.CAN_Prescaler=2;       //72MHz/2=36MHz=PCLK1 / 2 => 18000KHz / (1+10+7) =>
1000KHz
  CAN_Init(&CAN_InitStructure);
  // CAN filter setup
  CAN_FilterInitStructure.CAN_FilterNumber=1;
  CAN_FilterInitStructure.CAN_FilterMode=CAN_FilterMode_IdMask;
  CAN_FilterInitStructure.CAN_FilterScale=CAN_FilterScale_32bit;
  CAN_FilterInitStructure.CAN_FilterIdHigh=0x0000;
  CAN_FilterInitStructure.CAN_FilterIdLow=0x0000;
  CAN_FilterInitStructure.CAN_FilterMaskIdHigh=0x0000;
  CAN_FilterInitStructure.CAN_FilterMaskIdLow=0x0000;
  CAN_FilterInitStructure.CAN_FilterFIFOAssignment=CAN_FIFO0;
  CAN_FilterInitStructure.CAN_FilterActivation=ENABLE;
  CAN_FilterInit(&CAN_FilterInitStructure);

  CAN_ITConfig(CAN_IT_FMP0,ENABLE);
}
```

ภาคผนวก ข

วงจรสมบูรณ์



ภาคผนวก ค

ผลงานตีพิมพ์เผยแพร่

1. บทความเรื่อง “Development Control Area Network communication using 2.4GHz Radio Frequency”,การประชุมวิชาการ งานวิจัยและพัฒนาเชิงประยุกต์ ครั้งที่ 3 (ECTI- CARD), 2554. โรงแรมอมารี ดอนเมือง แอร์พอร์ต ระหว่างวันที่ 5-6 พฤษภาคม 2554

2. บทความเรื่อง “Development of 1Mbps RF-CAN Communication Multi-node”, The Third International Conference on Information and Communication Technology for Embedded Systems (ICICTES2012) VIE Hotel, Bangkok, Thailand ระหว่างวันที่ 22-24 มีนาคม 2555.


ECTI-CARD 2011
การประชุมวิชาการ งานวิจัย และพัฒนาเชิงประยุกต์ ครั้งที่ 3
- เทคโนโลยีชีวภาพ การแพทย์ วิทยาศาสตร์กายภาพ วิทยาศาสตร์การกีฬา
- ระบบรักษาความปลอดภัย การควบคุมการเข้าถึง การยืนยันตัวตน ระบบตรวจจับ
- การสื่อสาร การสนับสนุนผู้ใช้ตามบ้าน เครือข่ายสังคม เครือข่ายไร้สาย
- การเรียนรู้การสอนทางไกล การศึกษาบันเทิง คอมพิวเตอร์เอนิเมชั่น
- การประหยัดพลังงาน การจัดการพลังงานบ้านอัตโนมัติ
- การขนส่ง การควบคุมจราจร การจัดการอุตสาหกรรม
- ธุรกิจการธนาคาร การท่องเที่ยว และการโรงแรม
- การกู้ภัย ระบบเตือนภัย และการพยากรณ์
- เกษตรกรรม อุตสาหกรรมเกษตร
RSU
มหาวิทยาลัยรังสิต
RANGSIT UNIVERSITY
ECTI
Association
หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
มหาวิทยาลัยรังสิต
จัดทำและดำเนินงานโดย
วิทยาลัยวิศวกรรมศาสตร์
มหาวิทยาลัยรังสิต

รายชื่อผู้พิจารณาบทความ

ผู้พิจารณาบทความ

ดร.สมบูรณ์	ศุขสาตร	มหาวิทยาลัยรังสิต
รศ.ดร.โอภาส	จุฑาเทพ	มหาวิทยาลัยรังสิต
ดร.ไพศาล	งามจรรยาภรณ์	มหาวิทยาลัยรังสิต
รศ.วรศักดิ์	นิรัคฆนาภรณ์	มหาวิทยาลัยรังสิต
ผศ.ดร.รัชชัย	แสงอุดม	มหาวิทยาลัยรังสิต
อ.สราวุธ	จันทร์ผง	มหาวิทยาลัยรังสิต
Dr.Jin-man	Yang	มหาวิทยาลัยรังสิต
ผศ.ดร.ดวงอาทิตย์	ศรีมูล	มหาวิทยาลัยรังสิต
ดร.สุวรรณ	จันทร์อินทร์	มหาวิทยาลัยรังสิต
อ.เอนก	กนกอภิวัฒน์	มหาวิทยาลัยรังสิต
อ.พหล	สมบูรณ์ธรรม	มหาวิทยาลัยรังสิต
อ.สมหมาย	บัวแย้มแสง	มหาวิทยาลัยรังสิต
อ.อภิรักษ์	ภักดีวงษ์	มหาวิทยาลัยรังสิต
อ.จตุพล	ศรีวิลาศ	มหาวิทยาลัยรังสิต
ดร.ธรรมนูญ	สุสำเภา	มหาวิทยาลัยรังสิต
ผศ.วิศิษฐ์	อยู่ยงวัฒนา	มหาวิทยาลัยรังสิต
รศ.สมาน	เสนงาม	มหาวิทยาลัยรังสิต
ดร.วรรณี	เอกศิลป์	มหาวิทยาลัยรังสิต
อ.จตุพร	สถากุลเจริญ	มหาวิทยาลัยรังสิต
ผศ.ดร.สานนท์	ฉิมมณี	มหาวิทยาลัยรังสิต
อ.สุมนา	เกษมสวัสดิ์	มหาวิทยาลัยรังสิต
ดร.วฤต	ศิลป์ศรีกุล	มหาวิทยาลัยนอร์ท-เชียงใหม่
ดร.สุรชัย	สานติสุขรัตน์	มหาวิทยาลัยนอร์ท-เชียงใหม่
ดร.คมศักดิ์	เมฆสมุทร	มหาวิทยาลัยเชียงใหม่
ผศ.ดร.พีรวัฒน์	วัฒนพงศ์	มหาวิทยาลัยเกษตรศาสตร์
ผศ.ดร.ชัยพร	ใจแก้ว	มหาวิทยาลัยเกษตรศาสตร์
รศ.ดร.ฐิติวรรณ	ศรีนาค	มหาวิทยาลัยเกษตรศาสตร์
ดร.มนต์ชัย	โสพิศกมล	มหาวิทยาลัยเกษตรศาสตร์
รศ.ดร.อนันต์	ผลเพิ่ม	มหาวิทยาลัยเกษตรศาสตร์
อ.อภิรักษ์	จันทร์สร้าง	มหาวิทยาลัยเกษตรศาสตร์
นอ.ศ.ดร.สรกฤช	ศรีเกษม	โรงเรียนนายเรืออากาศ
ผศ.ดร.ชูวงค์	พงศ์เจริญพาณิชย์	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ดร.เอกรัฐ	บุญภูงา	มหาวิทยาลัยเทคโนโลยีมหานคร
ดร.จักรกฤช	ดรรกพาณิชย์	มหาวิทยาลัยเทคโนโลยีมหานคร

รายชื่อผู้พิจารณาบทความ

ผู้พิจารณาบทความ

ดร.โชคชัย	แสงดาว	มหาวิทยาลัยเทคโนโลยีมหานคร
ดร.สมมาตร	แสงเงิน	มหาวิทยาลัยเทคโนโลยีมหานคร
ดร.ธีรวิศิฏฐ์	เลาหะเพ็ญแสง	มหาวิทยาลัยเทคโนโลยีมหานคร
ดร.วรพล	ลีลาเกียรติสกุล	มหาวิทยาลัยเทคโนโลยีมหานคร
ดร.ศุภกร	กังพิศดาร	มหาวิทยาลัยเทคโนโลยีมหานคร
อ.อนุรี	หล่อสวัสศิริ	มหาวิทยาลัยเทคโนโลยีมหานคร
อ.ธนทรรศน์	แซ่ลิ้ม	มหาวิทยาลัยเทคโนโลยีมหานคร
อ.วุฒิพร	เลิศวาสนา	มหาวิทยาลัยเทคโนโลยีมหานคร
อ.วินัย	ศิลารวม	มหาวิทยาลัยเทคโนโลยีมหานคร
ผศ.ดร.ชาญชัย	ทองโสภา	มหาวิทยาลัยเทคโนโลยีสุรนารี
ดร.จิรัฐภู	เหมือนชู	มหาวิทยาลัยเอเชียน
ดร.สุธาสินี	ละมุนตรี	มหาวิทยาลัยเอเชียน
ผศ.ดร.พรรณฤมล	เต็มดี	มหาวิทยาลัยแม่ฟ้าหลวง
ผศ.ดร.รังสรรค์	ชัยศรีเจริญ	มหาวิทยาลัยแม่ฟ้าหลวง
ดร.ศราวุธ	ชัยมูล	มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ดร.ณรงค์ฤทธิ	วราภรณ์	มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
ดร.ภัควัฒน์	จันทร์ตรี	มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ

ระบบกระจายข้อมูลโทรทัศน์การศึกษาทางไกลผ่านดาวเทียมทางอินเตอร์เน็ต
ก่อเกียรติ วิริยะสมบัติ, สุนทร วิทูสุรพจน์ 42
การเพิ่มประสิทธิภาพบริการเสริม Joomla! CMS เพื่องานส่งการบ้านผ่านเว็บไซต์
ดนัยศักดิ์ กาโร 47
9 ช่องมหัศจรรย์
ชยพร ศุภวิไล, ณัฐพงศ์ สุระเสถียร, สุขุมาล กิติสิน, ชาคริต วัชโรภาส 53
เกมผจญภัยในโลกแมลง
ฐิติชญา เห็นพร้อม, นูรุดดีน มะนอ, ลัดดา ปรีชาวีรกุล 59
โมดูลเพิ่มเติมในมูเด้ิล เพื่อวิเคราะห์พฤติกรรมการเรียนและตรวจหานักเรียนที่มีพฤติกรรมเสี่ยง
สมนึก พ่วงพรพิทักษ์, อุมาพร จันโสภา, จิตลัดดา เนืองนิตย์ 65
การพัฒนาระบบวิดีโอสตรีมมิ่งแบบกลุ่มเมฆด้วยเทคโนโลยีเพียร์ทูเพียร์
รวิภาส อิ่งตระกูล, ภุชงค์ อุทโยภาศ 71
การเรียนรู้ในพิพิธภัณฑ์ทางมานุษยวิทยาผ่านระบบ 4 มิติแบบออนไลน์
หัสดี พิมพ์สุวรรณ, จิติมนต์ อั่งสกุล, ธรา อั่งสกุล 76
เกมออนไลน์แบบหลายผู้เล่นในระบบสามมิติบนโทรศัพท์เคลื่อนที่
วสันต์ ศรีธัญญลักษณา, ภูผา สังข์สาย, ศิริทรัพย์ อำนาจศิริกุล, ฐิติวรรณ ศรีนาค 82
World of Creativity 3D : A Fun Approach to Learn Java Programming
Keeratipong Ukachoke, Thitiwan Srinark 88

กลุ่มที่ 6 : การสื่อสาร การสนับสนุนผู้ใช้ตามบ้าน เครือข่ายสังคม เครือข่ายไร้สาย

AREA 6 : Telecommunication, Home Support, Social Networks, Mobile Network

Multiband C-Shaped Printed Antenna for Wireless Communication Systems
Punyawi Jamjareekul 94
ระบบบริหารจัดการและวิเคราะห์การใช้งานโทรศัพท์ผ่าน VoIP
อนุวัตร สมบุญ, บุญชัย งามวงศ์วัฒนา 100
เครื่องต้นแบบรบกวนสัญญาณโทรศัพท์เคลื่อนที่
ธนัสถ์ นนทพุทธ, ขจรศักดิ์ พงศ์ธนา, วิชาญ เพชรมณี 106
Development Control Area Network communication using 2.4GHz Radio Frequency
วุฒิชัย บัวนาค, จักรี ศรีนนท์ฉัตร 111
ระบบทดสอบประสิทธิภาพเครือข่ายโทรศัพท์เคลื่อนที่
ภาธร เต็งเกียรติ์ตระกูล, ปรัชญา ชิตชูสกุล, สุขุมาล กิติสิน 117
ระบบประมวลผลคำตอบปรนัยแบบอัตโนมัติผ่านระบบสื่อสารแบบไร้สาย 2.4 GHz
ขจรศักดิ์ พงศ์ธนา, วิชาญ เพชรมณี, ธนัสถ์ นนทพุทธ 123
การพัฒนาสายอากาศความถี่กว้างมากแบบไมโครสตริปด้วยการบากทำมุมที่ระนาบกราวด์
ระพีพันธ์ แก้วอ่อน 129

© ECTI-CARD 2011, May' 5-6, Bangkok, Thailand. ISBN: 978-974-350-301-6

Development Control Area Network communication using 2.4GHz Radio Frequency

วุฒิชัย บัวนาค
ห้องปฏิบัติการและวิจัยทางด้านสัญญาณ
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์และโทรคมมาคม
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
wuttichai@incotec-automation.com

จักรี ศรีนนท์ฉัตร
ห้องปฏิบัติการและวิจัยทางด้านสัญญาณ
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์และโทรคมมาคม
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
Jakkree.s@en.rmutt.ac.th

บทคัดย่อ

Control Area Network (CAN) เป็นรูปแบบการสื่อสารข้อมูลในรูปแบบ BUS มีการตอบสนองการรับ-ส่งข้อมูลแบบ real-time มีความถูกต้องของข้อมูลสูงและนำมาใช้ในงานควบคุมเครื่องจักรอุตสาหกรรมและในระบบฝังตัว (Embedded Controller) แต่ด้วยการทำงานที่ต้องอยู่ร่วมกันบนบัสทำให้เกิดข้อจำกัดการเชื่อมต่อในกรณีที่ต้องมีการทำงานในรูปแบบที่ต้องมีการเคลื่อนที่อิสระ บทความนี้ได้นำเสนอถึงวิธีการเปลี่ยนรูปแบบการรับส่งข้อมูลแบบBUSให้มาอยู่ในรูปแบบไร้สายแทนการรับส่งแบบเดิม โดยใช้ไมโครคอนโทรเลอร์ขนาด 32Bit เบอร์ STM32F103VCT ซึ่งเป็นไมโครคอนโทรเลอร์ประสิทธิภาพสูงในตระกูล ARM Cortex™-M3 ไมโครคอนโทรเลอร์ทำหน้าที่ในการประมวลผลของสัญญาณเพื่อใช้ในการสื่อสาร รับ-ส่ง ข้อมูลกับโมดูลไร้สายที่ย่านความถี่ 2.4GHz พบว่าสามารถนำข้อมูลในรูปแบบ CAN มาเปลี่ยนแปลงรูปแบบในการสื่อสารเพื่อรับ-ส่งข้อมูลแบบไร้สาย โดยที่ข้อมูลยังคงความถูกต้องของข้อมูลเดิมไว้หลังจากที่ได้รับข้อมูลแล้ว พบว่าความเร็วในการรับส่งข้อมูลสามารถส่งได้สูงสุดที่ไม่เกิน 500 kbps ดังนั้นการศึกษาการพัฒนารูปแบบการรับส่งข้อมูล CAN ในรูปแบบไร้สายนั้น มีความน่าสนใจในการพัฒนาระบบนี้สำหรับระบบสมองกลฝังตัวในอนาคต

Abstract

Controller Area Network (CAN) protocol is a real time transmitted-received BUS communication and is popular in industrial automation and embedded systems networked applications. However CAN is designed for working on BUS communication system with limited connectivity. This paper presents a solution data communication from CAN BUS to Radio Frequency using 32 Bit Microcontroller STM32F103VCT which is a high-performance ARM Cortex™-M3, Microcontroller processes data communication between CAN and Radio Frequency module working on 2.4GHz. The results show that it can process data from CAN to Radio frequency and provide the 100% accuracy when the transmission rate is limited, not more than 500kbps. Thus, the development of CAN wireless data communication is interesting for the future embedded system.

คำสำคัญ

Controller Area Network, Radio Frequency, Microcontroller, ARM Cortex™-M3.

1. บทนำ

จากการที่ได้ศึกษาข้อมูลงานวิจัยที่เกี่ยวข้อง [1] [2] พบว่าการนำการสื่อสารในรูปแบบ CAN เป็นแบบไร้สายได้มีผู้พัฒนาได้ทำการทดสอบไว้แต่การเปลี่ยนรูปแบบมาเป็นไร้สายนั้นมีข้อจำกัดอยู่มาก เพราะไม่สามารถนำการการสื่อสารแบบ CAN มาแปลงเป็นไร้สายได้โดยตรงหรือโดยการแปลงให้อยู่ในรูปแบบอื่นก่อนแล้วส่งข้อมูลในรูปแบบไร้สายเพราะว่าเนื่องด้วยการออกแบบการสื่อสารของ CAN นั้นถูกออกแบบให้อยู่ในรูปแบของ BUS จึงเกิดปัญหาเวลามีการรับส่งสัญญาณพร้อมกันหลายจุดเพราะไม่สามารถรับประกันได้ว่าข้อมูลนั้นจะส่งจากจุดใดไปจุดใด จากข้อมูลที่ได้ศึกษามานั้น [3] การแปลงการสื่อสารของ CAN มาเป็นมาเป็นรูปแบบไร้สายสามารถทำได้แต่ต้องรับ-ส่งแบบจุดต่อจุดหรือตัวใดตัวหนึ่งเป็นตัวส่งและอีกตัวเป็นตัวรับ ซึ่งทำให้สูญเสียคุณลักษณะที่ดีของ

© ECTI-CARD 2011, May' 5-6, Bangkok, Thailand. ISBN: 978-974-350-301-6

ระบบ CAN และความเร็วในการทำงานของระบบนั้นก็ถูกลดทอนลงไป กล่าวคือ ความสามารถในการรับส่งของ CAN นั้นสามารถรับส่งได้ถึง 1M แต่จากที่ได้ศึกษามาสามารถรับส่งได้ไม่เกิน 500 kbps ซึ่ง เป็นแค่มาตรฐานการเชื่อมต่อเท่านั้น

ในงานวิจัยนี้ ได้นำเสนอการพัฒนารูปแบบและการออกแบบเพื่อสร้างอุปรณ์เพื่อทดสอบการทำงานโดยใช้ไมโครคอนโทรเลอร์ขนาด 32 Bit แบบ ARM Cortex™-M3 และโมดูล RF 2.4GHz ในการทดสอบการทำงานของงานวิจัย โดยมุ่งเน้นเพื่อนำผลของการพัฒนางานวิจัยมาใช้งานได้ และสามารถนำไปพัฒนาต่อเพิ่มเติมได้อีกต่อไป

2. ที่มาและแรงจูงใจของปัญหา

ข้อจำกัดของการสื่อสารข้อมูลแบบ CAN นั้นโดยหลักแล้วจะทำงานในรูปแบบ BUS ถึงแม้จะมีความยาวและมีการเชื่อมต่อโหนดหลายตัวแต่ถ้ามีอุปกรณ์บางชนิดนั้นต้องมีการเคลื่อนที่อยู่ตลอดการเชื่อมต่อในรูปแบบนี้ก็จะไม่สามารถตอบสนองความต้องการได้ ดังนั้นการพัฒนารูปแบบการสื่อสารข้อมูลในรูปแบบของ CAN นั้นจะช่วยให้สามารถทำการเชื่อมต่ออุปกรณ์ที่ใช้ได้โดยที่ไม่ต้องมีการเปลี่ยนแปลงรูปแบบการสื่อสารเดิมไปเป็นรูปแบบอื่นซึ่งเป็นการเพิ่มภาระให้กับการออกแบบโปรแกรมหรือระบบหลักให้ครอบคลุมอุปกรณ์ที่ต้องใช้รูปแบบการสื่อสารแบบไร้สาย จาก Wireless CAN ที่ได้ศึกษานั้นไม่สามารถรับส่งในกรณีที่มีปริมาณของข้อมูลและมีโหนดที่ใช้ในการเชื่อมต่อหลายจุด [2] [3] จากปัญหาการรับส่งข้อมูลโดยใช้ ไมโครคอนโทรเลอร์ที่มีการประมวลผลที่ 8 bit ทำให้เกิดการสูญเสียของข้อมูลสูงและไม่สามารถทำงานในกรณีที่มีการรับส่งปริมาณมาก จึงเกิดแนวคิดในการพัฒนารูปแบบการทำงานโดยใช้ไมโครคอนโทรเลอร์ที่มีการประมวลผลที่ 32 bit เพื่อที่จะทำให้การรับส่งมีโครงสร้างและรูปแบบการทำงานที่ดีขึ้นในการจัดวางรูปแบบการรับ-ส่งข้อมูลเพื่อให้เกิดการสูญเสียที่น้อยลงและความเร็วในการรับ-ส่งระหว่างมีมากขึ้น

3. งานและทฤษฎีที่เกี่ยวข้อง

ในงานวิจัยนี้ได้นำเสนอข้อมูลเกี่ยวกับสถาปัตยกรรมของอุปกรณ์บัส CAN[4] โดย CAN มีสถาปัตยกรรมที่เข้ากันได้ตามมาตรฐาน ISO 11898 ซึ่งว่าด้วยการถ่ายทอดข้อมูลระหว่างอุปกรณ์ในระบบโครงข่ายที่สอดคล้องตามโมเดล OSI (Open System Interconnection(ในบัส CAN ข้อความจะสามารถส่งออกมาจากทุกโนด และทุกโนดสามารถเข้าถึงเพื่อรับข้อมูลนั้นได้ในบัส CAN ไม่มีการกำหนดแอดเดรสทั้งโนดส่งและโหนดรับ รายละเอียดของข้อความจะถูกกำหนดโดยส่วนระบุอุปกรณ์หรือ Identifier ที่มีลักษณะเฉพาะตัวในเครือข่าย ทุก ๆ โหนดบนบัสจะรับข้อความและคำสั่งการทำงานจากนั้นแต่ละโหนด จะตัดสินใจว่า จะรับข้อความนั้นไว้หรือไม่ โดยดูจากบิตระบุอุปกรณ์

RF 2.4GHz เครือข่ายไร้สาย (Wireless Local Area Network) เป็นระบบการสื่อสารข้อมูลโดยใช้การส่งคลื่นความถี่วิทยุในย่าน RF และคลื่นอินฟาเรด ช่องความถี่ไร้สายถูกกำหนดให้มี 14 ช่อง ตามมาตรฐาน IEEE 802.11g 2.4-GHz โดยย่านความถี่ที่ศึกษานั้นใช้ย่านความถี่ 2400-2483.5MHz ISM/SRD และใช้รูปแบบการการมอดูเลตเชิงเลขทางความถี่ (Frequency-Shift Keying : FSK) การมอดูเลตแบบ FSK ขนาดของคลื่นพาห์จะไม่เปลี่ยนแปลง ลักษณะของสัญญาณมอดูเลตนั้น เมื่อค่าของบิตของสัญญาณข้อมูลดิจิตอลมีค่าเป็น 1 ความถี่ของคลื่นพาห์จะสูงกว่าปกติ และเมื่อบิตมีค่าเป็น 0 ความถี่ของคลื่นพาห์ก็จะต่ำกว่าปกติ

STM32F103VCT [5] เป็นไมโครคอนโทรเลอร์แบบ 32 Bit ARM Cortex-M3 Processor, มีความเร็วในการประมวลผล 72MHz Clock / 90MIPS (1.25DMIPS/MHz) มี CAN Interface ภายในตัวและรองรับมาตรฐาน CAN 2.0B (High-speed)

RF โมดูลที่นำมาใช้นั้นใช้ chip ของ TI เบอร์ CC2500 [6] ซึ่งถูกนำมาใช้มาเป็นโมดูล [7] RF ST-TR2.4D Transceiver ซึ่งเป็นโมดูลสำเร็จรูป

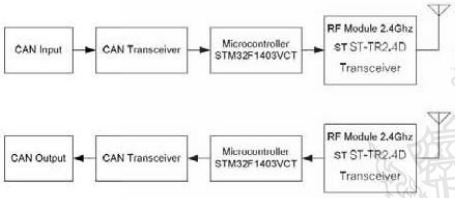
4. รายละเอียดการพัฒนา

จากข้อมูลที่ศึกษาไมโครคอนโทรเลอร์ STM32F103VCT ที่มีขนาด 32 bit นั้นมีความเร็วในการประมวลผลมากกว่าไมโครคอนโทรเลอร์ขนาด 8 bit และสามารถออกแบบโปรแกรมให้นำข้อมูลที่รับส่งเข้าถึงรีจิสเตอร์ได้โดยตรงเพื่อที่จะรับ-ส่งข้อมูลกับโมดูล RF 2.4GHz ซึ่งเป็น

© ECTI-CARD 2011, May' 5-6, Bangkok, Thailand. ISBN: 978-974-350-301-6

โมดูลสำเร็จรูปที่สามารถรับส่งข้อมูลได้ในตัว มา ออกแบบวงจรเพื่อใช้การทดสอบจำนวน 2 ชุด เพื่อที่จะนำวงจรที่ทำสำเร็จแล้วมาทำการทดสอบกับโปรแกรมที่ออกแบบไว้ โดยการทดสอบนั้น จะใช้อุปกรณ์ที่ใช้ในการทดสอบสัญญาณ CAN มาใช้ในการทดสอบ นำไมโครคอนโทรเลอร์ที่ได้ออกแบบมาทำการติดติดต่อสื่อสารกับอุปกรณ์ เพื่อนำข้อมูลที่ได้รับนั้นมาทำการเปลี่ยนแปลงรูปแบบ เพื่อที่จะส่งข้อมูลที่ได้รับมาออกทางโมดูล RF เมื่อโมดูลไมโครคอนโทรเลอร์ปลายทางได้รับข้อมูลก็จะถูกนำมาประมวลผลแล้วส่งออกมาเพื่อติดต่อกับอุปกรณ์ปลายทาง ในรูปแบบของ CAN ต่อไป

4.1 ภาพรวมของระบบ



รูปที่ 1 ภาพรวมหลักการทำงานของระบบ

จากรูปหลักการทำงานโดยรวมนั้นเมื่อนำอุปกรณ์ที่มีการใช้งานการสื่อสารในรูปแบบ CAN มาทำการเชื่อมต่อกับวงจร ข้อมูลที่ได้รับจะถูกเปลี่ยนแปลงให้อยู่ในรูปแบบที่ MCU สามารถนำมาประมวลผลได้ โดยใช้ IC CAN Transceiver หลังจากที่ไมโครคอนโทรเลอร์ ได้รับข้อมูลมาแล้วก็จะประมวลผลข้อมูลให้อยู่ในรูปแบบที่สามารถติดต่อกับโมดูล RF ได้คือรูปแบบการสื่อสารแบบ SPI เมื่อโมดูล RF ได้รับข้อมูลที่ถูกส่งมาจากไมโครคอนโทรเลอร์ โมดูล RF จะทำการแปลงข้อมูลให้อยู่ในรูปแบบของความถี่เพื่อที่จะส่งสัญญาณออกไปให้กับโมดูล RF ตัวรับเมื่อโมดูล RF ตัวรับได้รับข้อมูล ข้อมูลก็จะถูกส่งต่อไปให้กับไมโครคอนโทรเลอร์ผ่านทาง SPI ด้านตัวรับเพื่อจะประมวลผลกลับไปให้อยู่ในรูปแบบของ CAN เพื่อที่จะได้ส่งข้อมูลออกไปให้กับอุปกรณ์ปลายทางได้ ในขณะเดียวกันทางด้านรับเมื่อมีการรับข้อมูลที่จะถูกส่งกลับก็จะดำเนินการแบบเดียวกัน

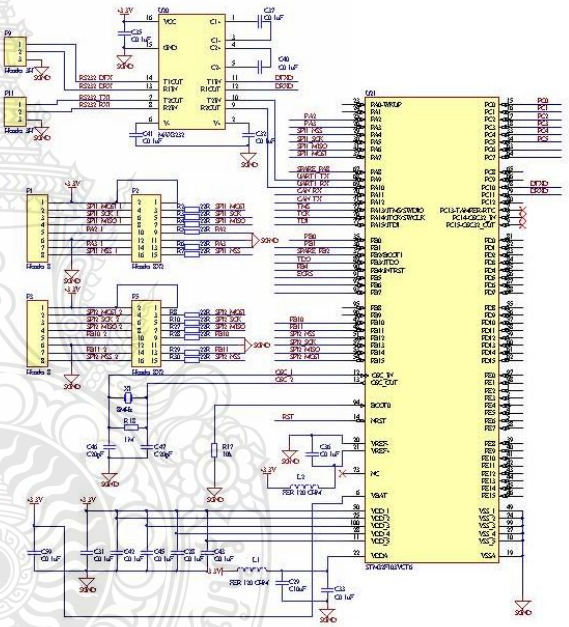
4.2 การออกแบบและพัฒนาระบบ

4.2.1 ศึกษาหลักการทำงานของรูปแบบและวิธีการสื่อสารในระบบ CAN

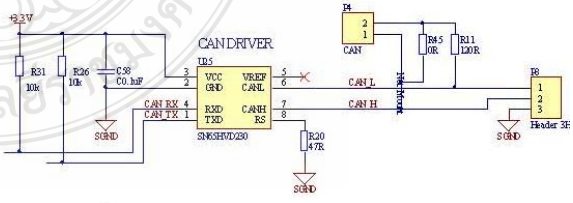
4.2.2 ค้นคว้าข้อมูลเกี่ยวกับไมโครคอนโทรลเลอร์ที่มีความสามารถในการสื่อสารในรูปแบบของ CAN โดย MCU ที่เลือกใช้คือ STM32F103VCT

4.2.3 ออกแบบวงจรโดยกำหนดการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ กับโมดูลไร้สาย ที่นำมาใช้งานตามคุณลักษณะของ Chip และโมดูลให้มีความสัมพันธ์กัน

4.2.3.1 ออกแบบวงจรของไมโครคอนโทรลเลอร์ และส่วนติดต่อกับ CAN communication



รูปที่ 2 วงจรไมโครคอรโทรเลอร์ STM32F103VCT



รูปที่ 3 วงจร CAN Driver SN65HVD230

© ECTI-CARD 2011, May' 5-6, Bangkok, Thailand. ISBN: 978-974-350-301-6

4.2.4 ประกอบวงจรโดยการแยกทดสอบทีละส่วนของวงจร

4.2.5 ออกแบบโปรแกรมเพื่อทดสอบการทำงานของวงจรเบื้องต้น

4.2.6 ออกแบบโปรแกรมเพื่อทดสอบการทำงานโดยรวมของระบบและการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับโมดูลRF

4.2.7 ออกแบบโปรแกรมเพื่อทำการทดสอบการเชื่อมต่อระหว่างตัวรับและตัวส่ง

4.2.8 ออกแบบโปรแกรมเพื่อควบคุมชุดข้อมูลในการรับส่งระหว่างวงจรรวม (Board)

4.2.9 ทดสอบการรับส่งข้อมูลระหว่างกัน รวมทั้งทดสอบโดยการเพิ่มความเร็วในการรับส่งเพื่อให้ได้ตามรูปแบบที่กำหนดไว้

4.2.10 ตรวจสอบข้อมูลในการรับส่งว่ามีความถูกต้องของข้อมูลและเวลาที่ใช้แตกต่างจากเดิมเท่าใด

4.3 ข้อจำกัดของระบบ

4.3.1 ไม่สามารถรับส่งข้อมูลในรูปแบบของ CAN ที่ baud rate ตั้งแต่ 500K -1M ได้อันเนื่องมาจากคุณสมบัติของโมดูล RF ข้อจำกัดของขนาดการรับ-ส่งข้อมูลที่ไม่เกิน 500K

4.3.2 ระยะทางและพื้นที่ในการรับส่งข้อมูลจำเป็นต้องอยู่ในพื้นที่จำกัด คือ อยู่ในพื้นที่โล่ง ไม่มีสิ่งกีดขวาง ระยะทางไม่ไกลมากเกิน จึงจะทำให้การรับส่งมีความสมบูรณ์

4.3.3 ในขณะที่มีการรับส่งอาจเกิดสูญเสียข้อมูลอันเนื่องมาจากสัญญาณรบกวนจากภายนอกได้

5. การทดสอบการใช้งาน

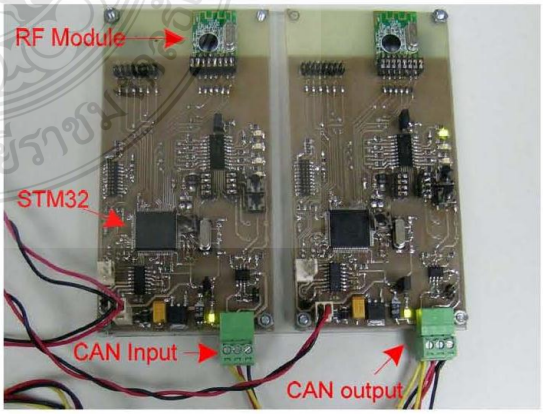
ในงานวิจัยนี้จะดำเนินการทดสอบระบบการทำงานของการรับส่งข้อมูลในรูปแบบ CAN โดยการนำอุปกรณ์ที่มีความสามารถในการสร้างสัญญาณและรับข้อมูลในรูปแบบ CAN โดยส่งข้อมูลเข้าทาง input ของ CAN driver บนบอร์ด Master โดยกำหนดขนาดของข้อมูลที่ใช้ในการทดสอบ เริ่มต้นตั้งแต่ค่าน้อยที่สุดที่ใช้งานจริงของรูปแบบการสื่อสารข้อมูลในรูปแบบ CAN เท่ากับ 5 kbps จนถึง 500 kbps และทำการรับข้อมูลที่บอร์ด Slave นำข้อมูลมาเปรียบเทียบกันว่ามีความถูกต้องของข้อมูลเมื่อเทียบกับข้อมูลทางด้าน Input หรือไม่ โดยกำหนดระยะทางในการสื่อสารเมื่อมีการรับส่งข้อมูลสูงสุดของการสื่อสาร โดยการเพิ่มระยะทางให้ห่างจากบอร์ด Master ออกมาเป็นลำดับขั้น ของระยะทาง เพื่อทดสอบการทำงานของวงจร ว่ามีความสามารถในการรับ-ส่งข้อมูลที่สมบูรณ์ได้ไกลสุดเท่าใดโดยที่ข้อมูลไม่มีความผิดพลาดหรือมีความผิดพลาดของข้อมูลน้อยที่สุด

5.1 สภาพแวดล้อมในการทดสอบ

อุปกรณ์ที่นำมาใช้ในการทดสอบคืออุปกรณ์ในการสร้างและรับสัญญาณหรือข้อมูลในรูปแบบ CAN (PCAN USB), โปรแกรม PCAN-View, บอร์ดวงจร MCU STM32F103VCT ที่ออกแบบ ต่อร่วมกับ โมดูล RF 2.4GHz (ST-TR2.4D) จำนวน 2 บอร์ด ซึ่งเป็นโมดูลRF นั้น เป็นโมดูลสำเร็จรูปที่มีกำลังส่ง 2.4 GHz และมีเสาอากาศอยู่บนโมดูล โดยกำหนดให้บอร์ดแรกเป็น Master ส่วนบอร์ดที่ 2 เป็น Slave ทดสอบรับส่งภายในพื้นที่โล่งไม่มีสิ่งกีดขวาง

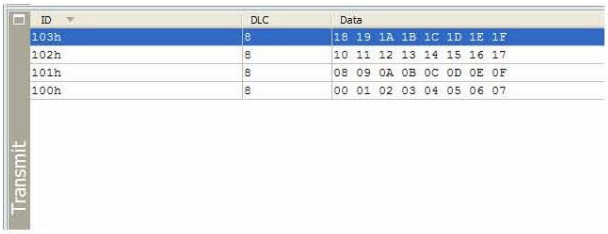
5.2 ผลการทดสอบและการวิจารณ์ผล

การทดสอบเบื้องต้นนั้น ทำการทดสอบโดยการนำสัญญาณของ CAN ป้อนให้ Input ของวงจรโดยใช้โปรแกรมและอุปกรณ์กำเนิดสัญญาณ CAN เข้าที่บอร์ด Master จากนั้นทำการรับข้อมูลที่ถูกส่งออกมาทางบอร์ด Slave เข้าอุปกรณ์ทดสอบสัญญาณ CAN



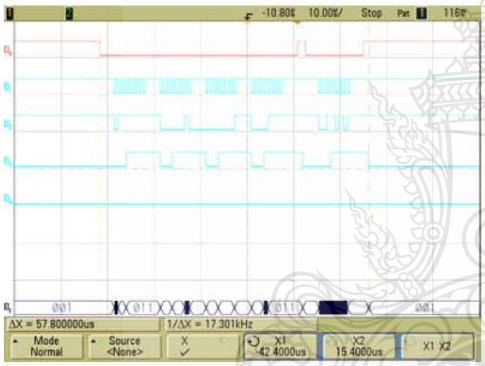
รูปที่ 4 การต่อและทดสอบการทำงานเบื้องต้น

© ECTI-CARD 2011, May' 5-6, Bangkok, Thailand. ISBN: 978-974-350-301-6



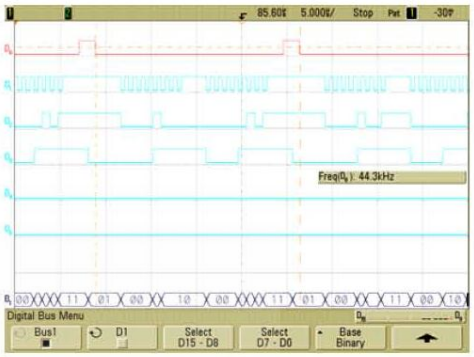
รูปที่ 5 ทดสอบการส่งข้อมูล CAN ให้กับไมโครคอนโทรเลอร์

วัดสัญญาณระหว่างการเชื่อมต่อของไมโครคอนโทรเลอร์ กับโมดูล RF โดยจับสัญญาณ ของ SPI BUS ระหว่างไมโครคอนโทรเลอร์กับโมดูล RF เพื่อทดสอบว่ามีการรับ-ส่งข้อมูลระหว่างกันในขณะส่งข้อมูลที่บอร์ด Master หรือไม่



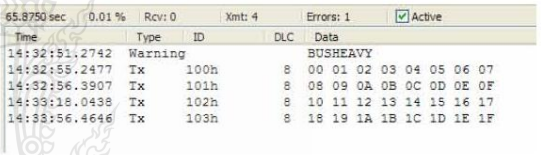
รูปที่ 6 วัดสัญญาณระหว่าง STM32 กับโมดูล RF ST-TR2.4D (ส่งข้อมูล)

รับข้อมูลกโดยใช้โมดูลตัวรับทางบอร์ด Slave โดยทำการวัดสัญญาณระหว่างการเชื่อมต่อของไมโครคอนโทรเลอร์ กับโมดูล RF โดยวัดสัญญาณ ของ SPI BUS ทางด้านรับว่ามีการวนรับข้อมูลที่ถูกส่งมาแล้วหรือไม่ เมื่อมีการส่งข้อมูลออกมาเพื่อที่จะนำข้อมูลที่ถูกส่งเข้ามาที่ไมโครคอนโทรเลอร์ทำการประมวลผลและตรวจสอบข้อมูลแล้วส่งต่อข้อมูลให้กับ CAN Driver ในการติดต่อกับอุปกรณ์ในการรับและทดสอบสัญญาณ CAN



รูปที่ 7 วัดสัญญาณระหว่าง STM32 กับโมดูล RF ST-TR2.4D (รับข้อมูล)

ทำการวัดข้อมูลที่ได้รับมาโดยอุปกรณ์กำเนิดสัญญาณ CAN เพื่อเทียบ Data ระหว่างต้นทางและปลายทางว่ามีข้อมูลที่ถูกต้องตรงกัน เพื่อพิสูจน์ได้ว่าเป็นข้อมูลที่ได้รับมาจาก โมดูล RF มีความถูกต้อง



รูปที่ 8 ผลการทดสอบการรับข้อมูล CAN จาก ไมโครคอรโทรเลอร์

เพิ่มระยะห่างระหว่างบอร์ดให้มากขึ้น โดยทดสอบความถูกต้องของข้อมูลว่ายังมีความถูกต้องหรือไม่และเพื่อทดสอบว่าระยะทางที่ดีที่สุดในการรับส่งข้อมูลมีระยะทางเท่าไหรในการใช้งานจริง

6. บทสรุป

จากการทดสอบพบว่า เมื่อมีการเพิ่มความเร็วในการรับ-ส่ง มากขึ้นก็จะเกิดปัญหาในเรื่องของความถูกต้องของข้อมูลที่ได้รับ เหตุผลอันเนื่องมาจากขีดจำกัดของการรับ-ส่งข้อมูล ของโมดูล RF ที่สามารถ รับ-ส่ง ได้ไม่เกิน 500kbps.เมื่อมีขนาดข้อมูลมากทำให้ความถูกต้องของข้อมูล ลดลง และเวลาที่ใช้ในการรับส่งจนครบก็เพิ่มขึ้นอีกด้วย จากการวนรอบรับข้อมูลเพื่อตรวจสอบความถูกต้อง รวมถึงระยะทางที่ทดสอบพบว่า เมื่อมีขนาดข้อมูลมากขึ้น ระทางที่ทดสอบถ้ามากกว่า 3 เมตร จะมีความผิดพลาดของข้อมูลสูงแต่ถ้าลดขนาดของ

© ECTI-CARD 2011, May' 5-6, Bangkok, Thailand. ISBN: 978-974-350-301-6

ข้อมูลลงก็จะได้ระยะทางมากขึ้นจากเดิม อันเนื่องมาจากเวลาที่สูญเสียไปในการวิเคราะห์ข้อมูลที่ถูกต้องในการใช้งานจริง

6.1 แนวทางการพัฒนาต่อ

สามารถนำมาพัฒนาเพิ่มเติมโดยการออกแบบโปรแกรมให้มีความรวดเร็วให้การวิเคราะห์ข้อมูลเพิ่มขึ้นรวมทั้งเพิ่มความเร็วในการรับ-ส่งให้มากขึ้นเพราะว่ารูปแบบการสื่อสารในรูปแบบ CAN นั้นสามารถรับ-ส่งได้ถึง 1Mbps หรือเพิ่มความสามารถของการรับส่งที่มากกว่า 2 ชุดขึ้นไปโดยที่ทุกตัวสามารถทำงานได้พร้อมกัน

7. เอกสารอ้างอิง

[1] Ai Chunli, Zhang Fengdeng,Liu Rongpeng "Research on Wireless Backup for CAN in Process Control System" College of Optical & Electronic Information Engineering , IEEE

[2] Harish Chincholi "Wireless Controller Area Network Based Cross Channel Data Link" Electronics & Communication Department, 978-1-4244-4740-4/09/$25.00 ©2009 IEEE

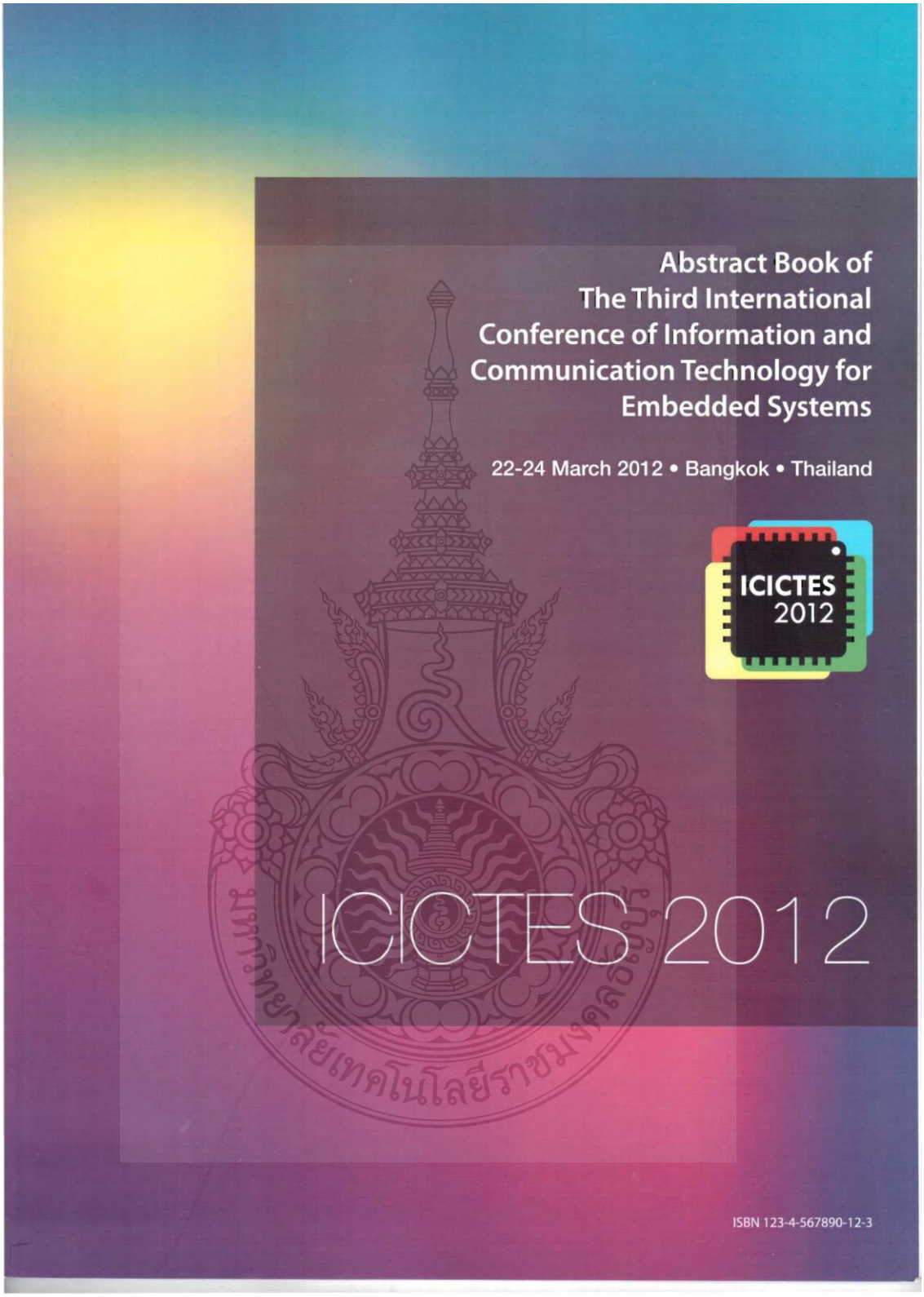
[3] CANRF™ UHF Wireless CAN module , datasheet : http://www.canrf.com

[4] ปฏิบัติการไมโครคอนโทรลเลอร์ ARM Cortex – M3 กับ STM32 : นคร ภักดีชาติ, ชัยวัฒน์ ลิ้มพรจิตรวิไล

[5] Microcontroller STM32F103VCT , ARM Cortex™-M3 datasheet. http:// www.st.com

[6] Chipcon Corporation. CC2500 Transceiver datasheet. http://www.chipcon.com.

[7] FSK 2.4GHz FSK/MSK/ASK/OOK TRANSCEIVER MODULE , datasheets http://www.thaieasyelec.net/archives/Manual/ST-TR2.4D.pdf


Abstract Book of
The Third International
Conference of Information and
Communication Technology for
Embedded Systems
22-24 March 2012 • Bangkok • Thailand
ICICTES
2012
ICICTES 2012
ISBN 123-4-567890-12-3

Organizing Committee

Honorary Chairs:

- Nobuo Fujii, Tokyo Institute of Technology
- Somkit Lertpaithoon, Thammasat University
- Vudtechai Kapilakanchana, Kasetsart University
- Pansak Siriruchatapong, NECTEC

General Chairs (Conference Chairs):

- Chongrak ulolprasert, SIIT, Thammasat University
- Asanee Kawtrakul, NECTEC
- Thanya Kiatiwat, Kasetsart University
- Akinori Nishihara, Tokyo Institute of Technology
- Somnuk Sirisoonthorn, NSTDA
- Takashi Kitahara, Tokyo Institute of Technology

Technical Program Chairs:

- Nobuhiko Sugino, Tokyo Institute of Technology
- Kanokvate Tunpimolrut, NECTEC
- Pisut Raphisak, Kasetsart University
- Thanaruk Theeramunkong, SIIT, Thammasat University

Finance Chairs:

- Gun Srijuntongsiri, SIIT, Thammasat University
- Itthisek Nilkhamhang, SIIT, Thammasat University
- Suneat Pranonsatit, Kasetsart University
- Teera Phatrapornnant, NECTEC

International Advisory:

- Hiroaki Kunieda, Tokyo Institute of Technology
- Holger Voos, University of Luxembourg
- Kin-Man Lam, The Hong Kong Polytechnic University

Local Arrangement Chairs:

- Chalie Charoenlarpnopparut, SIIT, Thammaset University
- Boontawee Suntisrivaraporn, SIIT, Thammaset University
- Bunyarit Uyyanonvara, SIIT , Thammasat University
- Cholwich Nattee, SIIT, Thammasat University
- Komwut Wipusitwarakun, SIIT, Thammasat University
- Nirattaya Khamsemanan, SIIT, Thammasat University
- Prapun Suksompong, SIIT, Thammasat University
- Srijidtra Mahapakulchai, Kasetsart University
- Surapa Thiemjarus, SIIT, Thammasat University
- Waree Kongprawechnon, SIIT, Thammasat University

Special Session Chair:

- Tsuyoshi Isshiki, Tokyo Institute of Technology
- Stanislav S. Makhanov, SIIT, Thammasat University
- Steven Gordon, SIIT, Thammasat University
- Datchakorn Tancharoen, Panyapiwat Inst of Management
- Ruengsak Kawtummachai, Panyapiwat Inst of Management

Publicity Chairs:

- Banlue Srisuchinwong, SIIT, Thammasat University
- Chalie Charoenlarpnopparut, SIIT, Thammasat University
- Ekawit Nantajeewarawat, SIIT, Thammasat University
- Teerasit Kasetkasem, Kasetsart University
- Somrote Komolavanij, Panyapiwat Inst of Management

Publication Chairs:

- Prapun Suksompong, SIIT, Thammasat University
- Somsak Kittipiyakul, SIIT, Thammasat University
- Toshiaki Kondo, SIIT, Thammasat University
- Siriroj Sirisukpresert, Kasetsart University
- Denchai Worasawate, Kasetsart University

Web Master:

- Cholwich Nattee, SIIT, Thammasat University

Proceeding Editors:

- Lerit Nuksawn, SIIT, Thammasat University
- Netchanok Tanyawiwat, SIIT, Thammasat University
- Pakpoom Patompak, SIIT, Thammasat University
- Prarinya Siritanawan, SIIT, Thammasat University
- Thanachai Poombansao, SIIT, Thammasat University
- Thanakorn Bamrungkitjaroen, SIIT, Thammasat University
- Theerasak Sangyam, SIIT, Thammasat University

Supporter:

- National Electronics and Computer Technology Center
- ECTI Association
- IEEE Thailand Section

Technical Committee Members

- Akinori Nishihara
- Banlue Srisuchinwong
- Boontawee Suntisrivaraporn
- Bunyarit Uyyanonvara
- Chalie Charoenlarpnopparut
- Cholwich Nattee
- Choochart Haruechaiyasak
- Chugiat Garagate
- Datchakorn Tancharoen
- Denchai Worasawate
- Dongju LI
- Dusit Thanapatay
- Ekachai Phaisangittisagul
- Ekawit Nantajeewarawat
- Gun Srijuntongsiri
- Hiroaki Kunieda
- Itthisek Nilkhamhang
- Jongkol Ngamwiwit
- Kanokvate Tungpimolrut
- Komwut Wipusitwarakun
- Miti Ruchanurucks
- Nobuhiko Sugino
- Pakinee Aimmanee
- Peerayot Sanposh
- Phakphoom Boonyanant
- Pisal Yenradee
- Pisut Raphisak
- Poonlap Lamsrichan
- Prapun Suksompong
- Rachaporn Keinprasit

- Ruengsak Kawtummachai
- Siriroj Sirisukprasert
- Somrote Komolavanij
- Somsak Kittipiyakul
- Somying Thainimit
- Srijidtra Charoenlarpnopparut
- Stanislav Makhanov
- Steven Gordon
- Surapa Thiemjarus
- Taworn Benjanarasuth
- Teera Phatrapornnant
- Teerasit Kasetkasem
- Thanaruk Theeramunkong
- Thepchai Supnithi
- Toshiaki Kondo
- Tsuyoshi Isshiki
- Usana Tuntoolavest
- Vutipong Areekul
- Wachira Chongburee
- Waree Kongprawechnon
- Wirat Chinnan
- Wiroonsak Santipach

Organizers

- Sirindhorn International Institute of Technology, Thammasat University
- Department of Electrical Engineering, Kasetsart University
- Tokyo Institute of Technology

Session 1B:

Application of Pipelining Technique in Concatenated Tomlinson Harashima Precoder for Downlink Multiuser MIMO Systems 15

Thanakorn Bamrungkitjaroen, Prapun Suksompong, Chalie Charoenlarpnopparut, Kamol Kaemarungsi, and Kazuhiko Fukawa

Performance of Uplink Multi-User MIMO WiFi with Capacity Maximizing User Selection and Uniformly Distributed User Locations 16

Theerasak Sangyam, Chalie Charoenlarpnopparut., Prapun Suksompong. , Kamol Kaemarungsi, and Kazuhiko Fukawa

Design and Implement a Multi-node Wireless CAN Communication 17

Wuttichai Buanark, and Jakkree Srinonchat

Session 2A:

Automatic Screening of Angle-Closure Glaucoma Using Peripheral Anterior Chamber Depth Estimation in Slit-Lamp Images 19

K. Rakjaeng, W Kongprawechnon, T Kondo, K. Tungpimolrut, N. Sugino, and A. Manassakorn

A Visual Tracking Method using the Hamming Distance 20

Prarinya Siritanawan, Toshiaki Kondo, Kanokvate Tungpimolrut, and Itsuo Kumazawa

Optic Cup and Disc Segmentation for Automatic Glaucoma Detection 21

C. Burana-Anusorn, W Kongprawechnon, T Kondo, S. Sintuwong and K. Tungpimolrut

Comparison of Position and Trajectory Tracking Controller for Underwater Robotics Vehicles 22

Pakpoom Patompak, Itthisek Nilkhamhang, Kanokvate Tungpimolrut, and Hiroaki Kunieda

A Study of Fingerprint Image Enhancement by Example-Based Super-Resolution 23

Supawan Kumpituck, Dongju Li, Tsuyoshi Isshiki, and Hiroaki Kunieda

Scalable Template Matching for Eye Tracking on Mobile Phone 24

Verachad Wongsawangtham, Somying Thainimit, Sanparith Marukatat , and Xiaolin Zhang

Session 2B:

Performance Analysis of MPEG-4 Medical Image Transmission Systems over Nakagami-m Block-fading Channels 26

Waraporn Tumchart, and Srijidtra Mahapakulchai

Positioning Accuracy Impact of Calibration Point Reduction in Location Fingerprinting based on Wi-Fi Indoor Positioning Systems 27

Chavalit Koweerawong, Kamol Kaemarungsi, and Komwut Wipusitwarakun

Ultrasound RF Signals Data Compression Before Beamforming 28

Teerapat Juthakanjana, Udomchai Techavipoo, Poonlap Lamsrichan, and Tsuyoshi Isshiki

On the Effect of Subcarrier Initial Location in SC-FDMA with Localized FDMA Mapping 29

Nut Jira-aroon, Nuttanont Promdontree, and Prapun Suksompong

Design and Implement a Multi-node Wireless CAN Communication

Wuttichai Buanark
Rajamangala University of Technology Thanyaburi
39 Muh1, Rangsit-Nakhonnayok Rd. Klong Hok,
Thanyaburi Pathum Thani Thailand.
E-Mail: wuttichai@incotec.co.th

Jakkree Srinonchat
Rajamangala University of Technology Thanyaburi
39 Muh1, Rangsit-Nakhonnayok Rd. Klong Hok,
Thanyaburi Pathum Thani Thailand.
E-Mail: Jakkree.s@en.rmutt.ac.th

***Abstract*—CAN (Control Area Network) communication is usually a BUS communication, but its problems when the system is operated the condition of independent circulation environment. This article design a wireless CAN communications multi-node on a stable speed to solve above problem. The ARM Cortex™-M3 Nr. STM32F103RET6 is used to be a CAN device with 1Mbps transmission rate. The NRF24L01 wireless module which is operated on 2.4GHz is used as a communication receiver – transmitter. The results show that this system can be completely process the data at 1Mbps within the 5 meters of module distance. Therefore each module can be repeater to cover the long distance controlling. However, when the distance between two modules is increased more than 5 meters, it requires time to be check the transmission data. This may cause the time delay process. This design can be apply to the industrial such as controlling or automation system.**

Keywords*—*Controller Area Network, ARM Cortex™-M3, wireless module, 2.4GHz, 1Mbps

I. Introduction

The Control Area Network or CAN [1] communication is a type of BUS communication and is mostly used for the control of industrial machinery, automobiles and embedded systems. It was initially developed for the use in motor vehicles by Robert Bosch GmbH in year 1985. The CAN communication is including three cables which are the CAN High, CAN Low and CAN Ground that are connected together at many different points on a single line. The highest transmission rate is at 1Mbps which has a BUS beneath 40 Meter which can be used to create a connection of more than 1 kilometer, but the transmission rate will decrease. Problems might occur in the CAN design when the device requires an independent circulation specially on wireless modules. When a wireless module is added, an additional program has to be designed for the additional device at the transmitter and receiver, or a new design is required for the whole system in order to support the data transmission. At some cases, the appliance of wireless modules will not enable the connection with the microcontroller using a wide distance cables. In [2-4] it is shown that wireless modules can be developed for the CAN communication system, but it can only be applied for the transmitting and receiving. It cannot be used as a replacement for a 1Mbps communication.

This research designs the wireless CAN communication of 1Mbps based on 32bit microcontroller Nr. STM32F103RET. This controller is an ARM Cortex™-M3 64pin type. The NRF24L01wireless module is applied for SPI communication system at 2.4GHz. Two testing sets are used for the transmission, at which the first set is stable and the second operates on an independent circulation. For the transmission tests, the programs PCAN View and PCAN USB [5] will function as a transmitter of CAN messages to the transmitting module at a data transmission frequency of 1Mbps. When the microcontroller has received and processed the data, it will send the data to the wireless module to the receiving module. The received data will be tested and processed. If the data is accurate and complete, a summation of the received values of each bit has to be done by the value 0x01h with the all received CAN data and send back to the transmitting module in order to send the data to the PCAN-USB and the PCAN-View program. The value of the received data will be different than the value of the transmitted data. This will be done in order to gain the accuracy of the data and will prove that the module is able to transmit the data.

II. Theory

A. Controller Area Network

The information regarding the architecture of the BUS device CAN [6] in this research is according to ISO 11898, which concerns the transmission of data between the networks system. It is consistent with the OSI (Open System Interconnection) in BUS CAN. Messages can be sent from each node which each node is able to receive data in the CAN. An address has been determined for the transmitting and receiving nodes. The details of the message will be determined by the identifier which has a unique function within the network. Every node on the BUS will receive messages and commands and will decide if they have received the message by monitoring the bit defining device.

B. Microcontroller ARM Cortex-M3 Processor

System-on-chip solutions based on ARM embedded processors address many different market segments including enterprise applications, automotive systems, home networking and wireless technologies. The ARM Cortex™ family [7] of processors provides a standard architecture to address the broad performance spectrum required by these diverse technologies. The ARM Cortex family includes processors based on the three distinct profiles of the ARMv7 architecture; the A profile for sophisticated, high-end applications and

complex operating systems; the R profile for real-time systems; and the M profile for cost-sensitive and microcontroller applications. The Cortex-M3 processor is the first ARM processor based on the ARMv7-M architecture and has been specifically designed to achieve high system performance in power and cost-sensitive embedded applications, such as microcontrollers, automotive body systems, industrial control systems and wireless networking. It is significantly simplifying programmability to make the ARM architecture an option for even the simplest applications.

C. Radio Frequency ISM Band

The 2.4 GHz ISM band [8] allows for primary and secondary uses. Secondary uses are unlicensed, ISM band are radio bands (portions of the radio spectrum) reserved internationally for the use of radio frequency (RF) energy for industrial, scientific and medical purposes other than communications. Examples of applications in these bands include radio-frequency process heating, microwave ovens, and medical diathermy machines. The powerful emissions of these devices can create electromagnetic interference and disrupt radio communication using the same frequency, so these devices were limited to certain bands of frequencies. In general, communications equipment operating in these bands must tolerate any interference generated by ISM equipment, and users have no regulatory protection from ISM device operation.

III. Hardware and Software design

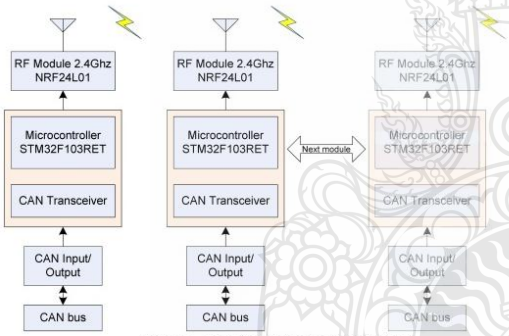


Figure 1. Display of System interaction

Figure 1 shows the process of the system. In the receiving the CAN signal, the received data will be adjusted by IC CAN Transceiver in order to send them to the microcontroller. When the processing is done, the system will send data to the RF module by using the SPI. The RF modules will convert the data in order to transmit as a signal to the RF receiver. When the RF module has received the data, the module will send the data to the microcontroller by SPI to process the data. The CAN data will then be sent to the destination device. At the same time, when the receiver receives the data, the same process will be applied to return a feedback by dividing into two parts, the soft- and hardware.

A. Hardware design

1) CAN Interface

The CAN interface circuit design show in Fig 2. The major is the CAN Transceiver which is IC CAN Transceiver SN65HVD230 [9] from TI, It is IC operating on a voltage range of 3.3V, which is connected to the CAN TX and the CAN RX port of the microcontroller. It operates on the same voltage range.

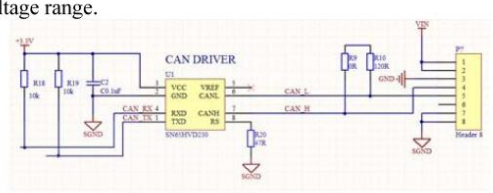


Figure 2. CAN Transceiver Circuit

2) Microcontroller

STM32F103RET6 [10] is a microcontroller of ST microelectronic type ARM Cortex-M3 Processor of 32 bits, which has the processing speed of 72MHz Clock/90MIPS (1.25DMIPS/MHz). It includes a flash memory of 512 kilobyte that requires 2.0-3.6V. It has a small size tank with 64 connection legs, which is a microcontroller with a CAN module interface and supports the standard CAN 2.0B (High-speed). It has input and output ports, which can be applied for several purposes. It includes the SPI communication, which can further be used with external wireless modules as well.

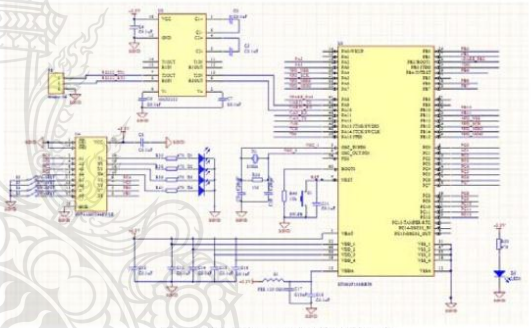


Figure 3. Microcontroller Circuit

The microcontroller circuit is displayed in Fig 3. The STM32F103RET microcontroller is used as a main processor. It is used for receiving-transmitting CAN data by the CAN TX and the CAN RX port, to process the CAN data into RF message, and to connect with RF module by SPI communication. It also has other features for the display of the results, system testing by LED and a switch with a RS232 connector used for the connection to the computer to display the transmission test results.

3) *Interface Module NRF24L01*

The RF module, which is used as a ready-made module NRF24L01 [11] uses a chip of Nordic Semiconductor operates on the frequency between 2.4 - 2.4835GHz .It is used for the

modulation of the GFSK signal that is able to transmit data at up to 2Mbps . It is used for the data communication with the microcontroller type SPI.

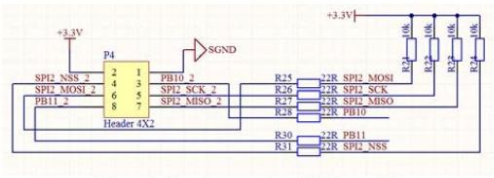


Figure 4. Circuit connection of RF type SPI

The RF 2.4GHz module transceiver is a NRF24L01 wireless module. It is connected to the microcontroller that communicates by using the SPI as show in Fig 4. The receive data has been processed by this microcontroller to the CAN receiver module.

B. Software Design

The design of the software is divided into two different sections that include the receiving and the transmitting section as shown in Fig 5.

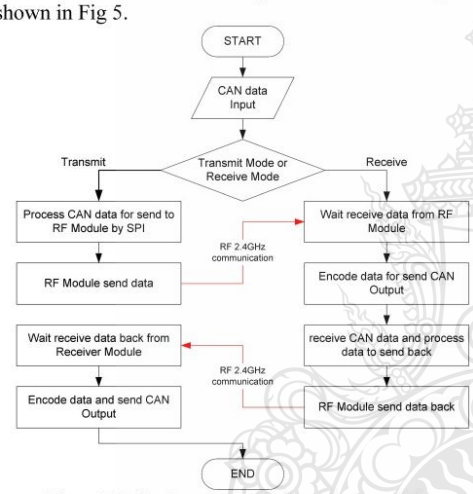


Figure 5. Design Layout of the Software for test solution

When a module receives a CAN message and is determined to function as a transmitter, the message frame will be divided into parts. The microcontroller will separate the data to store it. The data set includes data bits, which are used to verify the accuracy of the data to check the errors. When the data has been verified, the data will be sent to the RF module which functions to transmit the data and receives. The data will be feedback after the transmission has been completed. The receiver will receive the data. The RF module will receive and send it to the microcontroller to processes. At this point, the receiver becomes a CAN to transmit the data to the destination device and to receive CAN data feedback to confirm a complete transmission.

IV. System Testing

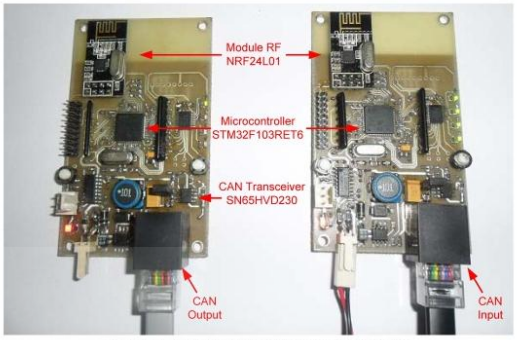


Figure 6. Receivers and Transmitter Module

Fig 6 shows the CAN design circuit. These modules are tested in two sets, receiver and transmitter part. The system is tested in closed environment that does not have any signal interference. The PCAN-View program and PCAN-USB are used in the testing of the receiver and transmitter process.

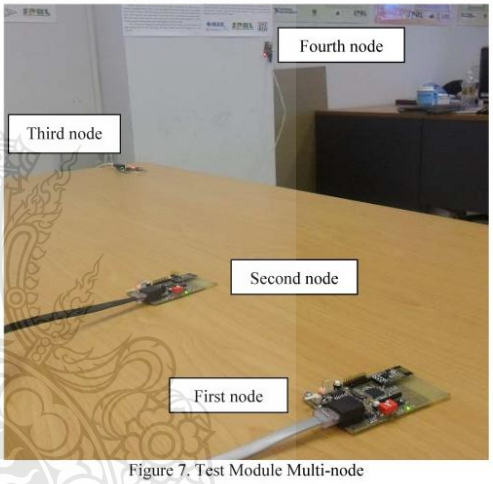


Figure 7. Test Module Multi-node

Fig 7 shows the testing of four modules. The modules receive CAN data from first module and send data to each module. The CAN data is send back to PCAN-USB. The PCAN-View program is used for testing CAN communication.

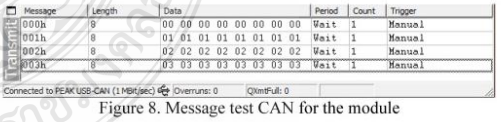


Figure 8. Message test CAN for the module

When a CAN message is sent to the microcontroller (Fig. 8), the message will be processed and send it to the RF

module. The LED is used to show the status of receiver or transmitter the data.

The data, which has been received by the module, is processed by the microcontroller to become a CAN message. The value of each bit is added together by 0x01h and is returned to the transmitting module. When the microcontroller receives the data from the receiver, the data is examined and processed to become a CAN message in order to send the message to the PCAN-USB. The data of CAN message received is difference from transmitted message as shown in Fig.9.

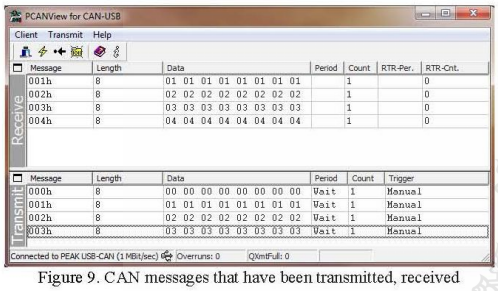


Figure 9. CAN messages that have been transmitted, received

Distance/meter	Send-Receive	Average of errors	Reconnection Modules
1-3	Yes	0%	Not
4-5	Yes	10%	Not
6-7	Yes/not stable	50%	Not
>8	Not stable	>80%	Yes

Table 1. test results the distance of modules in range

Table 1. shows the test results comparing to the distance of modules. In the range of 1-3 meters, it can completely communicate without re-sending message. In the range of 4-5 meters, the modules can communicate but it is not stable. This is because the message data is not true and requests re-sending of a new message from master module. In the range of 6-7 meters, the module is not stable if it is obstacled or the position is out of antenna range. If the range more than 8 meters, the modules can not communicate to other modules and it always requests RF-Module address. The module is not operating if RF communication is not stable because each module needs to check RF-Module address and standby for communication the other modules in range.

V. Test result conclusion

The results show that the wireless CAN data communication at a range of 1Mbps can work completely in the 5 meters distance of each module. It requires also no obstacles present. However, if the distance between the four modules has been increased, the accuracy of the data decreases as the signal starts to fade over and also the time range increases as well. But the module is still able to operate at 1Mbps. This tests results can be used to develop extend power of antenna for working long distance. Also it can be used to develop hardware and software which have memory for recording address of many module and design protocol for manage module in network group.

References

[1] CAN Specification, Version 2.0, Bosch GmbH, September 1991

[2] Ai Chunli, Zhang Fengdeng,Liu Rongpeng "Research on Wireless Backup for CAN in Process Control System", RFID Eurasia, 2007 1st Annual , pp. 1 - 6, 5-6 Sept. 2007.

[3] Harish Chincholi "Wireless Controller Area Network Based Cross Channel Data Link" Application of Information and Communication Technologies, 2009. AICT 2009, pp. 1-5, 14-16 Oct. 2009.

[4] CANRF™ UHF Wireless CAN module, datasheet : http://www.canrf.com

[5] PCAN-View for CAN USB Monitor and PCAN-USB is CAN interface for the USB port : http://www.peak-system.com

[6] International Standard Organizaion (ISO). "Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication", ISO 11898, 1993.

[7] Shyam Sadasivan" An Introduction to the ARM Cortex-M3 Processor'' WHITE PAPER ARM. October 2006.

[8] ISM Band, Wikipedia, the free encyclopedia.

[9] SN65HVD230 3.3V CAN transceiver Datasheet, http://www.ti.com/product/sn65hvd230

[10] Microcontroller STM32F103RET6, ARM Cortex™-M3 datasheet. http:// www.st.com

[11] NRF24L01 Single Chip 2.4GHz Transceiver http://www.nordicsemi.com

ประวัติผู้เขียน

ชื่อ - นามสกุล	นายวุฒิชัย บัวนาค
วัน เดือน ปีเกิด	1 มิถุนายน 2526
ที่อยู่	125 หมู่ 1 ต.ดอนเจดีย์ อ.ดอนเจดีย์ จ.สุพรรณบุรี
การศึกษา	สำเร็จการศึกษาระดับวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี ปี พ.ศ. 2549

ประสบการณ์การทำงาน

พ.ศ. 2549 – ปัจจุบัน	Hardware Engineer INCOTEC Automation (Thailand),.Ltd.