รายงานสหกิจศึกษาฉบับสมบูรณ์

การพัฒนาระบบควบคุมการทำงานของเครื่องจ่ายไฟผ่านโครงข่าย
Development of Power Supply Operation Control System Via Network

นายณัฐสิทธิ์ กิตติธนโสภณ
นายทรงกรด วารีศรลาภ

ภาควิชาอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

รายงานสหกิจศึกษาฉบับสมบูรณ์

การพัฒนาระบบควบคุมการทำงานของเครื่องจ่ายไฟผ่านโครงข่าย

Development of Power Supply Operation Control System Via Network

นายณัฐสิทธิ์ กิตติธนโสภณ

นายทรงกรด วารีศรลาภ

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ชื่อโครงงานสหกิจศึกษา การพัฒนาระบบควบคุมการทำงานของเครื่องจ่ายไฟผ่านโครงข่าย

ชื่อ-สกุล นักศึกษา นายณัฐสิทธิ์ กิตติธนโสภณ

นายทรงกรด วารีศรลาภ

คณะ วิศวกรรมศาสตร์ **ภาควิชา** อิเล็กทรอนิกส์

ชื่อ-สกุล อาจารย์นิเทศ ผศ.เกรียงไกร สุขสุด

อาจารย์ชินภัทร นันทจิวากรชัย

ชื่อ-สกุล ผู้นิเทศงาน นายภุมรินทร์ อิ่มจิตร

ชื่อสถานประกอบการ บริษัท ซิเลซติกา (ประเทศไทย) จำกัด

บทคัดย่อ

ในระบบการผลิตอุปกรณ์อิเล็กทรอนิกส์ของบริษัท ซิเลซติกา (ประเทศไทย) จำกัด จะมีขั้นตอนหนึ่งที่ต้องนำผลิตภัณฑ์มาทดสอบการทำงานให้เป็นไปตามที่ได้ออกแบบไว้ โดยจะมีทีมวิศวกรทดสอบออกแบบเครื่องทดสอบผลิตภัณฑ์ (Tester) ซึ่งหนึ่งในอุปกรณ์ของเครื่องทดสอบผลิตภัณฑ์คือเพาเวอร์ซัพพลาย ในระบบปัจจุบันการติดตั้งเพาเวอร์ซัพพลายจะต่อเข้ากับ PDU เพื่อรวมเอาต์พุตของทุกตัวเข้าด้วยกันแล้วควบคุมการทำงานผ่าน PDU ปัญหาของการติดตั้งแบบปัจจุบันคือ เพาเวอร์ซัพพลายทุกตัวต้องทำงานอยู่ตลอดเวลาแม้ว่าการทดสอบผลิตภัณฑ์จะใช้โหลดน้อยก็ตามและการที่ต้องใช้ PDU ซึ่งมีมูลค่าประมาณหนึ่งแสนบาทต่อตัวเพื่อใช้ควบคุมการเปิด/ปิดเอาต์พุตของเพาเวอร์ซัพพลายเพียงอย่างเดียวจึงไม่เหมาะสมกับต้นทุนที่ต้องเสียไปในการสร้างเครื่องทดสอบผลิตภัณฑ์แต่ละเครื่อง ทีมวิศวกรทดสอบจึงคิดที่เปลี่ยนระบบการติดตั้งแบบเดิมเป็นการใช้ไมโครคอนโทรลเลอร์ควบคุมการทำงานแทน

คำสำคัญ : โปรโตคอล PMBus, แอพพลิเคชั่น TCP/IP, เพาเวอร์ซัพพลาย, เครื่องทดสอบผลิตภัณฑ์

Co-operative Title : Development of Power Supply Operation Control System Via Network

Student Intern Name : Mr.Natthasit Kittithanasophon
Mr.Songkrod Wareesonlap

Faculty : Engineering **Department :** Eletronics

Advisor Name : Asst. Prof. Kriangkrai Sooksood
Mr.Chinnapat Nantajiwakornchai

Mentor Name : Mr.Bhumarin Imjit

Company : Celestica (Thailand) Ltd.

ABSTRACT

In the Celestica's electronic production system, there is one step where the product must be tested to work as designed. The testing team will design a Tester, one of component in the tester is the Power Supply Unit. In the current system, the power supply is connected to the Power Distribution Unit to integrate all the outputs and control it via the Power Distribution Unit. The problem with the current build is Power Distribution Unit. It is worth about one hundred thousand baht per unit but its use is mostly to control the on / off output of the power supply unit. Therefore, it is not suitable for the cost to build each Tester with the Power Distribution Unit. Test engineers are thinking of replacing Power Distribution Unit with microcontrollers and write a software to control the power supply on it.

Keywords : PMBus protocol, TCP/IP application, Power Supply Unit, Tester

กิตติกรรมประกาศ

ทางผู้เข้าร่วมโครงการสหกิจศึกษาขอขอบคุณ บริษัท ซิเลซติกา (ประเทศไทย) จำกัด ที่ให้โอกาสในการเข้าร่วมโครงการสหกิจศึกษาครั้งนี้ ขอขอบคุณพี่ๆ แผนก Test Development Engineer ซึ่งประกอบไปด้วยทีม Test Engineer, Software Engineer, Machanicial Engineer และ Electronics Engineer ทุกคน ที่คอยช่วยเหลือดูแลและให้คำแนะนำต่างๆตลอดระยะเวลา 4 เดือน ขอขอบคุณอาจารย์นิเทศ ที่คอยช่วยเหลือ ติดต่อประสานงานและให้คำปรึกษา และขอขอบคุณเพื่อนๆที่เข้าร่วมโครงการสหกิจศึกษาทุกคนที่คอยช่วยเหลือดูแลตลอดมา ส่งผลให้ผู้เข้าร่วมโครงงานสหกิจได้รับความรู้และประสบการณ์ต่างๆ

นายณัฐสิทธิ์ กิตติธนโสภณ

นายทรงกรด วารีศรลาภ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ในเครื่องทดสอบผลิตภัณฑ์ (Tester) ที่ติดตั้งอยู่ในโรงงานอุตสาหกรรมจำเป็นต้องมีเพาเวอร์ซัพพลายให้กับผลิตภัณฑ์ที่ต้องการทดสอบ โดยระบบควบคุมการจ่ายไฟที่ใช้อยู่ในปัจจุบันมีราคาค่อนข้างสูงและระบบค่อนข้างซับซ้อน จึงได้ทำการออกแบบระบบใหม่ทั้งหมดทั้งในส่วนของซอฟต์แวร์และฮาร์ดแวร์เพื่อลดต้นทุนการผลิตและลดความซับซ้อนของระบบ

1.2 วัตถุประสงค์ของการวิจัย

- ลดต้นทุนของอุปกรณ์ที่ใช้ โดยตัว Power Distribution Unit มีราคาค่อนข้างสูง
- ลดกำลังงานสูญเสียจากการเปิดเพาเวอร์ซัพพลายตลอดเวลา
- ลดการใช้พื้นที่ของการติดตั้งอุปกรณ์ในตู้ Rack
- เพิ่มเสถียรภาพของสัญญาณไฟ เนื่องจากการเปิด-ปิดจากเพาเวอร์ซัพพลายโดยตรงจะให้สัญญาณไฟที่มีเสถียรภาพมากกว่าการเปิด-ปิดจาก Power Distribution Unit

1.3 ขอบเขตของการวิจัย

ทำการออกแบบระบบการเชื่อมต่อใน 1 ชั้นวาง ออกแบบวงจรของบอร์ดอินเตอร์เฟซระหว่างเพาเวอร์ซัพพลายและคอนโทรลเลอร์ ออกแบบลายวงจรของบอร์ดอินเตอร์เฟซ เขียนโปรแกรมควบคุมคอนโทรลเลอร์เพื่อส่งสัญญาณไปควบคุมเพาเวอร์ซัพพลาย เพื่อพัฒนาระบบควบคุมเพาเวอร์ซัพพลายใหม่ให้มีคุณสมบัติทัดเทียมระบบเดิมแต่ใช้ต้นทุนที่ต่ำกว่า

1.4 วิธีการดำเนินการวิจัย

- ศึกษาการทำงานของระบบเดิม
- ศึกษาการใช้งาน คุณสมบัติ ของอุปกรณ์ที่จะนำมาออกแบบระบบใหม่
- กำหนดสัญญาณที่จะนำมาใช้ในระบบและเลือกอุปกรณ์อิเล็กทรอนิกส์ที่จะนำมาใช้ในวงจร
- ออกแบบการวางระบบและออกแบบวงจรของบอร์ดอินเตอร์เฟซ
- สร้าง footprint และ 3D Model ของอุปกรณ์และออกแบบลายวงจร PCB
- ติดตั้งอุปกรณ์บน PCB และทดสอบการใช้งาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- ลดค่าใช้จ่ายของอุปกรณ์ที่ใช้ในระบบควบคุมเพาเวอร์ซัพพลายผ่านโครงข่าย
- ลดการใช้งานพื้นที่ของอุปกรณ์ในตู้แร็ค
- ลดกำลังงานสูญเสียจากการเปิดใช้งานเพาเวอร์พลายตลอดเวลา
- เพิ่มทักษะทางด้านอิเล็กทรอนิกส์ให้กับผู้จัดทำโครงงาน
- สามารถพัฒนาระบบควบคุมเพาเวอร์ซัพพลายผ่านโครงข่ายใหม่ให้มีความสามารถทัดเทียมระบบเดิมแต่ใช้ต้นทุนการผลิตที่ต่ำกว่า

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในการออกแบบและเขียนโปรแกรมที่จะใช้ในการควบคุมเพาเวอร์ซัพพลายฟ้ากระแสตรงรุ่น CP2725AC54TE จะต้องศึกษาความรู้เบื้องต้นเพื่อใช้ในการดำเนินงานและวิเคราะห์แก้ไขปัญหาของโครงงานได้อย่างมีประสิทธิภาพ โดยมีข้อมูลที่ต้องศึกษาดังนี้

1. ข้อมูลทางเทคนิคของเพาเวอร์ซัพพลายฟ้ากระแสตรง GE รุ่น CP2725AC54TE
2. บอร์ดไมโครคอนโทรลเลอร์ BeagleBone Green, ระบบปฏิบัติการ Linux และการเขียนโปรแกรมใน userspace
3. I^2C, SMBus, PMBus protocol
4. โปรโตคอล TCP/IP (TCP/IP protocol suite)

2.1 เพาเวอร์ซัพพลาย GE รุ่น CP2725AC54TE [1]

ในหัวข้อที่ 2.1 นี้จะเป็นการศึกษาสัญญาณและข้อมูลอื่นๆที่เกี่ยวข้องของเพาเวอร์ซัพพลาย CP2725AC54TE ที่จำเป็นต่อการออกแบบและเขียนโปรแกรมควบคุมการทำงาน CP2725AC54TE

เพาเวอร์ซัพพลายฟ้ากระแสตรงรุ่น CP2725AC54TE เป็นเพาเวอร์ซัพพลายฟ้ากระแสตรงแบบยึดกับแล๊ค (rack mount) ขนาด 1RU ใช้งานสำหรับจ่ายพลังงานให้กับผลิตภัณฑ์ที่จะนำมาทดสอบ (unit under test) หลังจากกระบวนการผลิต โดยเพาเวอร์ซัพพลายฟ้ากระแสตรงรุ่นนี้รองรับการทำงานสามแบบคือ RS485, analog และ I^2C ซึ่งในโครงงานนี้จะใช้โหมด I^2C ในการสื่อสารระหว่างเพาเวอร์ซัพพลายฟ้ากระแสตรงกับไมโครคอนโทรลเลอร์ BeagleBone Green ในการออกแบบและเขียนโปรแกรมสำหรับควบคุมการทำงานจำเป็นจะต้องศึกษาคุณสมบัติของสัญญาณต่างๆดังนี้

2.1.1 ความหมายและคำอธิบายของสัญญาณ

สัญญาณต่างๆของเพาเวอร์ซัพพลายฟ้ากระแสตรงรุ่น CP2725AC54TE ที่ใช้ในการอินเตอร์เฟสกับไมโครคอนโทรลเลอร์ BeagleBone แสดงดังตารางที่ 2.1 ขาคอนเน็คเตอร์สำหรับสัญญาณต่างๆแสดงในภาพที่ 2.1

ตารางที่ 2.1 แสดงคำอธิบายของสัญญาณ

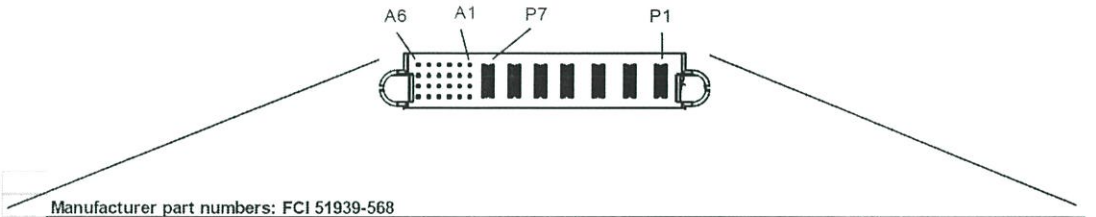
สัญญาณแจ้งเตือนของอุปกรณ์ทุกสัญญาณ (Fault, PFW, OTW, Power Capacity) เป็นแบบ open drain FETs ทุกสัญญาณต้องเชื่อมต่อกับไฟ 3.3V หรือ 5V และทุกสัญญาณมีค่าเทียบกับ LOGIC_GRD ถ้าไม่มีการระบุเพิ่มเติม

ชื่อสัญญาณ	**หน้าที่**	**ชนิด**	**คำอธิบาย**
Enable	Output Enable	อินพุท	เชื่อมต่อกับ LOGIC_GRD เพื่อจ่ายเอ้าพุทในโหมดอนาลอคหรือPMBus
PFW	Power Fail Warning	เอ้าพุท	เปลี่ยนสถานะเป็น 'LO' ก่อนที่เอ้าพุทจะลดต่ำกว่า $40V_{DC}$
Fault	Rectifier Fault	เอ้าพุท	สถานะปกติ 'HI' แล้วเปลี่ยนเป็น 'LO'
ON/OFF	ON/OFF	อินพุท	ควบคุมเอ้าพุทขณะhot-insertion และ extraction สัญญาณเทียบกับ V_{out}(-)
Protocol	Protocol select	อินพุท	เลือกโหมดการทำงาน สัญญาณมีค่าเทียบกับ V_{out}(-) - RS485 ต่อตัวต้านทาน 10kΩ - PMBus ปล่อยขานี้ว่างไว้
Margin	Margining	อินพุท	เปลี่ยนค่าเริ่มต้นของแรงดันเอ้าพุท
OTW	Over-Temperature Warning	เอ้าพุท	ปกติมีค่าเป็น 'HI' และเปลี่ยนเป็น 'LO' เมื่ออุณหภูมิมีค่าต่ำกว่า thermal shutdown 5°C
Power_Cap	Power Capacity	เอ้าพุท	'HI' ระบุว่าทำงานที่กำลังงาน 2725W 'LO' ระบุว่าทำงานที่กำลังงาน 1200W
UNIT_ADDR	Rectifier Address	อินพุท	ระดับแรงดันไฟฟ้าใช้ระบุ address ของเพาเวอร์ซัพพลายภายในแล๊คที่อยู่ชั้นเดียวกัน สัญญาณมีค่าเทียบกับ V_{out}(-)
SHELF_ADDR	Shelf Address	อินพุท	ระดับแรงดันไฟฟ้าใช้ระบุ address ของเพาเวอร์ซัพพลายภายในแล๊คที่มีอยู่หลายชั้น สัญญาณมีค่าเทียบกับ V_{out}(-)
SCL_0	I^2C Line 0	อินพุท	PMBus line 0

SDA_0	I^2C Line 0	อินพุท/เอ้าพุท	PMBus line 0
SCL_1	I^2C Line 1	อินพุท	PMBus line 1
SDA_1	I^2C Line 1	อินพุท/เอ้าพุท	PMBus line 1
ALERT#_0	SMBALERT# Line 0	เอ้าพุท	PMBus line 0 interrupt
ALERT#_1	SMBALERT# Line 1	เอ้าพุท	PMBus line 1 interrupt

Output Connector

Mating Connector: right angle PWB mate - all pins: TE 6450572-1, right angle PWB mate except pass-thru input power: TE 6450378-1



Manufacturer part numbers: FCI 51939-568

	SIGNAL						OUTPUT POWER				INPUT POWER		
	6	5	4	3	2	1	P7	P6	P5	P4	P3	P2	P1
A	SCL_0	MOD_PRES	PFW	LOGIC_GRD	RS_485+	UNIT_ADDR							
B	SCL_1	OTW	Alert#_0	Alert#_1	RS_485-	8V_INT	V_OUT (-)	V_OUT (+)	V_OUT (+)	V_OUT (-)	EARTH (GND)	LINE-2 (Neutral)	LINE-1 (HOT)
C	SDA_0	Margin	Enable	Reset	Ishare	Protocol							
D	SDA_1	Fault	5VA	Power_Cap	ON/OFF	SHELF_ADDR							

Note: Connector is viewed from the rear positioned inside the rectifier
Signal pins columns 1 and 2 are referenced to V_OUT(-)
Signal pins columns 3 through 6 are referenced to Logic GRD
Last to make-first to break shortest pin
Earth — First make-last to break longest pin implemented in the mating connector

ภาพที่ 2.1 แสดงขาคอนเน็คเตอร์ที่ด้านหลังของเพาเวอร์ซัพพลายฟ้ากระแสตรง CP2725AC54TE

ที่มา : General Electric Company CP2725AC54TE Compact Power Line High Efficiency Rectifier, September 21, 2016 : Page 19

2.1.2 การควบคุมและสถานะ

2.1.2.1 สัญญาณควบคุม

สัญญาณ Enable

ควบคุมเอ้าพุท $54V_{DC}$ ของเพาเวอร์ซัพพลาย ไม่ว่าจะใช้งานเพาเวอร์ซัพพลายในโหมดอนาล็อคหรือ PMBus สัญญาณนี้จะต้องเป็นโลจิก 'LO' เพื่อที่จะทำให้เอ้าพุทเปิดใช้งาน ถ้าสัญญาณ Enable หรือ ON/OFF เป็นโลจิก 'HI' เอ้าพุทของเพาเวอร์ซัพพลายจะถูกปิด

สัญญาณ ON/OFF

ขาของสัญญาณนี้จะมีขนาดสั้นกว่าขาอื่นเพื่อใช้สำหรับการใช้งานแบบ hot-plug สัญญาณนี้ต้องต่อกับ V_{out}(-) เพื่อที่จะทำให้เอ้าพุทเปิดใช้งานได้

สัญญาณ Margining

เอ้าพุทของเพาเวอร์ซัพพลายสามารถปรับได้ตั้งแต่ 44 -58V_{DC} โดยการควบคุมแรงกันไฟฟ้าที่ขาสัญญาณนี้ อาจใช้แรงดันไฟฟ้าจากแหล่งจ่ายไฟภายนอกหรือโดยการต่อตัวต้านทานในแบบแบ่งแรงดันภายในเพาเวอร์ซัพพลายมีตัวต้านทานต่อกับไฟ 3.3V ขนาด 10kΩ อยู่แล้ว เมื่อต่อตัวต้านทานระหว่างขา Margin และ LOGIC_GRD จะทำให้เกิดวงจรแบ่งแรงดันขึ้นที่ขานี้ สำหรับการกำหนดแรงดันไฟฟ้าเอ้าพุทด้วยโปรแกรมจะเขียนทับค่าที่ถูกกำหนดด้วยแรงดันไฟฟ้าที่ขา Margin และจะมีผลไปจนกระทั่งไมโครคอนโทรลเลอร์ภายในเพาเวอร์ซัพพลายรีสตาร์ทหรือหยุดทำงาน

สัญญาณ Protocol

ใช้สำหรับกำหนดโหมดการสื่อสารของเพาเวอร์ซัพพลายระหว่างโหมดอนาลอค, PMBus และ RS485 การตั้งค่าสำหรับโหมดอนาลอคและ PMBus ให้ปล่อยขาสัญญาณนี้ว่างไว้สำหรับในโครงงานนี้จะใช้โหมดการสื่อสารแบบอนาลอคและ PMBus

สัญญาณ Rectifier Address

การกำหนด address ให้กับเพาเวอร์ซัพพลายทำได้โดยกำหนดแรงดันไฟฟ้าที่ขาสัญญาณนี้ ภายในเพาเวอร์ซัพพลายจะมีตัวต้านทาน 10kΩ ต่ออยู่กับไฟ 3.3V อยู่ เมื่อต่อตัวต้านทานตามตารางที่ 2.2 จะทำให้ได้ address ต่างกัน 4 ค่า

ตารางที่ 2.2 แสดงค่าตัวต้านทานที่ใช้กำหนด address ของเพาเวอร์ซัพพลาย

เพาเวอร์ซัพพลาย	ค่าตัวต้านทาน (Ω)	แรงดันไฟฟ้า(V)	I^2C address	
			A1	A0
1	30k	2.477	0	0
2	14k	1.925	0	1
3	6k	1.243	1	0
4	2.5k	0.654	1	1

สัญญาณ Shelf Address

กำหนด shelf address ด้วยแรงดันไฟฟ้าที่ขาสัญญาณนี้เทียบกับ V_{out}(-) ดังแสดงในตารางที่ 2.3 ในโครงงานนี้กำหนดให้มีค่าเป็น 1 โดยต่อขาสัญญาณเขากับ V_{out}(-)

ตารางที่ 2.3 แสดงความสัมพันธ์ระหว่างแรงดันไฟฟ้าที่ขาสัญญาณกับค่า shelf address

Shelf	1	2	3	4	5	6	7	8
Maximum voltage	3.45	2.97	2.56	2.14	1.70	1.25	0.8	0.25
Nominal voltage	3.30	2.86	2.40	1.96	1.50	1.10	0.6	0.01
Minimum voltage	3.00	2.60	2.18	1.73	1.29	0.84	0.3	0
Address bit-A2	0	1	0	1	0	1	0	1

เมื่อนำ address บิต A2 A1 A0 มารวมกันจะสามารถกำหนด address ของเพาเวอร์ซัพพลายได้แสดงตามตารางที่ 2.4 บิตที่ 0 ของ address byte เป็นตัวกำหนดว่าเป็นการเขียน ('0') หรืออ่าน ('1') Global Broadcast คือ address ที่ทำให้เพาเวอร์ซัพพลายทุกตัวบนบัสตอบสนองต่อคำสั่งเขียน

ตารางที่ 2.4 แสดงการระบุ address ของเพาเวอร์ซัพพลาย CP2725AC54TE

	Address bit							
	7	6	5	4	3	2	1	0
PCA9541	1	1	1	0	A2	A1	A0	R/W
Micro controller	1	0	0	0	A2	A1	A0	R/W
External EEPROM	1	0	1	0	A2	A1	A0	R/W
Global Broadcast	0	0	0	0	0	0	0	0

2.1.2.2 สัญญาณสถานะ

สัญญาณ Power Capacity

โลจิก 'HI' ระบุได้ว่าเพาเวอร์ซัพพลายทำงานที่กำลังงานเอ้าพุท 2725W โลจิก 'LO' ระบุได้ว่าเพาเวอร์ซัพพลายทำงานที่กำลังงานเอ้าพุท 1200W

สัญญาณ Power Fail Warning

สถานะเป็นโลจิก 'HI' เมื่อเอ้าพุทเปิดใช้งาน เปลี่ยนเป็นโลจิก 'LO' ก่อนที่ระดับแรงดันเอ้าพุทลดต่ำกว่า $40V_{DC}$ ประมาณ 5ms

สัญญาณ Fault

สัญญาณนี้จะเปลี่ยนเป็นโลจิก 'LO' เมื่อเกิดการทำงานล้มเหลวที่อาจเนื่องมาจากสาเหตุบางอย่าง เช่น การทำงานของพัดลมระบายความร้อนล้มเหลว, เกิดสัญญาณเตือนเนื่องจากอุณหภูมิสูงเกินกำหนด (over-temperature warning), เกิดการหยุดการทำงานเนื่องจากอุณหภูมิสูงเกินกำหนด (over-

temperature shutdown), เกิดการหยุดทำงานเนื่องจากแรงดันไฟฟ้าสูงเกินกำหนด (over-voltage shutdown), ความล้มเหลวในการทำงานของอุปกรณ์ภายใน (internal rectifier fault)

2.1.3 การเปลี่ยนแปลงสถานะ

การเปลี่ยนแปลงสถานะเป็นตัวบ่งบอกว่ามีเหตุการณ์เกิดขึ้นที่ระบบควบคุมเพาเวอร์ซัพพลายควรจะรับรู้ การเปลี่ยนแปลงสถานะจะเกิดขึ้นเนื่องจากเหตุการณ์ดังนี้

- เริ่มต้นการใช้งานโดยเสียบปลั๊กไฟกระแสสลับเข้ากับเพาเวอร์ซัพพลาย เอ้าพุทจะถูกเปิดใช้งาน ทำให้เกิดการเปลี่ยนแปลงสถานะ
- เมื่อมีการ hot-plugged เป็นการแจ้งเตือนระบบควบคุมว่ามีเพาเวอร์ซัพพลายตัวใหม่เข้ามาในระบบ
- เกิดการเปลี่ยนของบิตในรีจิสเตอร์สถานะ

2.1.4 คุณสมบัติทางดิจิตอล

PMBus compliance

เพาเวอร์ซัพพลายรุ่นนี้รองรับข้อกำหนด Power Management Bus (PMBus) rev1.2 แต่มีข้อยกเว้นเรื่องการแจ้งเตือนความผิดพลาด ตัวอย่างเช่น ถ้าเกิดเหตุการณ์ไฟกระชากทำให้เพาเวอร์ซัพพลายหยุดการทำงานชั่วคราวแล้วจึงเริ่มการทำงานใหม่ด้วยตัวเอง ในขณะที่หยุดการทำงานไปชั่วคราว เพาเวอร์ซัพพลายมีการแจ้งเตือนระบบที่ควบคุมเพาเวอร์ซัพพลายว่าเกิดการเปลี่ยนแปลงสถานะของเพาเวอร์ซัพพลาย ถ้าระบบที่ควบคุมสั่งอ่านค่าสถานะจากเพาเวอร์ซัพพลายในขณะที่ยังหยุดการทำงานชั่วคราวอยู่ ระบบที่ควบคุมจะอ่านค่าสถานะได้ถูกต้องตามความเป็นจริงว่าเกิดไฟกระชากและเพาเวอร์ซัพพลายหยุดทำงาน แต่ถ้าระบบควบคุมเรียกอ่านสถานะหลังจากที่เพาเวอร์ซัพพลายเริ่มการทำใหม่แล้ว ระบบที่ควบคุมจะอ่านสถานะได้ว่าเพาเวอร์ซัพพลายทำงานเป็นปกติและจะไม่ทราบถึงเหตุผลของการเริ่มการทำงานใหม่ เนื่องจากเพาเวอร์ซัพพลายมีการอัพเดทสถานะอยู่ตลอดเวลา

ในโครงงานนี้จึงออกแบบโปรแกรมให้เรียกอ่านสถานะจากเพาเวอร์ซัพพลายทันทีที่มีการเปลี่ยนแปลง

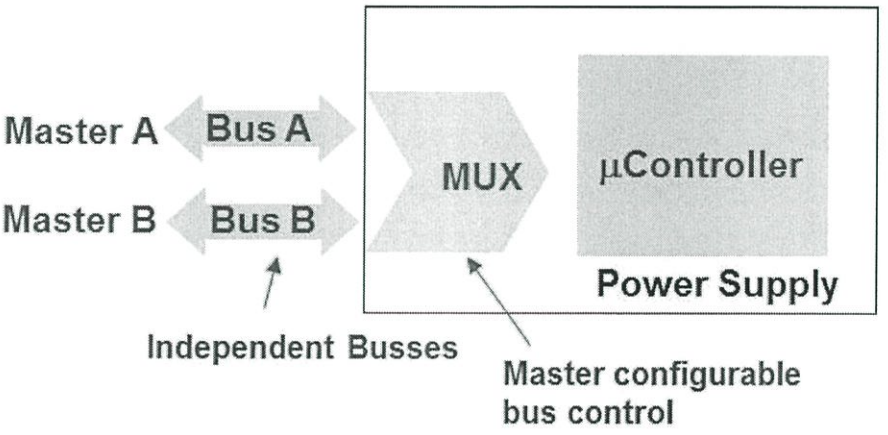
Packet Error Checking (PEC)

PEC เป็นวิธีการตรวจสอบข้อมูลผิดพลาดแบบ CRC-8 โดยใช้พหุนาม $C(x) = x^8+x^2+x+1$ และสอดคล้องกับข้อกำหนดของ PMBus การคำนวณจะคำนวณไบต์ข้อมูลทั้งหมดรวมทั้ง address และไบต์คำสั่ง PEC จะต้องถูกส่งต่อท้ายทุกข้อความโดยอุปกรณ์ที่เป็นตัวส่งไบต์สุดท้าย

ในการสื่อสารกับเพาเวอร์ซัพพลาย CP2725AC54TE ในโหมด PMBus ถ้าไม่มีการส่ง PEC เพาเวอร์ซัพพลายจะไม่ตอบสนองต่อคำสั่ง PMBus ใดๆเลย

SMBusAlert#

เพาเวอร์ซัพพลาย CP2725AC54TE สามารถส่งสัญญาณ SMBAlert# ได้ทั้งจากไมโครคอนโทรลเลอร์หรือ PCA9541 I^2C bus master selector ภายในเพาเวอร์ซัพพลาย กล่าวคือสัญญาณที่ส่งออกจากไมโครคอนโทรลเลอร์ภายในจะส่งผ่าน PCA9541 แล้วแยกออกเป็นสองสัญญาณตามภาพที่ 2.2 แล้วส่งไปยังขาสัญญาณ ALERT# 0 และ ALERT# 1 ที่ด้านหลังของเพาเวอร์ซัพพลาย



ภาพที่ 2.2 แผนภาพแสดงโครงสร้างของบัส I^2C แบบคู่

ที่มา : General Electric Company CP2725AC54TE Compact Power Line High Efficiency Rectifier, September 21, 2016 : Page 10

SMBAlert# ที่ส่งจากไมโครคอนโทรลเลอร์ภายในเพาเวอร์ซัพพลาย

สัญญาณ SMBAlert# แจ้งให้ระบบควบคุมทราบถึงการเปลี่ยนของสถานะของเพาเวอร์ซัพพลายในภาวะปกติสัญญาณมีสถานะเป็นโลจิก 'HI' และเป็นเป็นโลจิก 'LO' เมื่อเพาเวอร์ซัพพลายมีการเปลี่ยนแปลงสถานะ และจะดึงสัญญาณค้างไว้จนกว่าจะได้รับการจัดการจากระบบควบคุมตามที่ระบุไว้ดังนี้

- ได้รับคำสั่งอ่านสถานะ
- ได้รับคำสั่ง CLEAR_FAULTS
- รีเซ็ตเอ้าพุทโดยใช้ขา Enable
- รีเซ็ตเอ้าพุทโดยใช้คำสั่ง PMBus

สัญญาณ SMBAlert# จะเปลี่ยนสถานะเนื่องจากสภาวะดังต่อไปนี้

V_{IN} under or over voltage

V_{OUT} under or over voltage

I_{OUT} over current

Over Temperature warning or fault

Fan Failure

Communication error

PEC error

Invalid command

Internal faults

Read back delay

เพื่อหลีกเลี่ยงการอ่านสถานะที่กำลังเปลี่ยนแปลงซึ่งอาจทำให้เกิดสัญญาณ SMBAlert# อย่างต่อเนื่องจนกว่าสถานะจะเปลี่ยนแปลงสิ้นสุด ระบบควบคุมควรสั่งอ่านสถานะหลังจากได้รับสัญญาณ SMBAlert# 2 วินาที เพื่อให้ค่าที่อ่านได้มีความถูกต้อง

Successive read backs

การสั่งอ่านสถานะอย่างต่อเนื่องจากเพาเวอร์ซัพพลาย CP2725AC54TE ควรเว้นช่วงเวลาอย่างน้อย 1 วินาที เพื่อให้เพาเวอร์ซัพพลายได้มีการอัพเดทข้อมูลใหม่ได้ทันเวลา

2.1.5 PMBus Command (PMBus protocol หัวข้อ 2.5)

2.1.5.1 Standard command

คือ รหัสคำสั่งมาตรฐานที่ระบุไว้ในข้อกำหนดโปรโตคอล PMBus ในดาต้าชีทของเพาเวอร์ซัพพลาย CP2725AC54TE ระบุรหัสคำสั่งที่ใช้กับเพาเวอร์ซัพพลายไว้ดังนี้

Operation (0x01)

ถ้าขาสัญญาณ Enable มีสถานะโลจิก 'LO' โดยการตั้งค่าเริ่มต้นเอ้าพุทจะเปิดใช้งานเมื่อต่อเข้ากับไฟฟ้าแหล่งพลังงาน คำสั่ง Operation ใช้เพื่อเปิด/ปิดเอ้าพุทผ่าน PMBus command ข้อมูลที่ส่งตามรหัสคำสั่งนี้มี 1 ไบต์ คือ 0x80 เพื่อเปิดใช้งานเอ้าพุท, 0x00 เพื่อปิดใช้งานเอ้าพุท

Clear_faults (0x03)

คำสั่งนี้ใช้ในการรีเซ็ตข้อมูลแต่ละบิทในรีจิสเตอร์สถานะ โดยไม่ต้องมีข้อมูลตามหลังจากรหัสคำสั่ง ข้อมูลบิทสถานะที่จะถูกรีเซ็ตประกอบด้วย Isolation OK, Isolation test failed, Restarted OK, Invalid command, Invalid data, PEC error

Vout_command (0x21)

คำสั่งนี้ใช้เพื่อเปลี่ยนค่าเอ้าพุทของเพาเวอร์ซัพพลาย สำหรับการกำหนดแรงดันไฟฟ้าเอ้าพุทด้วยโปรแกรมจะเขียนทับค่าที่ถูกกำหนดด้วยแรงดันไฟฟ้าที่ขา Margin และจะมีผลไปจนกระทั่งไมโครคอนโทรลเลอร์ภายในเพาเวอร์ซัพพลายรีสตาร์ทหรือหยุดทำงาน ค่าแรงดันเอ้าพุทที่สามารถกำหนดได้จะอยู่ในช่วง 42 – 58V_{DC} หลังจากรหัสคำสั่งจะตามด้วยข้อมูล 2 ไบต์ ซึ่งได้มาจากการคำนวณสมการที่(1)โดยใช้ค่าคงที่จากตารางที่ 2.5 โดยที่ X คือค่าแรงดันไฟฟ้าที่เราต้องการ

$$y = (mX+b)10^{R} \quad (1)$$

y คือ ค่าที่ส่งไป/รับจากเพาเวอร์ซัพพลาย

X คือ ค่าจริงของข้อมูลที่ต้องการส่งหรือรับจากเพาเวอร์ซัพพลาย

*ยกเว้น input voltage กับ fan RPM, X คือค่าที่อ่านได้จากเพาเวอร์ซัพพลาย

ตารางที่ 2.5 แสดงค่าคงที่ที่ใช้กับสมการที่ (1)

Function	Operation	m	b	R
Output voltage Output voltage shutdown	Write/read	400	0	0
Output Current	Read	5	0	0
Temperature	Read	1	0	0
Input voltage	Read	1	75	0
Input Power	Read	1	0	0
Fan speed setting (%)	Read	1	0	0
Fan speed RPM	Read	100	0	0

จากข้อมูลในตารางที่ 2.5 จะนำไปใช้ประโยชน์ต่อในการนำข้อมูลเหล่านี้ที่อ่านค่าได้จากเพาเวอร์ซัพพลายมาแสดงผล ซึ่งจะนำสมการที่ (1) และข้อมูลจากตารางที่ 2.5 ไปเขียนโปรแกรมคำนวณให้ได้ค่าจริงก่อนที่จะแสดงผลให้ผู้ใช้งาน

Vout_OV_fault_limit (0x40)

คำสั่งนี้ใช้เพื่อกำหนดค่าที่เพาเวอร์ซัพพลายจะปิดการทำงานเมื่อแรงดันเอ้าพุทมีค่าถึงที่กำหนดไว้ การส่งคำสั่งทำเหมือนกันกับ Vout_command (0x21)

2.1.5.2 Manufacturer-specific command

คือ คำสั่ง PMBus ที่ใช้ได้เฉพาะกับอุปกรณ์ของผู้ผลิตนั้นๆ ในที่นี้คือเพาเวอร์ซัพพลายรุ่น CP2725AC54TE คำสั่งอ่านข้อมูลจากเพาเวอร์ซัพพลายจะได้รับข้อมูลกลับมาได้รูปแบบเลขฐาน 16 คำสั่งเฉพาะมีให้ใช้ทั้งหมด 17 คำสั่งดังนี้

Read_status (0xD0)

เป็นคำสั่งอ่านข้อมูลสถานะของเพาเวอร์ซัพพลาย CP2725AC54TE เพาเวอร์ซัพพลายจะส่งข้อมูลกลับมาให้ระบบควบคุมทั้งหมด 10 ไบต์(รวม PEC) ประกอบด้วย จำนวนข้อมูล(byte count), ข้อมูลใน Status-2 รีจิสเตอร์, ข้อมูลใน Status-1 รีจิสเตอร์, ข้อมูลใน Alarm-2 รีจิสเตอร์, ข้อมูลใน Alarm-1 รีจิสเตอร์, แรงดันไฟฟ้าเอ้าพุท LSB, แรงดันไฟฟ้าเอ้าพุท MSB, กระแสไฟฟ้าเอ้าพุท, อุณหภูมิภายในเพาเวอร์ซัพพลายและ PEC ตามลำดับ ซึ่งค่าแรงดันไฟฟ้าเอ้าพุทและกระแสไฟฟ้าเอ้าพุทที่อ่านได้ต้องนำไปคำนวณจากสมการที่(1)และตารางที่ 2.5 เพื่อให้ได้ค่าจริงก่อนจะนำมาแสดงผลให้ผู้ใช้งาน ส่วนข้อมูลใน status และ alarm รีจิสเตอร์จะมีความแตกต่างจากที่ระบุไว้ในข้อกำหนดโปรโตคอล PMBus ข้อมูลภายในรีจิสเตอร์แสดงดังตารางที่ 2.6, 2.7, 2.8 และ 2.9 โดยสถานะปกติมีค่าโลจิก '0' และสถานะเมื่อเกิด alarm มีค่าโลจิก '1'

ตารางที่ 2.6 แสดงข้อมูลภายใน Status-2 รีจิสเตอร์

Bit	Title	Description
7	PEC Error	Mismatch between computed and transmitted PEC. The instruction has not been executed. Clear_Flags resets this register
6	Will Restart	Restart after a shutdown = 1
5	Invalid Instruction	The instruction is not supported. An ALERT# will be issued. Clear_Flags resets this register.
4	Power Capacity	High line power capacity = 1
3	Isolation test failed	Information only to system controller
2	Restarted ok	Informs HOST that a successful RESTART occurred clearing the status and alarm registers
1	Data out of range	Flag appears until the data value is within range. A clear_flags command does not reset this register

		until the data is within normal range
0	Enable pin HI	State of the ENABLE pin, HI = 1 = OFF

ตารางที่ 2.7 แสดงข้อมูลภายใน Status-1 รีจิสเตอร์

Bit	Title	Description
7	spare	
6	Isolation test OK	Isolation test completed successfully
5	Internal fault	The power supply is faulty
4	Shutdown	
3	Service LED ON	ON = 1
2	External fault	the power supply is functioning OK
1	LEDs flashing	LEDs tested test ON = 1
0	Output ON	ON = 1

ตารางที่ 2.8 แสดงข้อมูลภายใน Alarm-2 รีจิสเตอร์

Bit	Title	Description
7	Fan Fault	
6	No primary	No primary detect
5	Primary OT	Primary section OT
4	DC/DC OT	DC/DC section OT
3	Output voltage lower than bus	Internal regulation failure
2	Thermal sensor failed	Internal failure of a temperature sensing circuit
1	5V out_of_limits	Either OVP or OCP occurred
0	Power delivery	a power delivery fault occurred

ตารางที่ 2.9 แสดงข้อมูลภายใน Alarm-1 รีจิสเตอร์

Bit	Title	Description
7	Unit in power limit	An overload condition that results in constant power
6	Primary fault	Indicates either primary failure or INPUT not present. Used in

		conjunction with bit-0 and Status_1 bits 2 and 5 to assess the fault.
5	Overtemp. shutdown	One of the over_temperature sensors tripped the supply
4	Over temp. warning	Temperature is too high, close to shutdown
3	In over current	Shutdown is triggered by low output voltage < 39VDC.
2	Over voltage shutdown	
1	Vout out_of_limit	Indication the output is not within design limits. This condition may or may not cause an output shutdown
0	Vin out_of_limit	The input voltage is outside design limits

LEDS test ON (0xD2)

คำสั่งนี้ทำให้ไฟ LED ทั้งหมดที่ด้านหน้าเพาเวอร์ซัพพลายติดเป็นเวลา 7 วินาทีและดับ 2 วินาที สลับกันไปเรื่อยๆจนกว่าจะมีคำสั่งให้ปิดไฟ LED จุดประสงค์ของคำสั่งคือการทดสอบการสื่อสารระหว่างเพาเวอร์ซัพพลายกับระบบควบคุม

LEDS test OFF (0xD3)

คำสั่งนี้ทำให้ไฟ LED ที่ด้านหน้าเพาเวอร์ซัพพลายดับลง

Service LED ON (0xD4)

คำสั่งนี้ทำให้ Service LED กระพริบเป็นเวลา 0.5 วินาทีจนกว่าจะมีคำสั่งให้ปิด

Service LED OFF(0xD5)

คำสั่งปิด Service LED

Read input string (0xDC)

คำสั่งอ่านค่าแรงดันไฟฟ้าอินพุทและกำลังไฟฟ้าอินพุท เพาเวอร์ซัพพลายจะส่งข้อมูลกลับมา 5 ไบต์(รวม PEC) ประกอบด้วย จำนวนข้อมูล (byte count), แรงดันไฟฟ้าอินพุท, กำลังไฟฟ้าอินพุท LSB, กำลังไฟฟ้าอินพุท MSB และ PEC ตามลำดับ ซึ่งค่าแรงดันไฟฟ้าและกำลังไฟฟ้าที่อ่านได้ต้องนำไปคำนวณจากสมการที่(1)และตารางที่ 2.5 เพื่อให้ได้ค่าจริงก่อนจะนำมาแสดงผลให้ผู้ใช้งาน

Fan_speed_set (0xDF)

คำสั่งนี้ใช้กำหนดความเร็วพัดลมของเพาเวอร์ซัพพลายได้ตั้งแต่ 0 – 100% คำสั่งนี้สามารถเพิ่มความเร็วพัดลมได้อย่างเดียวไม่สามารถสั่งให้ลดความเร็วพัดลมต่ำกว่าที่เพาเวอร์ซัพพลายกำหนดไว้ ข้อมูลที่ส่งจะต้องส่งในรูปของเลขฐาน 16 1 ไบต์ (0% = 0x00, 100% = 0x64) ต่อจากรหัสคำสั่ง

Fan_normal_speed (0xE0)

คำสั่งรีเซ็ตความเร็วรอบพัดลมให้กลับสู่ค่าเริ่มต้นของเพาเวอร์ซัพพลาย

Read_Fan_speed (0xE1)

คำสั่งอ่านค่าความเร็วรอบพัดลมจากเพาเวอร์ซัพพลาย เพาเวอร์ซัพพลายจะส่งข้อมูลกลับมา 6 ไบต์(รวม PEC) ประกอบด้วย ความเร็วรอบพัดลมในหน่วยเปอร์เซ็น, ความเร็วรอบพัดลมตัวที่หนึ่งในหน่วยรอบต่อนาที, ความเร็วรอบพัดลมตัวที่สองในหน่วยรอบต่อนาที, ความเร็วรอบพัดลมตัวที่สามในหน่วยรอบต่อนาทีและ PEC ถ้าพัดลมตัวไหนไม่ได้ถูกติดตั้งไว้ค่าที่อ่านได้จะเป็น 0 (0x00) ซึ่งค่าความเร็วพัดลมในหน่วยรอบต่อนาทีที่อ่านได้ต้องนำไปคำนวณจากสมการที่(1)และตารางที่ 2.5 เพื่อให้ได้ค่าจริงก่อนจะนำมาแสดงผลให้ผู้ใช้งาน

2.1.6 การจัดการความผิดพลาด (Fault Mangement)

เพาเวอร์ซัพพลาย CP2725AC54TE แยกความผิดพลาดออกเป็น 2 แบบ คือ ความผิดพลาดที่เกิดจากภายในเพาเวอร์ซัพพลายเอง (internal faults) และความผิดพลาดที่เกิดจากปัจจัยภายนอก (external faults)สำหรับการแจ้งเตือนความผิดพลาดที่เกิดจากภายในเพาเวอร์ซัพพลายจะใช้ไฟ Fault LED, ขาสัญญาณ Fault และขาสัญญาณ ALERT# 0,1 เป็นตัวบ่งบอก ความผิดพลาดที่เกิดจากภายนอกจะไม่สามารถระบุได้โดยใช้วิธีที่กล่าวมา จากข้อมูลนี้และตารางที่ 2.6 – 2.9 จะทำให้สามารถทดสอบโปรแกรมตรวจสอบความผิดพลาดที่เกิดจากภายในได้ด้วยขาสัญญาณ ALERT#

2.2 บอร์ดไมโครคอนโทรลเลอร์ BeagleBone Green

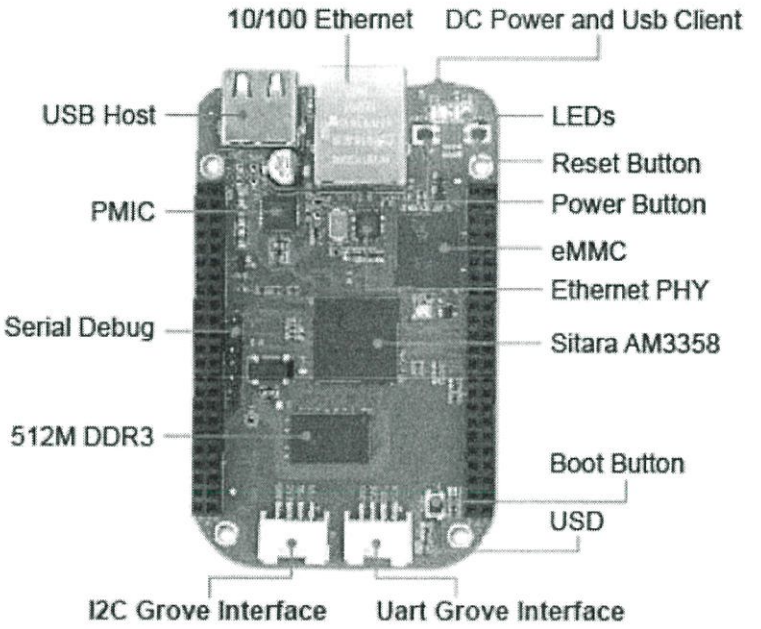
BeagleBone เป็นแพลตฟอร์มคอมพิวเตอร์แบบ Linux โอเพนซอร์สที่มีราคาถูกและสามารถใช้ในการสร้างแอพพลิเคชันที่ซับซ้อนซึ่งอินเตอร์เฟซกับซอฟต์แวร์ระดับสูงและวงจรอิเล็กทรอนิกส์ระดับต่ำได้

เป็นแพลตฟอร์มที่เหมาะสำหรับโครงการต้นแบบและการออกแบบผลิตภัณฑ์ที่ใช้ประโยชน์จากระบบปฏิบัติการ Linux อีกทั้งยังเข้าถึงการใช้งานขา GPIO และบัส (bus) ที่ง่ายช่วยให้สามารถอินเตอร์เฟซกับอุปกรณ์อิเล็กทรอนิกส์และอุปกรณ์ USB สะดวกยิ่งขึ้น [2]

ในโครงงานนี้บอร์ด BeagleBone Green จะใช้เป็นระบบควบคุมการสื่อสารกับเพาเวอร์ซัพพลาย CP2725AC54TE โดยการเขียนโปรแกรมสำหรับการสื่อสารด้วย PMBus ให้ทำงานบนบอร์ด BeagleBone Green ในหัวข้อที่ 2.2 นี้จะเป็นศึกษาข้อมูลฮาร์ดแวร์และซอฟแวร์ของบอร์ด BBG เพื่อเป็นข้อมูลในการเตรียมการอินเตอร์เฟซกับเพาเวอร์ซัพพลาย CP2725AC54TE

2.2.1 BeagleBone Green ฮาร์ดแวร์

ในหัวข้อนี้จะกล่าวถึงฮาร์ดแวร์ที่ใช้สำหรับการอินเตอร์เฟซกับเพาเวอร์ซัพพลาย CP2725AC54TE ภาพที่ 2.3 แสดงฮาร์ดแวร์เบื้องต้นของบอร์ด BeagleBone Green และภาพที่ 2.4 แสดงขา GPIO header P9 และ P8



ภาพที่ 2.3 แสดงฮาร์ดแวร์และอินเตอร์เฟซของ BeagleBone Green

ที่มา : http://wiki.seeed.cc/BeagleBone_Green/

I2C

Note

The first I2C bus is utilized for reading EEPROMS on cape add-on boards and can't be used for other digital I/O operations without interfering with that function, but you can still use it to add other I2C devices at available addresses. The second I2C bus is available for you to configure and use.

I2C

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3_3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUT	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
I2C1_SCL	17	18	I2C1_SDA	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
I2C2_SCL	21	22	I2C2_SDA	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	I2C1_SCL	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	I2C1_SDA	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_123	GPIO_86	27	28	GPIO_88
GPIO_121	29	30	GPIO_122	GPIO_87	29	30	GPIO_89
GPIO_120	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

ภาพที่ 2.4 แสดงขา GPIO header P9 และ P8 ของ BeagleBone Green

ที่มา : http://wiki.seeed.cc/BeagleBone_Green/

ในการอินเตอร์เฟซกับเพาเวอร์ซัพพลาย CP2725AC54TE จะใช้งาน GPIO ทั้งหมด 5 ขา ดังนี้

I2C2_SCL (ขา 19 header P9) สำหรับ I^2C clock line

I2C2_SDA (ขา 20 header P9) สำหรับ I^2C data line

GPIO_49 (ขา 23 header P9) สำหรับรับสัญญาณ SMBAlert# จากเพาเวอร์ซัพพลาย

VDD_3V3 สำหรับโลจิก V_{DD}

DGND สำหรับโลจิก GND

เนื่องจากบอร์ด BeagleBone ไม่มีอุปกรณ์ต่อพ่วงที่ทำให้ผู้ใช้งานควบคุมบอร์ดได้โดยตรง จึงต้องใช้งานผ่านคอมพิวเตอร์เดสก์ท็อป โดยใช้สาย USB ที่เป็นทั้งตัวจ่ายพลังงานให้กับบอร์ดและใช้ในการสื่อสารผ่านเครือข่ายกับคอมพิวเตอร์เดสก์ท็อป (internet-over-USB)

2.2.2 BeagleBone Green ซอฟแวร์

ในหัวข้อนี้จะศึกษาเกี่ยวกับระบบปฏิบัติการ Linux ที่ติดตั้งบนบอร์ด BeagleBone และ Cloud9 Integrate Development Environment ที่ใช้ในการเขียนโปรแกรมควบคุมเพาเวอร์ซัพพลายสำหรับโครงงานนี้

2.2.2.1 ระบบปฏิบัติการ Linux

ระบบปฏิบัติการ คือ ชุดของโปรแกรมและเครื่องมือพื้นฐานที่ทำให้คอมพิวเตอร์ทำงานได้ แกนหลักของระบบปฏิบัติการคือเคอร์เนล (kernel) เคอร์เนล คือ โปรแกรมพื้นฐานที่สุดที่ทำให้โปรแกรมอื่นทำงานได้

Linux distribution เป็น Linux เวอร์ชันที่เผยแพร่แบบสาธารณะซึ่งจะมาพร้อมกับโปรแกรมและเครื่องมือ ต่างๆ Linux distribution มีอยู่หลายแบบ ซึ่งโดยทั่วไปจะเน้นการใช้งานที่แตกต่างกัน ตัวอย่างเช่นเจ้าของเซิฟเวอร์ระดับไฮเอนด์อาจติดตั้ง Red Hat Enterprise, Debian หรือ OpenSUSE; ผู้ใช้งานเดสก์ท็อปอาจติดตั้ง Ubuntu, Debian, Fedora หรือ Linux Mint และกรณีอื่นๆอีกมากมาย แต่สิ่งที่ทุก distribution มีเหมือนกันคือ Linux kernel ที่ถูกพัฒนาขึ้นโดย Linux Torvalds

ในโครงงานนี้บอร์ด BeagleBone Green ติดตั้ง Debian GNU/Linux เวอร์ชัน 4.4.9-ti-r25 มาพร้อมกับบอร์ดตั้งแต่แรก Debian เป็นหนึ่งใน Linux distribution ที่ใช้ linux kernel และเครื่องมือพื้นฐานของ GNU ประกอบรวมกันเป็นระบบปฏิบัติการ ซึ่งผู้จัดทำเห็นว่าเหมาะสมกับการทดลองในโครงงานนี้อยู่แล้วเพราะว่ามีไดร์เวอร์สำหรับ I^2C ให้ใช้งาน จึงไม่มีความจำเป็นต้องเปลี่ยนระบบปฏิบัติการ

2.2.2.2 แอพพลิเคชั่นซอฟแวร์

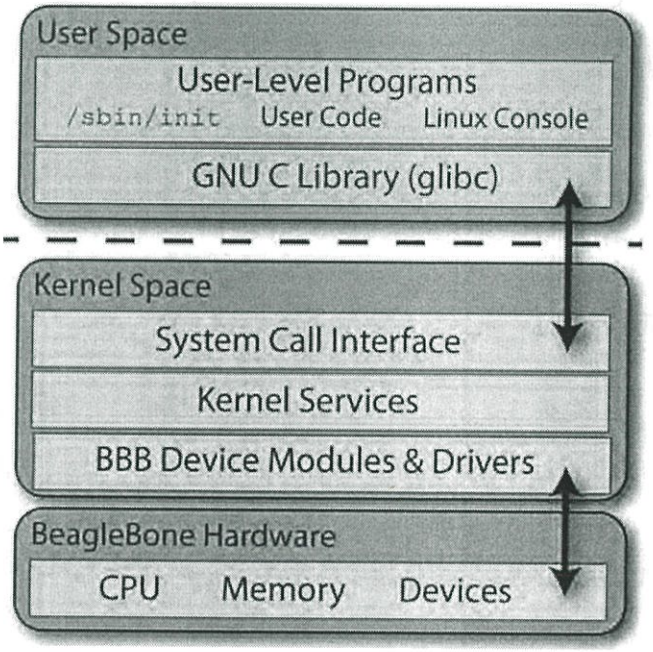
ยูสเซอร์สเปซและ**เคอร์เนลสเปซ** (kernel space and user space)[2]

เคอร์เนลสเปซ คือบริเวณของหน่วยความจำที่โปรแกรมฝั่งเคอร์เนลสเปซทำงานอยู่ ซึ่งประกอบไปด้วยตำแหน่งหน่วยความจำจริง (Physical memory) และตำแหน่งหน่วยความจำเสมือน (Kernel virtual memory) หน่วยความจำส่วนนี้จะไม่ถูกย้ายลงไปเก็บในดิสก์และไม่สามารถเรียกใช้ได้โดยตรงจากโปรแกรมฝั่งยูสเซอร์สเปซเนื่องจากเหตุผลเรื่องความปลอดภัยของระบบ หน่วยความจำส่วนนี้ครอบคลุมฮาร์ดแวร์ทั้งหมดซึ่งรวมถึงรีจิสเตอร์ต่างๆด้วย

โปรแกรมฝั่งเคอร์เนลสเปซ คือโค้ดที่ทำหน้าที่ติดต่อและควบคุมฮาร์ดแวร์ต่างๆ รวมถึงจัดการทรัพยากรที่มีในระบบทั้งหมดตามที่โปรแกรมฝั่งยูสเซอร์สเปซร้องขอ

ยูสเซอร์สเปซ คือบริเวณของหน่วยความจำที่โปรแกรมฝั่งยูสเซอร์สเปซทำงานอยู่ ซึ่งเป็นตำแหน่งหน่วยความจำเสมือน (Virtual memory) ซึ่งถูกจัดสรรโดยเคอร์เนล หน่วยความจำส่วนนี้ อาจถูกย้ายไปเก็บใน swap space บนดิสก์ ขึ้นกับระดับความสำคัญที่ระบบปฏิบัติการตัดสินใจ

โปรแกรมฝั่งยูสเซอร์สเปซ คือโค้ดที่ทำหน้าติดต่อกับผู้ใช้ รับผิดชอบประมวลผลสำหรับงานในส่วนที่ยูสเซอร์ต้องการ ส่วนโปรแกรมแอพพลิเคชั่นต่างๆที่ทำงานกับฮาร์ดแวร์นั้น จริงๆแล้วไม่ได้ควบคุมฮาร์ดแวร์นั้นโดยตรง แต่ทำงานโดยผ่านไดร์เวอร์ของอุปกรณ์นั้นอีกทีตามภาพที่ 2.5 โดยปกติการพัฒนาซอฟแวร์บน Linux ส่วนใหญ่จะอยู่ในส่วนยูสเซอร์สเปซเป็นหลัก



ภาพที่ 2.5 แผนภาพแสดงโครงสร้างของ Linux ยูสเซอร์สเปซและ**เคอร์เนลสเปซ**

ที่มา : Exploring BeagleBone, 2015 : Page 63

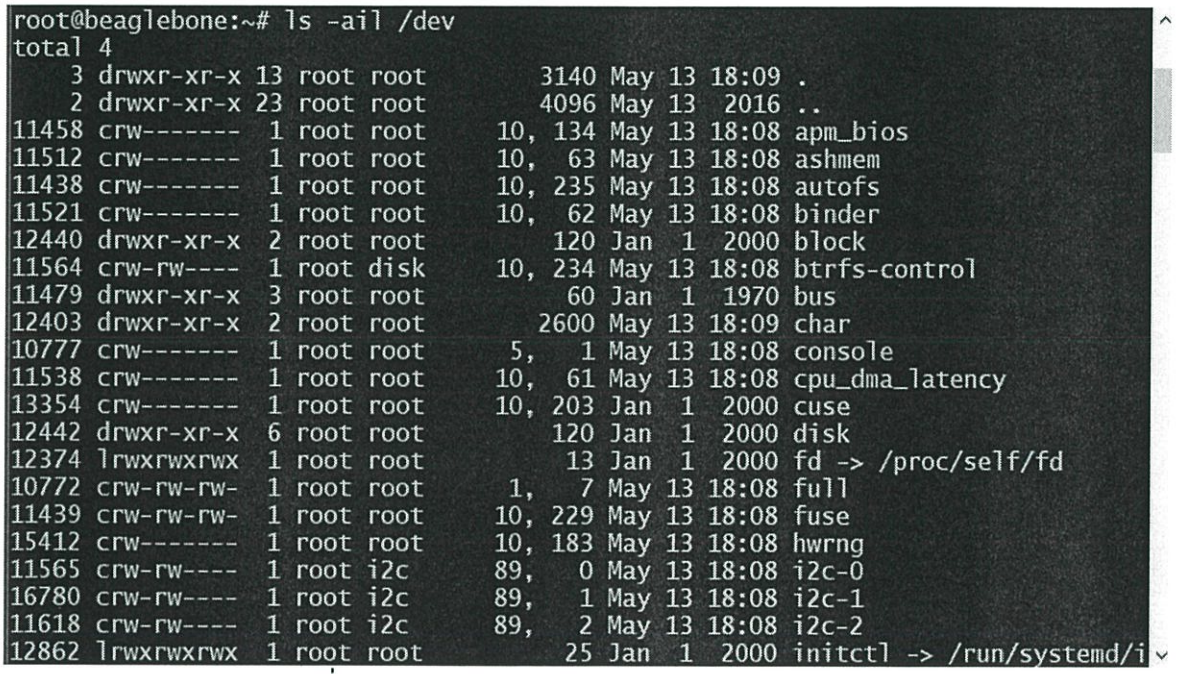
2.2.2.3 ระบบแฟ้มอุปกรณ์ (Linux file system)[2]

ทุกอย่างที่อยู่ในระบบลินุกซ์จะเข้าถึงได้ผ่านระบบแฟ้ม หรือ file system การออกแบบระบบลินุกซ์ดีไวซ์ไดร์เวอร์ทำให้ต้องมีไฟล์ที่เป็นตัวแทนของอุปกรณ์นั้นอยู่บนระบบไฟล์ สำหรับให้โปรแกรมฝั่ง user space ติดต่อได้ เราเรียกไฟล์เหล่านี้ว่า device files

Device โดยปกติจะถูกรวบรวมไว้ภายใต้ /dev/ ของระบบแฟ้มของลินุกซ์ จัดเป็นไฟล์ชนิดพิเศษของระบบลินุกซ์ตามภาพที่ 2.6

2.2.2.4 System calls[2]

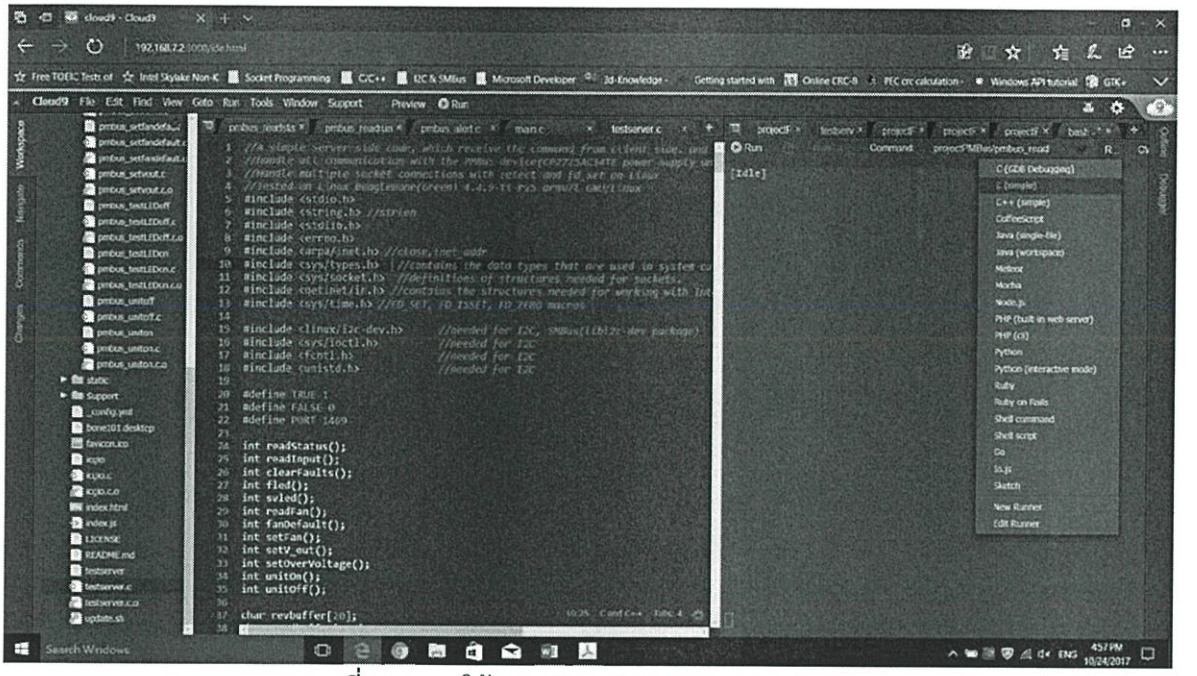
System calls คือ บริการที่เคอร์เนลเตรียมไว้ให้โปรแกรมฝั่งยูสเซอร์สเปซเรียกใช้ โปรแกรมฝั่งยูสเซอร์สเปซต้องร้องขอไปที่เคอร์เนลทั้งหมด เนื่องจากเคอร์เนลเป็นเจ้าของทรัพยากร system calls จะเป็นตัวเชื่อมระหว่าง โปรแกรมฝั่งยูสเซอร์สเปซไปสู่เคอร์เนลเพื่อที่จะส่งต่อไปยังดีไวซ์ไดร์เวอร์ของเราอีกทีหนึ่ง



ภาพที่ 2.6 แสดง device files ของ BeagleBone Green

2.2.2.5 Cloud9 Integrated Development Environment

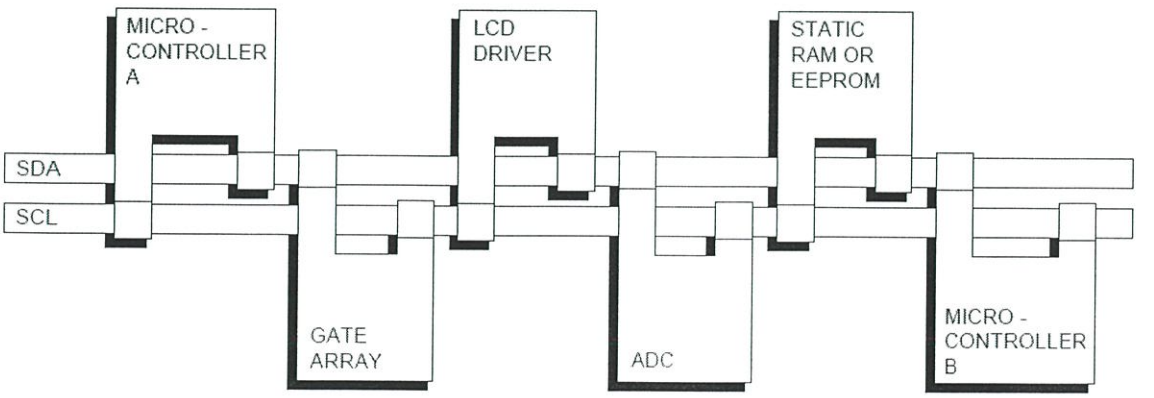
เป็นแพลตฟอร์มการเขียนโค๊ตบนเว็บบราวเซอร์ วิธีการใช้งาน Cloud9 IDE ให้เชื่อมต่อ BeagleBone เข้ากับเดสก์ท็อปคอมพิวเตอร์แล้วเปิดเว็ปบราวเซอร์ในคอมพิวเตอร์ พิมพ์ 192.168.7.2:3000 ที่ address bar จะปรากฏดังภาพที่ 2.7 หน้าต่างด้านซ้าย คือตัวจัดการไฟล์, ตรงกลางคือส่วนที่ใช้เขียนโค๊ดและด้านขวาคือส่วนแสดงเอาท์พุทของโปรแกรม



ภาพที่ 2.7 การใช้งาน Cloud9 IDE บน BeagleBone

2.3 การสื่อสารโดย I^2C protocol [3]

I^2C คือการติดต่อสื่อสารระหว่างไอซีผ่านบัส โดยใช้สายสัญญาณสองเส้น คือสายสัญญาณข้อมูล (Serial Data line, SDA) และสายสัญญาณนาฬิกา (Serial Clock line, SCL) สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-derectional line) ต้องมีการกับไฟเลี้ยงบวกผ่านตัวต้านทานพูลอัพไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการใช้งาน ทั้งยังช่วยป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง แต่ละอุปกรณ์บนบัสจะมี address ที่แตกต่างกัน อุปกรณ์บนบัสจะมีทั้ง master และ slave โดยที่ master เป็นตัวเริ่มการสื่อสารและสร้างสัญญาณนาฬิกาเพื่อติดต่อกับ slave ภาพที่ 2.7 แสดงตัวอย่างการใช้งาน I^2C



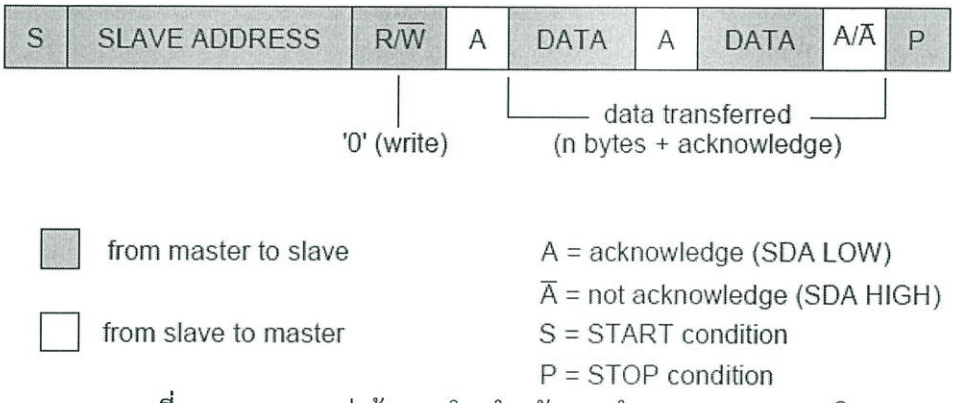
ภาพที่ 2.8 แสดงตัวอย่างการใช้งาน I^2C

ที่มา : NXP Semiconductor UM10204 I2C-bus specification and user manual Rev.6, 2014, page 7

โปรโตคอลการสื่อสาร I^2C สำหรับการกำหนด address 7 บิต มีดังนี้

การส่งข้อมูล

มาสเตอร์ส่งข้อมูลไปที่ slave โดยที่ address บิตที่ 0 ของ slave มีค่าเป็น '0' เป็นการระบุทิศทางข้อมูลว่าเป็นการส่งข้อมูล ภาพที่ 2.8 แสดงการส่งข้อมูลโดย I^2C สำหรับการกำหนด address 7 บิต

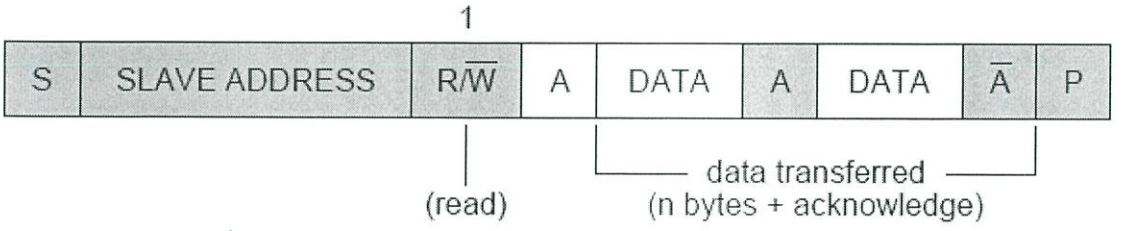


ภาพที่ 2.9 แสดงการส่งข้อมูล I^2C สำหรับการกำหนด address 7 บิต

ที่มา : NXP Semiconductor UM10204 I2C-bus specification and user manual Rev.6, 2014, page 15

การรับข้อมูล

มาสเตอร์อ่านข้อมูลจาก slave โดยที่ address บิตที่ 0 ของ slave มีค่าเป็น '1' เป็นการระบุทิศทางข้อมูลว่าเป็นการรับข้อมูล เมื่อสิ้นสุดการรับข้อมูลมาสเตอร์จะส่งสัญญาณ not acknowledge ภาพที่ 2.9 แสดงการรับข้อมูล I^2C สำหรับการกำหนด address 7 บิต

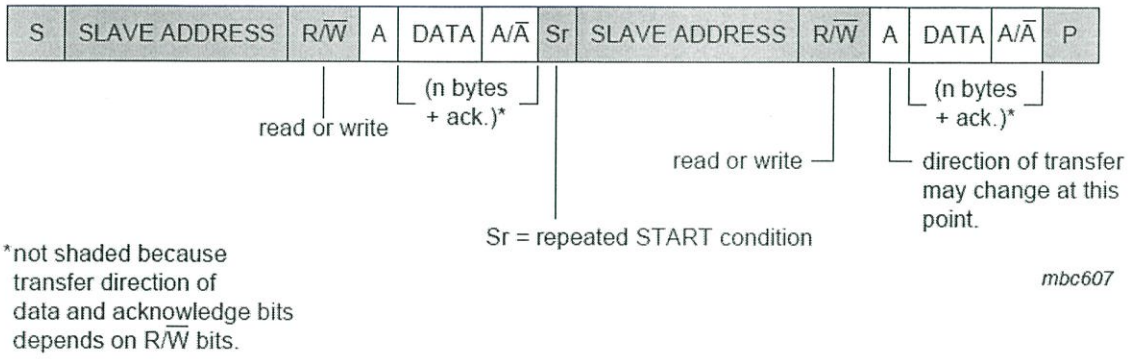


ภาพที่ 2.10 แสดงการรับข้อมูล I^2C สำหรับการกำหนด address 7 บิต

ที่มา : NXP Semiconductor UM10204 I2C-bus specification and user manual Rev.6, 2014, page 15

การสื่อสารแบบผสม

เป็นการสื่อสารที่มีทั้งรับข้อมูลและส่งข้อมูลในการสื่อสารครั้งเดียว โดยเมื่อมีการเปลี่ยนทิศทางของข้อมูลจะเกิดสัญญาณ repeated start แล้วตามด้วย slave address ภาพที่ 2.10 แสดงการสื่อสารแบบผสม I^2C สำหรับการกำหนด address 7 บิต



ภาพที่ 2.11 แสดงการสื่อสารแบบผสม I^2C สำหรับการกำหนด address 7 บิต

ที่มา : NXP Semiconductor UM10204 I2C-bus specification and user manual Rev.6, 2014, page 15

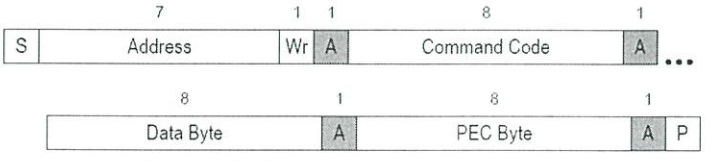
2.4 System Management Bus (SMBus) [4]

SMBus เป็นอินเตอร์เฟซสองสายซึ่งทำให้ชิปและอุปกรณ์ต่างๆของระบบสามารถสื่อสารกันเองและส่วนอื่นๆของระบบได้ด้วยหลักการทำงานของ I^2C bus หนึ่งในข้อแตกต่างระหว่าง SMBus และ I^2C คือ โปรโตคอล ข้อกำหนด I^2C ระบุแค่วิธีการรับส่งข้อมูลระหว่างอุปกรณ์โดยโครงสร้างของข้อความและแพคเกตจะถูกออกแบบโดยผู้ผลิตอุปกรณ์ แต่ข้อกำหนด SMBus กำหนดรูปแบบคำสั่งที่ต้องใช้เมื่ออุปกรณ์ SMBus มีการสื่อสารกัน

2.4.1 SMBus protocol

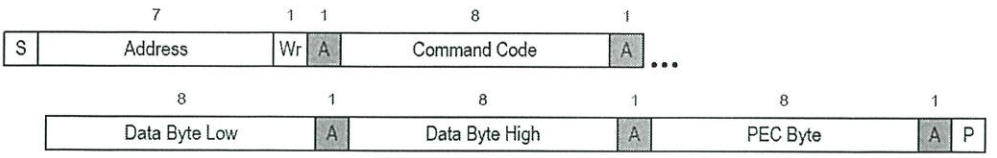
Write Byte/Word

ไบต์แรกถัดจากแอดเดรสไบต์คือรหัสคำสั่งส่วนไบต์ถัดไปจะเป็นข้อมูลต้องการส่ง ภาพที่ 2.11 แสดงโปรโตคอล write byte แบบใช้ PEC และภาพที่ 2.12 แสดงโปรโตคอล write worf แบบใช้ PEC



ภาพที่ 2.12 โปรโตคอล write byte แบบใช้ PEC

ที่มา : System Mangement Bus (SMBus) Specification version 3.0, 2014, page 41

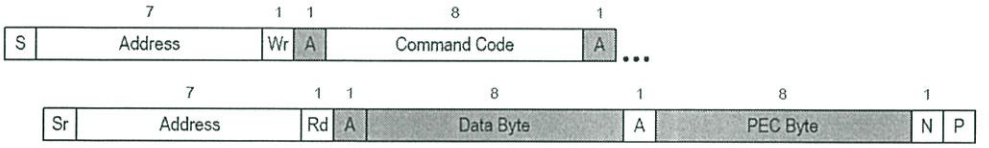


ภาพที่ 2.13 โปรโตคอล write word แบบใช้ PEC

ที่มา : System Mangement Bus (SMBus) Specification version 3.0, 2014, page 41

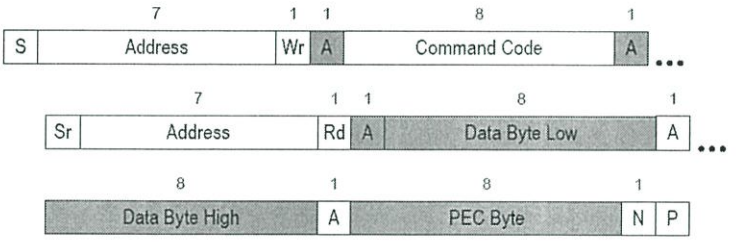
Read Byte/Word

การอ่านข้อมูลจะซับซ้อนกว่าการเขียนข้อมูลเล็กน้อย ในขั้นแรกต้องเขียนรหัสคำสั่งไปที่ slave ก่อนและตามด้วยสถานะ repeated START และแอดเดรสที่ระบุการอ่านข้อมูลจากนั้น slave จึงจะเริ่มส่งข้อมูลกลับมา ภาพที่ 2.13, 2.14 แสดงโปรโตคอล read byte แบบใช้ PEC และ read word แบบใช้ PEC ตามลำดับ



ภาพที่ 2.14 โปรโตคอล read byte แบบใช้ PEC

ที่มา : System Mangement Bus (SMBus) Specification version 3.0, 2014, page 41

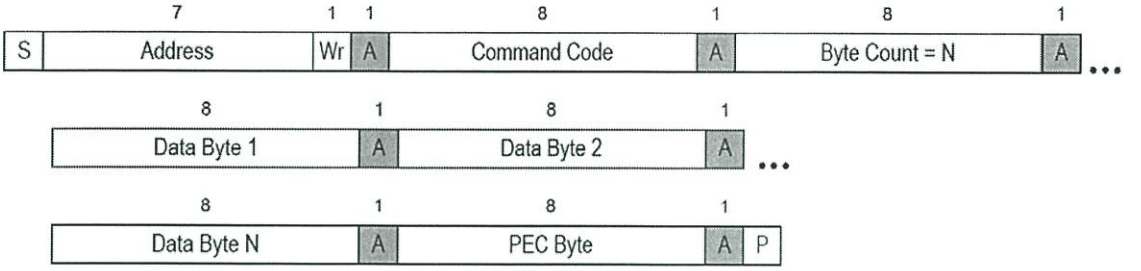


ภาพที่ 2.15 โปรโตคอล read word แบบใช้ PEC

ที่มา : System Mangement Bus (SMBus) Specification version 3.0, 2014, page 42

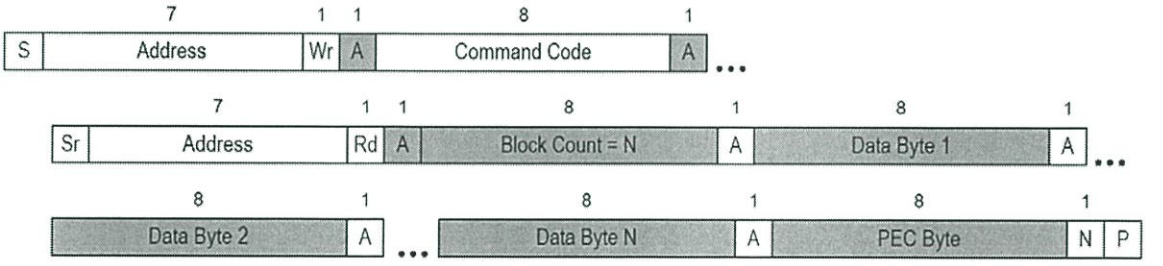
Block Write/Read

ทั้ง Block Write และ Read จะเริ่มด้วยการเขียนรหัสคำสั่งด้วยการส่ง slave แอดเดรสโดยบิทที่ 0มีค่า '0' ตามด้วยรหัสคำสั่ง สำหรับ Block Write จะตามด้วยไบต์เค้าท์ระบุจำนวนข้อมูลที่จะถูกส่ง (N) และตามด้วยข้อมูล N ไบต์ สำหรับ Block Read ต่างจาก Block Write คือหลังจากรหัสคำสั่งจะเกิดสัญญาณ repeated start ตามด้วย slave แอดเดรสโดยบิทที่ 0 มีค่า '1' เป็นการระบุการเปลี่ยนทิศทางของข้อมูล Block Write/Read รับส่งข้อมูลได้สูงสุด 255 ไบต์ ภาพที่ 2.15 แสดงโปรโตคอล Block Write แบบใช้ PEC ภาพที่ 2.16 แสดงโปรโตคอล Block Read แบบใช้ PEC



ภาพที่ 2.16 โปรโตคอล Block Write แบบใช้ PEC

ที่มา : System Mangement Bus (SMBus) Specification version 3.0, 2014, page 43



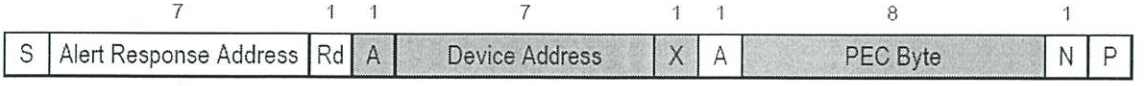
ภาพที่ 2.17 โปรโตคอล Block Read แบบใช้ PEC

ที่มา : System Mangement Bus (SMBus) Specification version 3.0, 2014, page 43

2.4.2 SMBAlert

สายสัญญาณ SMBAlert ใช้เพื่อให้อุปกรณ์บนบัสสามารถแจ้งเตือน system host controller ในกรณีที่อุปกรณ์เกิดการทำงานผิดปกติ สายสัญญาณนี้จะต่อเข้ากับไฟเลี้ยงผ่านตัวต้านทานพูลอัพเหมือนกับสัญญาณนาฬิกา(SCL)และสัญญาณข้อมูล(SDA) อุปกรณ์บนบัสสามารถทำให้สายสัญญาณเปลี่ยนค่าเป็น 'low' ได้เมื่อต้องการสื่อสารกับ system host controller

เมื่อ host ได้รับสัญญาณจาก SMBAlert# จะประมวลผลและเข้าถึงอุปกรณ์ที่ทำให้เกิดสัญญาณ SMBalert# ผ่าน Alert Response Address (ARA) ซึ่งมีค่าเท่ากับ 0001 100 โดยใช้โปรโตคอล receive byte เฉพาะอุปกรณ์ที่ส่งสัญญาณ SMBAlert# เท่านั้นที่จะ ACK แอดเดรสนี้แล้วส่งแอดเดรสของอุปกรณ์กลับมา โปรโตคอลแสดงดังภาพที่ 2.17

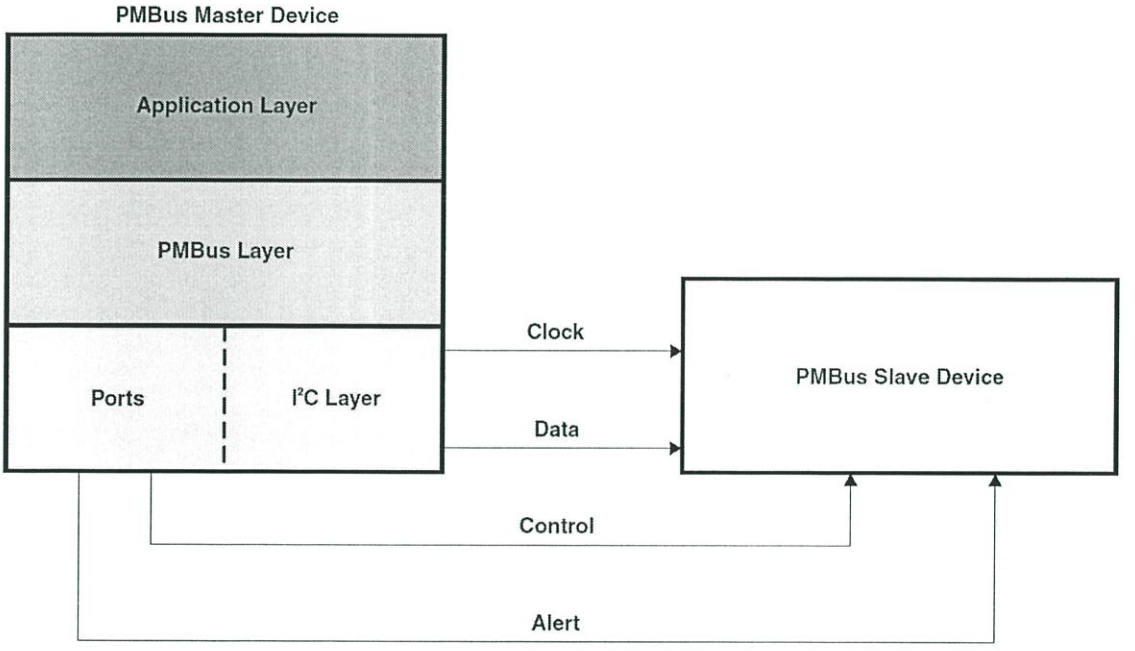


ภาพที่ 2.18 โปรโตคอล read byte ที่ดัดแปลงสำหรับใช้งานสัญญาณ SMBAlert#

2.5 Power Management Bus (PMBus) [5]

Power Mangement Bus เป็นโปรโตคอลมาตรฐานสากลด้านการจัดการพลังงานแบบดิจิตอลที่กำหนดขึ้นสำหรับการสื่อสารระหว่างอุปกรณ์แปลงพลังงานกับอุปกรณ์จัดการพลังงานดิจิตอลและหน่วยประมวลผลของระบบ PMBus ใช้โปรโตคอลการสื่อสาร SMBus ในระดับฮาร์ดแวร์และควบคุมด้วยคำสั่ง PMBus ผ่านซอฟแวร์อินเตอเฟซ การใช้คำสั่ง PMBus ทำให้สามารถกำหนดค่าพารามิเตอร์การทำงาน, ตรวจสอบสถานะการทำงานและดำเนินการแก้ไขข้อผิดพลาดในการทำงานของอุปกรณ์ได้

ในการออกแบบซอฟแวร์ที่ใช้ PMBus บนไมโครคอนโทรลเลอร์อาจจะแบ่งโครงสร้างได้เป็น 3 ส่วน แสดงในภาพที่ 2.18



ภาพที่ 2.19 แสดงโครงสร้างของ PMBus system host controller

2.5.1 Application layer

ชั้นแอพพลิเคชั่นอยู่ในระดับการใช้งานของผู้ใช้และยังสามารถมองได้ว่าเป็นชั้นที่ใช้ส่งผ่านข้อมูลไปยังชั้น PMBus หรือใช้ในการรับข้อมูลจากชั้น PMBus มาถอดรหัสข้อมูล เมื่อรับแอพพลิเคชั่นได้รับคำสั่งจากผู้ใช้งานจะส่งผ่านข้อมูลไปยังชั้นของ PMBus เพื่อประมวลผลต่อ

2.5.2 PMBus layer

ชั้นของ PMBus จะรับคำสั่งมาจากชั้นแอพพลิเคชั่นแล้วเปลี่ยนให้เป็นรหัสคำสั่ง (command code) ที่สอดคล้องกับสิ่งที่ผู้ใช้ป้อนเข้า จากนั้นส่งคำสั่งไปยังชั้นฮาร์ดแวร์อินเตอร์เฟซเพื่อบอกว่าต้องทำงานอย่างไร ชั้นของ PMBus ยังทำฟังก์ชันอื่นๆ เช่น การคำนวณ PEC ที่ต้องส่งต่อท้ายไปทุกครั้งที่มีการสื่อสาร, การประมวลผลสัญญาณ alert จากอุปกรณ์ PMBus เป็นต้น

โปรโตคอลของ PMBus

โปรโตคอลของ PMBus ใช้เหมือนกับโปรโตคอลของ SMBus (หัวข้อ 2.4.1) แตกต่างกันในส่วนของรหัสคำสั่ง PMBus ได้กำหนดรหัสคำสั่งสำหรับการทำงานแต่ละอย่างไว้แล้ว ซึ่งระบุไว้ในภาคผนวกของเอกสาร Power System Management Protocol Specification part II – Command Language หรือผู้ผลิตอุปกรณ์ที่สามารถใช้งาน PMBus อาจระบุไว้ในเอกสารของอุปกรณ์เช่นกัน

สัญญาณแจ้งเตือน SMBAlert

PMBus ได้นำสัญญาณ SMBAlert (หัวข้อ 2.4.2) จากโปรโตคอล SMBus มาใช้ เพื่อใช้ในการสื่อสารกับ PMBus system controller ในกรณีที่อุปกรณ์ PMBus มีการเปลี่ยนแปลงสถานะหรือทำงานผิดปกติ

สัญญาณควบคุม Control line

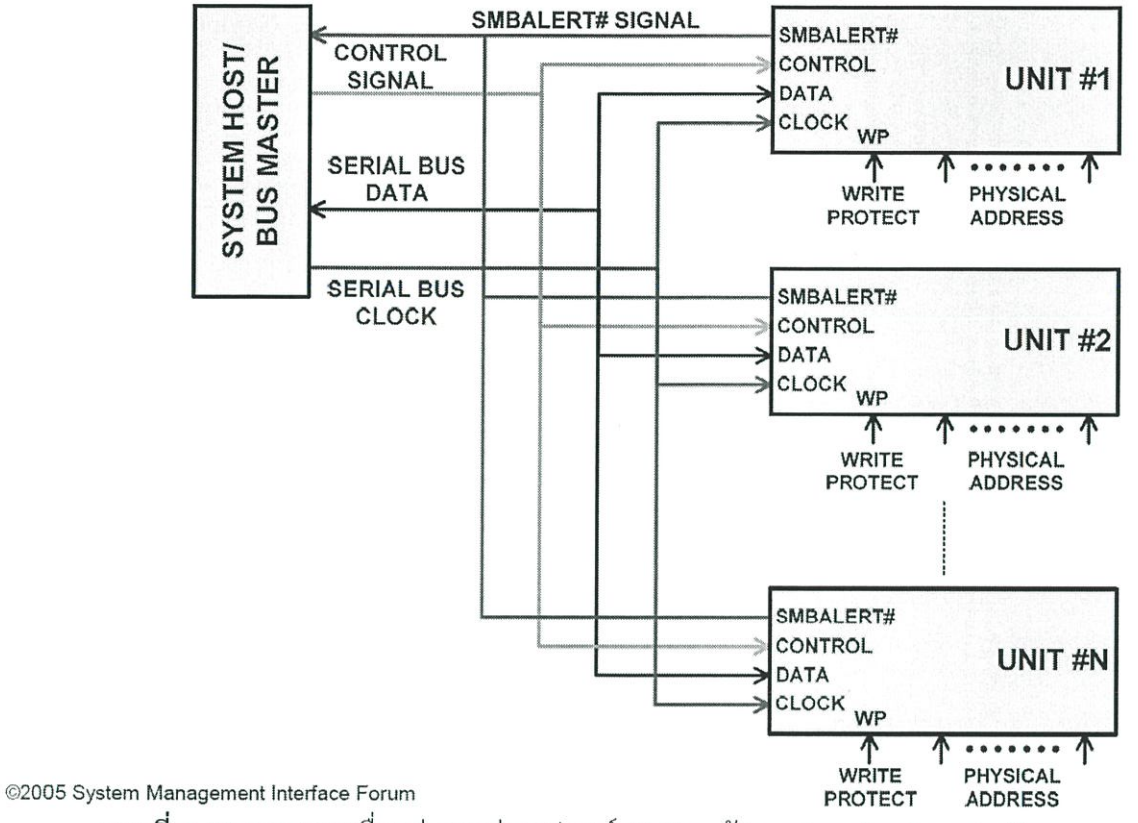
ใช้ควบคุมการเปิด/ปิดเอ้าพุทของอุปกรณ์ PMBus ทำได้โดยใช้สายควบคุมร่วมกับคำสั่งบนบัส สายสัญญาณควบคุมอาจถูกกำหนดค่าเป็น active low หรือ active high โดยขึ้นอยู่กับคำสั่ง ON_OFF_CONFIG จากนั้นคำสั่ง OPERATION จะใช้เพื่อเปิด/ปิดอุปกรณ์ PMBus โดยใช้ซอฟต์แวร์

PEC

PEC ถูกนำมาใช้สำหรับอุปกรณ์ PMBus เนื่องจากความถูกต้องของข้อมูลในระบบการจัดการพลังงานมีความสำคัญ PEC ถูกสร้างขึ้นโดยใช้อัลกอริธึม CRC-8 ซึ่งเป็นการดำเนินการ XOR บนอินพุตบิตสตรีมด้วยพหุนาม CRC PEC ถูกคำนวณจากไบต์ทั้งหมดในแพคเก็ต I^2C รวมถึงแอดเดรสของอุปกรณ์ การคำนวณ PEC ไม่รวมถึง Start/Stop, ACK/NACK และ repeated start bit

2.5.3 Physicacl layer

คือชั้นที่ใช้อินเตอร์เฟซกับอุปกรณ์ PMBus ด้วยสายสัญญาณต่างๆ เพื่อทำให้สามารถสื่อสารกับอุปกรณ์ PMBus ได้ PMBus ใช้สายสัญญาณในการสื่อสารสี่สาย โดยมีการนำสายสัญญาณข้อมูล (SDA) กับสายสัญญาณนาฬิกา (SCL) ของโปรโตคอลการสื่อสาร I^2C มาใช้, สายสัญญาณ SMBAlert# ของ SMBus และ PMBus เพิ่มสายสัญญาณ Control ที่ใช้สำหรับควบคุมการเปิดปิดเอ้าพุทของอุปกรณ์ PMBus ดังแสดงในภาพที่ 2.19



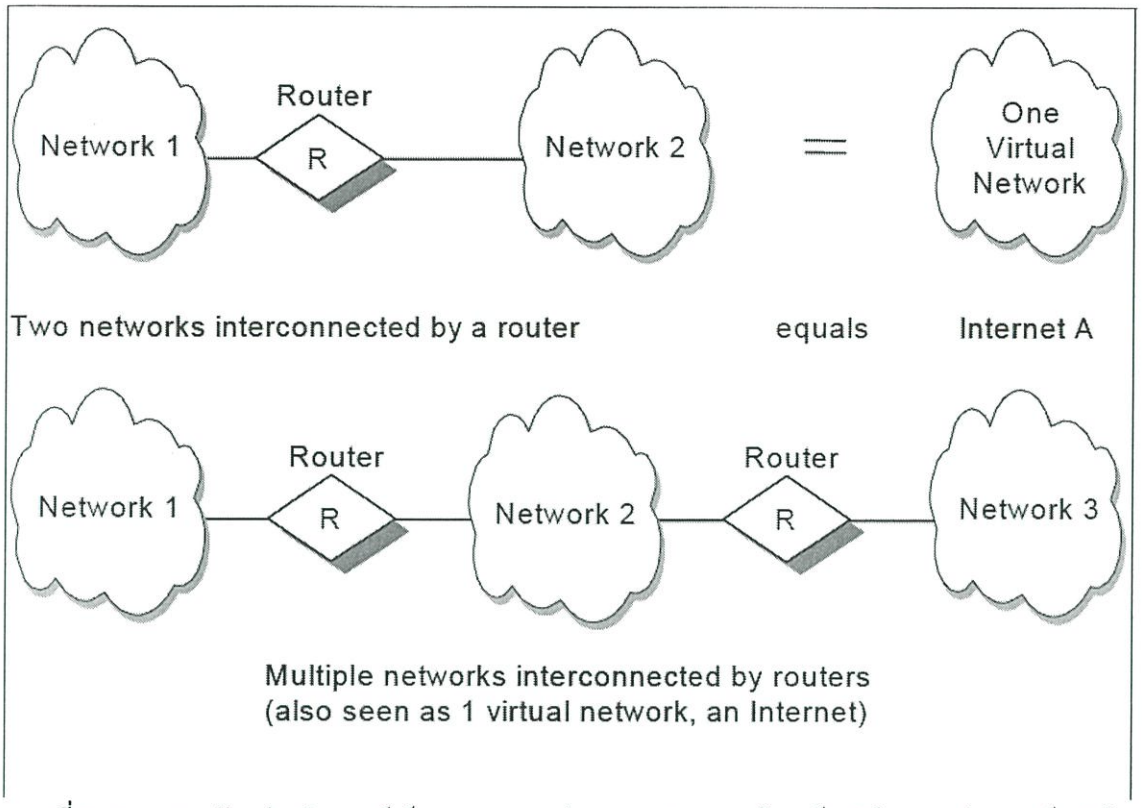
ภาพที่ 2.20 แสดงการเชื่อมต่อระหว่างอุปกรณ์ PMBus กับ system host controller

2.6 โปรโตคอล TCP/IP (TCP/IP protocol suite) [6]

2.6.1 Internetworking

จุดมุ่งหมายหลักในการออกแบบชุดโปรโตคอล TCP/IP คือการสร้างการเชื่อมต่อโครงข่ายซึ่งเรียกว่า อินเตอร์เน็ตเวิร์ก (internetwork) หรืออินเตอร์เน็ต (internet) ซึ่งให้บริการสื่อสารผ่านเครือข่ายที่แตกต่างกัน ประโยชน์ที่ชัดเจนของการทำงานแบบอินเตอร์เน็ตนี้คือการทำให้การติดต่อสื่อสารระหว่างโฮสต์บนเครือข่ายต่างๆที่อาจจะอยู่ห่างกันคนละพื้นที่ของโลกสามารถติดต่อสื่อสารกันได้ คำว่าอินเตอร์เน็ตเวิร์กหรืออินเตอร์เน็ตเป็นคำย่อของคำว่าเครือข่ายที่เชื่อมต่อกัน (interconnected network) อย่างไรก็ตามเมื่อเขียนคำว่าอินเตอร์เน็ตด้วยตัว "I" พิมพ์ใหญ่ (Internet) นั้นจะหมายถึงเครือข่ายที่เชื่อมต่อระหว่างกันทั่วโลก

อีกประการหนึ่งที่สำคัญของการเชื่อมต่อเน็ตเวิร์กด้วย TCP/IP คือการสร้างกลไกการสื่อสารที่มีมาตรฐานสำหรับเครือข่ายแต่ละประเภท แต่ละเครือข่ายมีอินเตอร์เฟซการสื่อสารขึ้นอยู่กับเทคโนโลยีของตัวเองในรูปแบบอินเตอร์เฟซการเขียนโปรแกรมที่ให้ฟังก์ชันการสื่อสารขั้นพื้นฐาน (primitives) TCP/IP ทำให้การสื่อสารที่ทำงานระหว่างอินเทอร์เฟซการเขียนโปรแกรมของเครือข่ายกับแอพพลิเคชันของผู้ใช้ TCP/IP ช่วยให้แอพพลิเคชั่นเหล่านี้ใช้งานร่วมกันได้โดยไม่ขึ้นกับโครงสร้างของเครือข่าย ดังนั้นสถาปัตยกรรมของเครือข่ายจึงถูกซ่อนจากผู้ใช้และจากผู้พัฒนาแอพพลิเคชั่น แอพพลิเคชั่นต้องใช้โค๊ดเฉพาะที่เป็นมาตรฐานเพื่อให้สามารถทำงานได้ทุกเครือข่ายและทุกระบบปฏิบัติการ



ภาพที่ 2.21 แสดงตัวอย่างอินเตอร์เน็ตสองแบบ แต่ละแบบประกอบด้วยเน็ตเวิร์กมากกว่าสองเน็ตเวิร์ก

ที่มา : TCP/IP Tutorial and Technical Overview, 2006, page 5

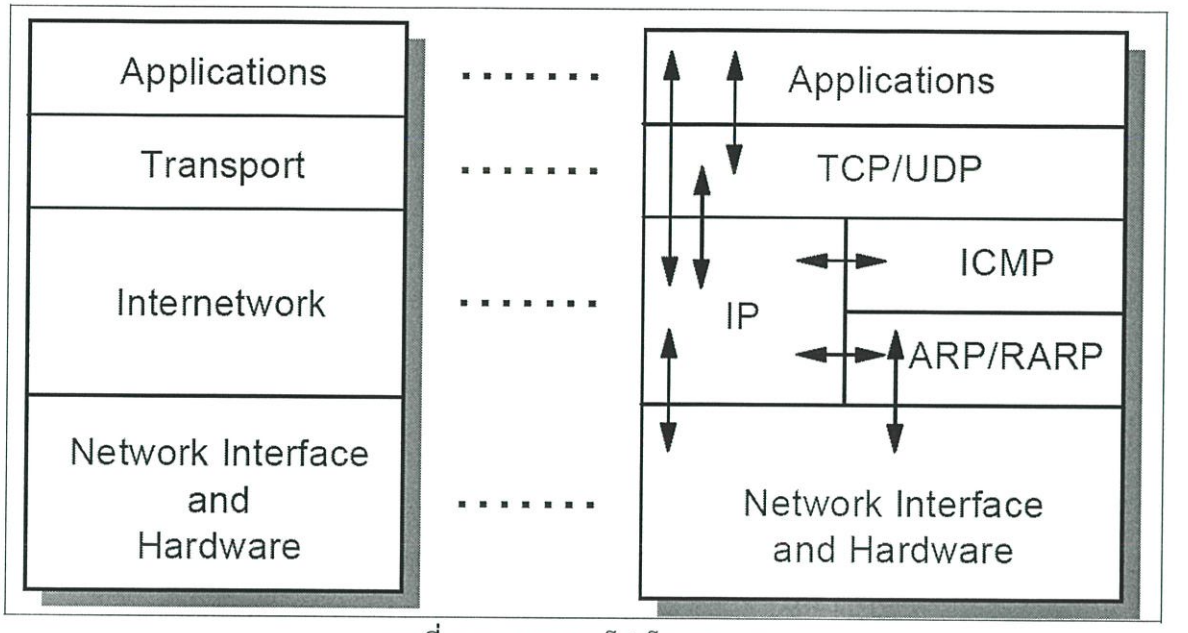
จากภาพที่ 2.19 เพื่อให้สองเครือข่ายเชื่อมต่อกันได้เราจำเป็นต้องมีคอมพิวเตอร์ที่เชื่อมต่อทั้งสองเครือข่ายและสามารถส่งต่อแพ็คเก็ตข้อมูลจากเครือข่ายหนึ่งไปยังอีกเครือข่ายหนึ่งได้ เครื่องนี้เรียกว่าเราเตอร์ (router) เพื่อให้สามารถระบุโฮสต์ภายในอินเตอร์เน็ต แต่ละโฮสต์จะกำหนดแอดเดรสซึ่งเรียกว่า IP address ประกอบด้วยสองส่วนคือ IP address = <network number><host number>

ส่วน network number ใช้ระบุเครือข่ายภายในอินเตอร์เน็ต ซึ่งมีค่าเฉพาะแตกต่างกับเครือข่ายอื่น ส่วน host number ใช้ระบุโฮสต์ที่อยู่ในเน็ตเวิร์กนั้นๆ

2.6.2 TCP/IP protocol layer

เช่นเดียวกับซอฟต์แวร์เครือข่ายส่วนใหญ่ TCP/IP เป็นโมเดลแบบเลเยอร์หรือโปรโตคอลสแตค (protocol stack) ซึ่งหมายถึงสแตคของเลเยอร์ในชุดโปรโตคอล โดยการแบ่งเป็นเลเยอร์ทำให้โปรโตคอลสแตคสามารถแบ่งงาน, เขียนและทดสอบโค๊ดได้ง่ายและยังสามารถพัฒนาเลเยอร์เพิ่มเติมได้ เลเยอร์สามารถสื่อสารได้ทั้งชั้นที่อยู่ด้านบนและชั้นที่อยู่ด้านล่าง โดยที่เลเยอร์ด้านล่างจะทำงานให้กับเลเยอร์ด้านบนและใช้ประโยชน์จากการทำงานของเลเยอร์ด้านล่างอีก ตัวอย่างเช่น IP layer ให้ความสามารถในการถ่ายโอนข้อมูลจากโฮสต์หนึ่งไปยังอีกที่หนึ่งโดยไม่รับประกันว่าข้อมูลจะไปถึงปลายทางหรือเกิดการทับ

ซ้อน โปรโตคอลการส่ง เช่น TCP ใช้ประโยชน์จากเลเยอร์นี้เพื่อทำให้การส่งข้อมูลมีประสิทธิภาพส่งถึงปลายทางและเรียงตามลำดับอย่างถูกต้อง ภาพที่ 2.20 แสดง TCP/IP โปรโตคอลสแตค



ภาพที่ 2.22 TCP/IP โปรโตคอลสแตค

ที่มา : TCP/IP Tutorial and Technical Overview, 2006, page 7

2.6.2.1 Application layer

แอพพลิเคชั่นเป็นโปรเซสของผู้ใช้ที่ทำงานร่วมกับโปรเซสอื่นโดยปกติจะอยู่ในโฮสต์อื่น ตัวอย่างแอพพลิเคชั่น ได้แก่ Telnet และ File Transfer Protocol (FTP) อินเตอร์เฟซระหว่างแอพพลิเคชั่นและเลเยอร์การส่งข้อมูลจะถูกกำหนดโดยหมายเลขพอร์ต (port number)และซ็อกเก็ต (socket) แนวคิดเกี่ยวกับพอร์ตและซ็อกเก็ตใช้เป็นแนวทางในการระบุการเชื่อมต่อและโปรแกรมและโฮสต์ที่ใช้งานร่วมกัน

พอร์ต (ports)

โปรเซสที่ต้องการสื่อสารกับอีกโปรเซสจะระบุตัวเองกับชุดโปรโตคอล TCP/IP ด้วยพอร์ตหนึ่งพอร์ตหรือมากกว่า พอร์ตคือตัวเลข 16 บิต มีพอร์ตสองประเภท:

- เลขพอร์ตมาตรฐาน (well-known port number) เป็นพอร์ตสำหรับเซิฟเวอร์มาตรฐาน เช่น Telnet ใช้พอร์ต 23 เลขพอร์ตมาตรฐานมีค่าตั้งแต่ 1 ถึง 1023 ถูกควบคุมและกำหนดขึ้นโดย Internet Assigned Number Authority (IANA) ในระบบส่วนใหญ่พอร์ตเหล่านี้จะถูกใช้งานโดยโปรเซสของระบบหรือโปรแกรมที่ใช้สิทธิการใช้งานพิเศษ เลขพอร์ตมาตรฐานทำให้ไคลเอนท์ค้นหาเซิฟเวอร์โดยไม่ต้องกำหนดค่าข้อมูล
- เลขพอร์ตชั่วคราว (ephemeral port number) ไคลเอนท์บางตัวไม่จำเป็นต้องใช้เลขพอร์ตมาตรฐานเพราะเลขพอร์ตที่ใช้รวมอยู่ใน UDP/TCP datagrams ที่ส่งไปยังเซิฟเวอร์ ไคลเอนท์แต่ละตัวจจัดสรรหมายเลขพอร์ตตราบเท่าที่ยังต้องการโดยโฮสต์ที่ใช้อยู่ เลขพอร์ตชั่วคราวมีค่าตั้งแต่ 1023 โดยปกติอยู่

ในช่วง 1024 ถึง 65535 พอร์ต เลขพอร์ตชั่วคราวไม่ได้ควบคุมโดย IANA และสามารถใช้งานได้โดยโปรแกรมที่พัฒนาโดยผู้ใช้ทั่วไปในระบบส่วนใหญ่

ซ็อกเก็ต (socket)

ซ็อกเก็ตอินเตอร์เฟซเป็นหนึ่งในส่วนติดต่อการเขียนโปรแกรมประยุกต์ (API) หลายๆตัวสำหรับโปรโตคอลการสื่อสาร ได้รับการออกแบบมาเพื่อเป็นส่วนติดต่อในการเขียนโปรแกรมการสื่อสารทั่วไป socket APIs ถูกนำเสนอครั้งแรกโดย Berkeley Software Distribution (BSD) แม้ว่าจะยังไม่ได้รับมาตรฐานแต่ Berkeley socket API เป็นที่นิยมในอุตสาหกรรมสำหรับการใช้งานซ็อกเก็ต TCP/IP บนเครือข่าย

ซ็อกเก็ตคือปลายทางของการสื่อสารที่กำหนดชื่อและแอดเดรสในเครือข่ายได้ ที่ปลายทางด้านหนึ่งของการเชื่อมต่อ TCP มีซ็อกเก็ตที่สามารถระบุได้โดยใช้ <TCP,IP address, port number> ถ้ามีการสื่อสารผ่าน TCP จะสามารถระบุปลายทางทั้งสองของการเชื่อมต่อได้ด้วย <TCP, local IP address, local port, remote IP address, remote port>

2.6.2.2 Transport layer

ชั้นการส่งข้อมูลจะให้การถ่ายโอนข้อมูลแบบ end-to-end โดยส่งข้อมูลจากแอพพลิเคชั่นไปยังปลายทาง โปรโตคอลชั้นการส่งข้อมูลที่ใช้กันมากที่สุดคือ Transmission Control Protocol (TCP) ซึ่งให้การสื่อสารอย่างเป็นกระบวนการและมีความถูกต้อง (conntection-oriented reliable data delivery), ลดการทับซ้อนของข้อมูล (duplicate data suppression), การควบคุมความหนาแน่น(congestion control) และการเคลื่อนที่ของข้อมูล (flow control)

2.6.2.3 Internetwork layer

ใช้ประเภทของระบบการสื่อสารที่เรียกว่า ระบบเครือข่ายแบบสลับช่องสื่อสารระดับแพคเกต (packet-switching network) ซึ่งเป็นการติดต่อแบบไม่ต่อเนื่อง (connectionless) หลักการทำงานคือการปล่อยให้ข้อมูลขนาดเล็กที่เรียกว่า แพคเกต (Packet) สามารถไหลจากโหนดผู้ส่งไปตามโหนดต่างๆ ในระบบจนถึงจุดหมายปลายทางได้โดยอิสระ หากว่ามีการส่งแพคเกตออกมาเป็นชุดโดยมีจุดหมายปลายทางเดียวกันในระหว่างการเดินทางในเครือข่าย แพคเกตแต่ละตัวในชุดนี้ก็จะเป็นอิสระแก่กันและกัน ดังนั้นแพคเกตที่ส่งไปถึงปลายทางอาจจะไม่เป็นไปตามลำดับก็ได้

Internet Protocol เป็นโปรโตคอลในระดับเน็ตเวิร์คเลเยอร์ ทำหน้าที่จัดการเกี่ยวกับแอดเดรสและข้อมูล และควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพคเกต ซึ่งกลไกในการหาเส้นทางของ IP จะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูล และมีระบบการแยกและประกอบดาต้าแกรม (datagram) เพื่อรองรับการส่งข้อมูลระดับ data link ที่มีขนาด MTU (Maximum Transmission Unit) ที่แตกต่างกัน ทำให้สามารถนำ IP ไปใช้บนโปรโตคอลอื่นได้หลากหลาย เช่น Ethernet ,Token Ring หรือ Apple Talk การเชื่อมต่อของ IP เพื่อทำการส่งข้อมูล

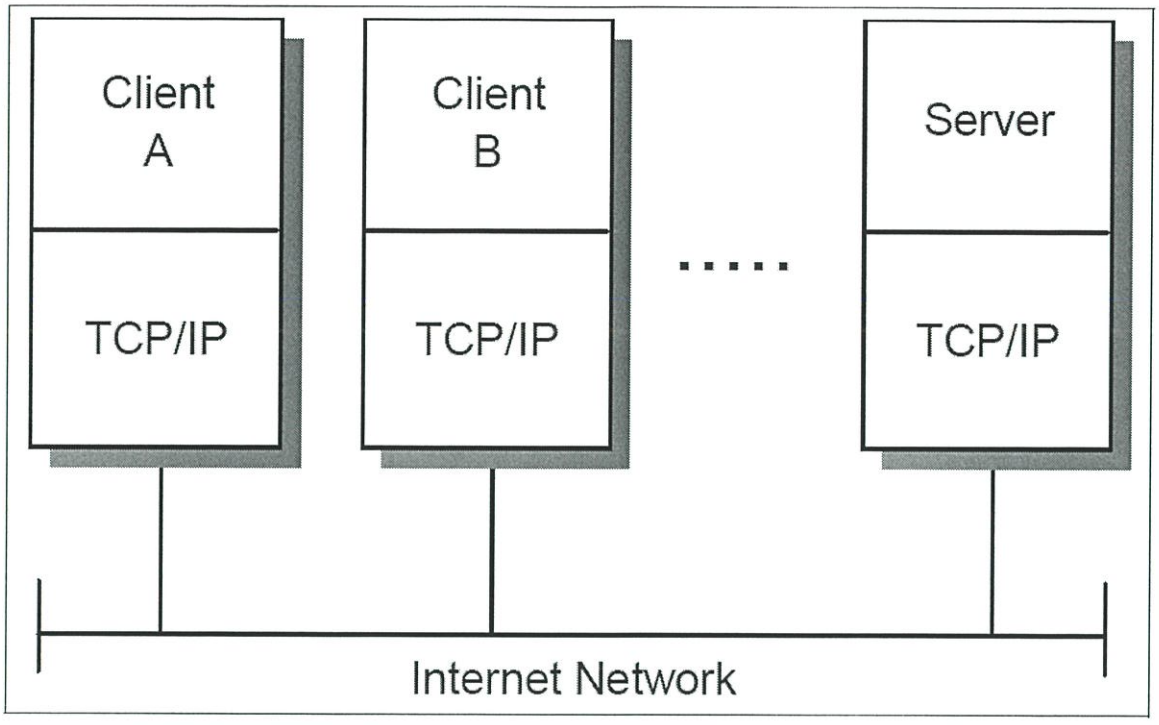
จะเป็นแบบ connectionless หรือเกิดเส้นทางการเชื่อมต่อในทุกๆครั้งของการส่งข้อมูล 1 ดาต้าแกรม โดยจะไม่ทราบถึงข้อมูลดาต้าแกรมที่ส่งก่อนหน้าหรือส่งตามมา แต่การส่งข้อมูลใน 1 ดาต้าแกรม อาจจะเกิดการส่งได้หลายครั้งในกรณีที่มีการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (fragmentation) และถูกนำไปรวมเป็นดาต้าแกรมเดิมเมื่อถึงปลายทาง

2.6.2.4 Network interface layer

ชั้นอินเตอร์เฟซเครือข่ายเป็นส่วนติดต่อกับฮาร์ดแวร์ระบบเครือข่าย อินเตอร์เฟซนี้อาจจะไม่สนับสนุนการส่งข้อมูลที่มีความถูกต้องก็ได้และข้อมูลอาจเป็น packet หรือ stream oriented ก็ได้ ในความเป็นจริง TCP/IP ไม่ได้ระบุโปรโตคอลใดๆในชั้นนี้ ทำให้สามารถใช้อินเตอร์เฟซเครือข่ายได้เกือบทุกแบบซึ่งแสดงถึงความยืดหยุ่นของชั้น IP ตัวอย่างคือ IEEE 802.2, X.25, ATM, FDDI และแม้แต่ SNA

2.6.3 Client/Server model

TCP เป็นโปรโตคอลแบบ peer-to-peer, connection-oriented ในส่วนแอพพลิเคชั่นมักใช้โมเดลไคลเอนท์/เซิฟเวอร์สำหรับการติดต่อสื่อสารระหว่างแอพพลิเคชั่นดังที่แสดงในภาพที่ 2.21
เซิฟเวอร์เป็นแอพพลิเคชั่นที่ให้บริการแก่ผู้ใช้อินเตอร์เน็ต ไคลเอนท์เป็นผู้ร้องขอใช้บริการ แอพพลิเคชั่นประกอบด้วยส่วนของเซิฟเวอร์และส่วนของไคลเอนท์ซึ่งอาจจะทำงานบนระบบเดียวกันหรือทำงานอยู่คนละระบบก็ได้ ผู้ใช้มักเรียกใช้ส่วนของไคลเอนท์แอพพลิเคชั่นซึ่งสร้างคำขอใช้บริการและส่งไปยังเซิฟเวอร์แอพพลิเคชั่นโดยใช้ TCP/IP ในการส่งข้อมูล
เมื่อเซิฟเวอร์ได้รับคำร้องเข้ามา เซิฟเวอร์จะทำการประมวลผลคำขอแล้วส่งผลลัพธ์กลับไปที่ผู้ร้องขอ(ไคลเอนท์)



ภาพที่ 2.23 แบบจำลองไคลเอนท์/เซิฟเวอร์

ที่มา : TCP/IP Tutorial and Technical Overview, 2006, page 10

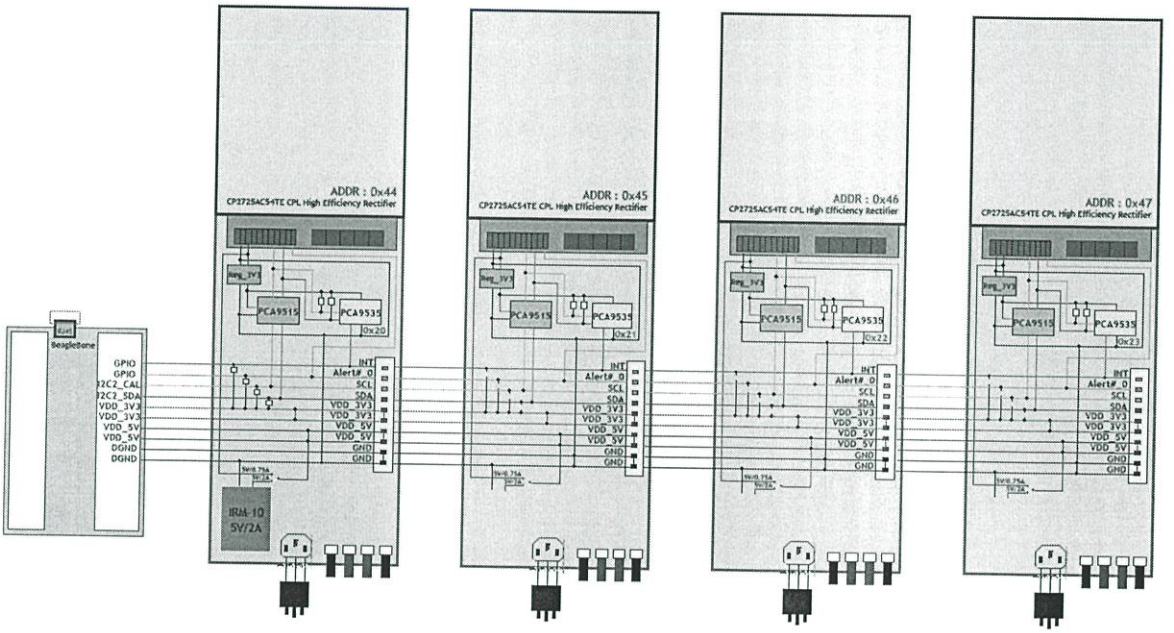
เซิฟเวอร์ส่วนใหญ่รอรับคำขอที่พอร์ตมาตรฐานเพื่อให้ไคลเอนท์ทราบว่าพอร์ตใด (และในทางกลับกันแอพพลิเคชั่นไหน) ที่จะต้องส่งคำขอไปหา แต่ไคลเอนท์อาจใช้พอร์ตชั่วคราวสำหรับการสื่อสารก็ได้ ไคลเอนต์ที่ต้องการติดต่อสื่อสารกับเซิฟเวอร์ที่ไม่ได้ใช้พอร์ตมาตรฐานต้องมีกลไกอื่นในการหาพอร์ตที่จะต้องส่งคำขอไปหาเซิฟเวอร์

บทที่ 3

วิธีดำเนินการวิจัย

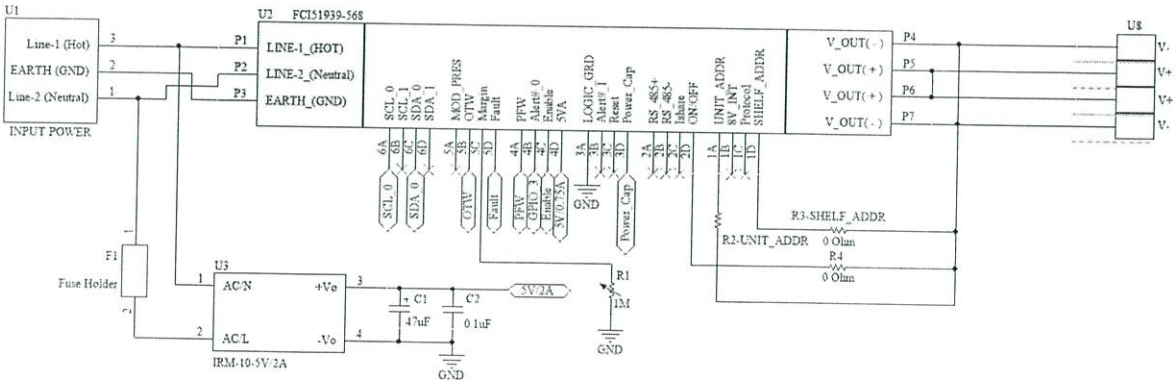
3.1 การออกแบบส่วนต่างๆ ด้านฮาร์ดแวร์

3.1.1 เค้าโครงระบบที่ออกแบบ

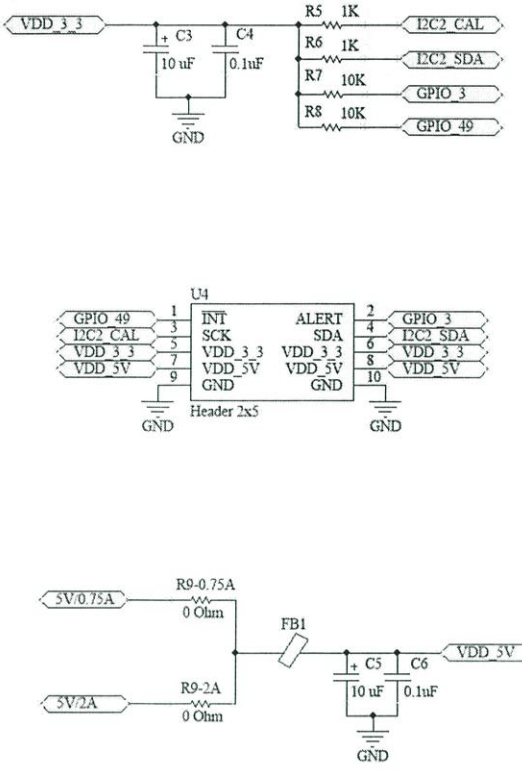


ภาพที่ 3.1 เค้าโครงระบบที่ออกแบบในโปรแกรม Altium Designer

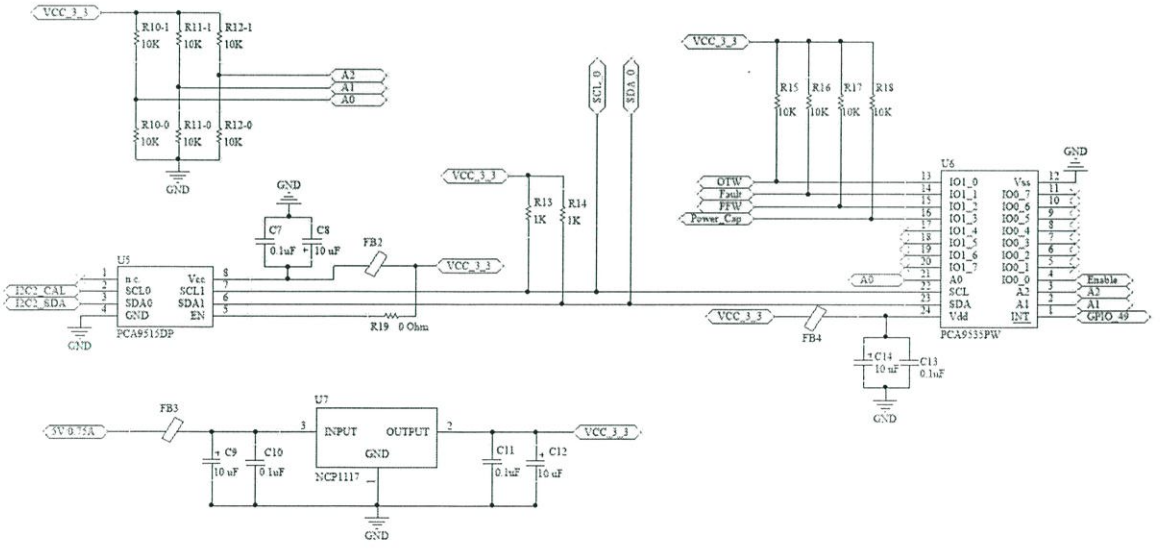
3.1.2 วงจรที่ออกแบบ



ภาพที่ 3.2 วงจรที่ออกแบบในโปรแกรม Altium Designer (1/3)

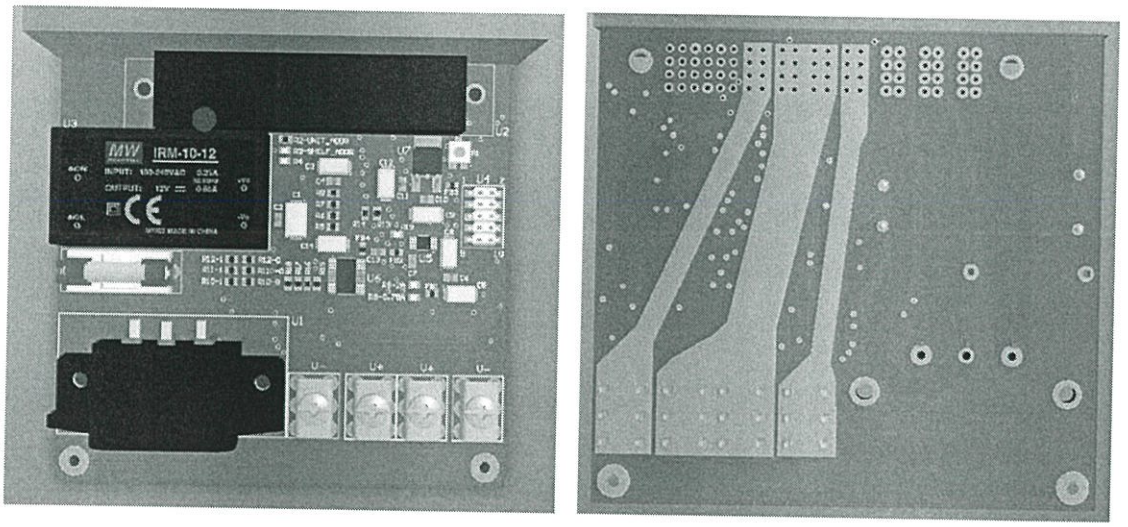


ภาพที่ 3.3 วงจรที่ออกแบบในโปรแกรม Altium Designer (2/3)

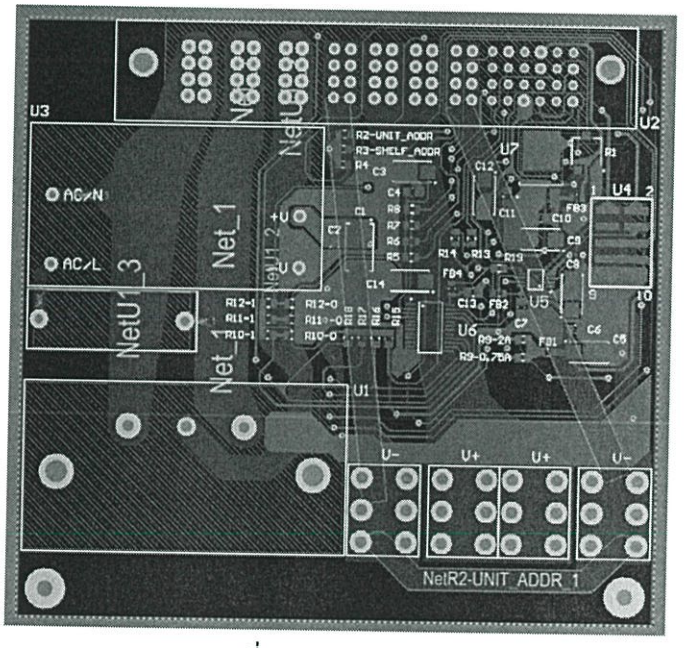


ภาพที่ 3.4 วงจรที่ออกแบบในโปรแกรม Altium Designer (3/3)

3.1.3 PCB ที่ออกแบบ



ภาพที่ 3.5 มุมมอง 3 มิติด้านบนและล่างของ PCB



ภาพที่ 3.6 ลายวงจร PCB

3.1.4 หลักการออกแบบวงจรและเลือกอุปกรณ์

3.1.4.1 U1 - AC Power Entry Modules (Input Power)

ไฟฟ้ากระแสสลับไหลเข้าสู่ CP2725AC54TE ตามที่ระบุใน Datasheet เท่ากับ 13.1 A_{AC}/240 V_{AC}

ไฟฟ้ากระแสสลับไหลเข้าสู่ IRM-10 ตามที่ระบุใน Datasheet เท่ากับ 0.15 A_{AC}/230 V_{AC}

ดังนั้น กระแสไฟฟ้ารวมเท่ากับ 13.25 A_{AC}

ดังนั้น เลือกใช้ AC Power Entry Modules ที่มี Current Rating 15A และ Voltage Rating 250V

3.1.4.2 U2 - Output Connector CP2725AC54TE

ตารางที่ 3.1 พินสัญญาณที่ใช้งาน

ตำแหน่ง	ชื่อ	ชนิด	ฟังก์ชัน	ประเภทของสัญญาณ
1A	Unit_addr	นำเข้า	Rectifier address	สัญญาณควบคุม
1D	Shelf_addr	นำเข้า	Shelf_address	สัญญาณควบคุม
2D	ON/OFF	นำเข้า	ON/OFF	สัญญาณควบคุม
3A	LOGIC_GRD	สัญญาณเปรียบเทียบ	Signal Reference	สัญญาณเปรียบเทียบ
3D	POWER_CAP	ส่งออก	Power Capacity	สัญญาณแสดงสถานะ
4A	PFW	ส่งออก	Power Fail Warning	สัญญาณแสดงสถานะ
4B	ALERT#_0	ส่งออก	SMBALERT# Line0	สัญญาณแสดงสถานะ
4C	Enable	นำเข้า	Output Enable	สัญญาณควบคุม
4D	5VA	ส่งออก	Standby Power	กำลังงานส่งออก
5B	OTW	ส่งออก	Over-Temperature Warning	สัญญาณแสดงสถานะ
5C	Margin	นำเข้า	Margining	สัญญาณควบคุม
5D	Fault	ส่งออก	Rectifier Fault	สัญญาณแสดงสถานะ
6A	SCL_0	นำเข้า	I^2C Line 0	สื่อสารผ่าน PMBus™ mode
6C	SDA_0	นำเข้า/ส่งออก	I^2C Line 0	สื่อสารผ่าน PMBus™ mode

ตารางที่ 3.2 พินกำลังที่ใช้งาน

ตำแหน่ง	ชื่อ	ชนิด
P1	LINE-1 (HOT)	กำลังงานนำเข้า
P2	LINE-2 (Neutral)	กำลังงานนำเข้า
P3	EARTH (GND)	กำลังงานนำเข้า
P4	V_OUT(-)	กำลังงานส่งออก
P5	V_OUT(+)	กำลังงานส่งออก
P6	V_OUT(+)	กำลังงานส่งออก
P7	V_OUT(-)	กำลังงานส่งออก

3.1.4.3 U3 - IRM-10-5 (10W Single Output Encapsulates Type)

IRM-10-5 เป็นตัวเลือกในการจ่ายกำลังงานให้กับบอร์ด BeagleBone Green ในกรณีที่กำลังงานจากพิน 5VA ของ CP2725AC54TE ไม่เพียงพอต่อการใช้งาน

IRM-10-5 มีคุณสมบัติกำลังด้านส่งออก ดังนี้

DC Voltage : 5 V

Rated Current : 2 A

Rated Power : 10 W

โดยออกแบบให้จ่ายกระแสไฟฟ้าเข้า Cape Expansion Headers ทางด้าน P9 โดยจ่ายเข้าที่พิน VDD_5V (ทั้งพินที่ 5 และ 6 เนื่องจากแต่ละพินสามารถรับกระแสไฟฟ้าได้สูงสุดที่ 1 A)

3.1.4.4 U4 – PCB Headers 2x5 pin

ใช้ในการเชื่อมต่อสัญญาณต่างๆ ที่ระบุตามตารางที่ 3.1 จากทั้ง 4 บอร์ดและสัญญาณจากบอร์ด BeagleBone Green เข้าด้วยกัน แบ่งออกเป็น 10 พิน ดังนี้

ตารางที่ 3.3 รายละเอียดสัญญาณบน Header 2x5 pin

พินที่	ชื่อ	รายละเอียด
1	INT	เป็นการรวมสัญญาณจากพิน interupt ของ PCA9535 ทั้ง 4 ตัว เพื่อจ่ายเข้าที่พิน GPIO_49 ของบอร์ด BeagleBone Green
2	ALERT	เป็นการรวมสัญญาณจากพิน ALERT#_0 ของ CP2725AC54TE ทั้ง 4 ตัว เพื่อจ่ายเข้าที่พิน GPIO_3 ของบอร์ด BeagleBone Green
3	SCK	เชื่อมต่อสัญญาณ Clock จากบอร์ด BeagleBone Green เข้ากับพิน SCL_0 ของ CP2725AC54TE ทั้ง 4 ตัว และเข้ากับพิน SCL ของ PCA9535 ทั้ง 4 ตัว
4	SDA	เชื่อมต่อสัญญาณ Data จากบอร์ด BeagleBone Green เข้ากับพิน SDA_0 ของ CP2725AC54TE ทั้ง 4 ตัว และเข้ากับพิน SDA ของ PCA9535 ทั้ง 4 ตัว
5, 6	VDD_3_3	นำเข้าแหล่งจ่ายแรงดัน 3.3 โวลต์จากพิน VDD_3_3 ของบอร์ด BeagleBone Green เพื่อใช้ Pull up ผ่านตัวต้านทานขนาด 10K โอห์ม ให้กับสัญญาณในพินที่ 1 ถึง 4
7, 8	VDD_5V	นำแหล่งจ่ายแรงดัน 5 โวลต์ จาก CP2725AC54TE ตัวใดตัวหนึ่งหรือจาก IRM-10-5 เพื่อจ่ายพลังงานให้กับพิน VDD_5V ของบอร์ด BeagleBone Green
9, 10	GND	เชื่อมต่อกราวด์ของทั้ง 4 บอร์ด เข้าสู่กราวด์ของบอร์ด BeagleBone Green

3.1.4.5 U5 - PCA9515DP (I^2C-bus repeater)

ใช้ PCA9515 เป็น Buffer เชื่อมต่อระหว่างสัญญาณด้านอินพุต คือสัญญาณ Clock และ Data จาก BeagleBone กับสัญญาณด้านเอาท์พุต คือสัญญาณ Clock และ Data จาก CP2725AC54TE และ PCA9535 โดยสามารถ Enable ในโหมด Active High กล่าวคือ ถ้าขา Enable เป็น 1 จะทำให้ Output มีสถานะเป็นไปตามสถานะของ Input แต่ถ้าขา Enable เป็น 0 จะทำให้ Output มีสถานะเป็น High Impedance โดยออกแบบให้รับสัญญาณ Enable จากพอร์ท 5VA ของ CP2725AC54TE (ปรับระดับแรงดันจาก 5 โวลต์เป็น 3.3 โวลต์โดยใช้ Fixed Voltage Regulators เบอร์ NCP1117)

เมื่อไม่มีสัญญาณไฟฟ้าจาก CP2725AC54TE (ไม่ได้ถูกเลือกใช้งาน) ทำให้ด้าน Output ของ Buffer อยู่ในสถานะความต้านทานสูง (High Impedance) เสมือนว่าไม่ได้ต่อเข้ากับระบบ ซึ่งเรียกว่าสภาวะลอยตัว (Floating) เพื่อป้องกันไม่ให้ Master ตรวจพบ Slave Address ที่ไม่ได้ถูกเลือกใช้งาน

3.1.4.6 U6 – PCA9535PW

ตารางที่ 3.4 พินที่ใช้งาน

ตำแหน่ง	ชื่อ	ชนิด	รายละเอียด
1	INT	Output	เมื่อสถานะของพินที่กำหนดให้เป็น Input มีการเปลี่ยน จะทำให้สัญญาณที่พินนี้เปลี่ยนจาก High เป็น Low (Active Low)
2	A1	Input	รับระดับแรงดันเพื่อกำหนด Address bit-A1
3	A2	Input	รับระดับแรงดันเพื่อกำหนด Address bit-A2
4	IO0_0	Output	เชื่อมต่อกับพิน Enable ของ CP2725AC54TE โดยเมื่อกำหนดให้พินนี้เป็นสถานะ Low จะสามารถ Enable Output หรือเปิดใช้งาน PMBus™ Mode ของ CP2725AC54TE ได้
12	V_{SS}	Supply Ground	เชื่อมต่อกับกราวด์ของแหล่งจ่ายไฟ
13	IO1_0	Input	เชื่อมต่อกับพิน OTW ของ CP2725AC54TE เพื่ออ่านสถานะ
14	IO1_1	Input	เชื่อมต่อกับพิน Fault ของ CP2725AC54TE เพื่ออ่านสถานะ
15	IO1_2	Input	เชื่อมต่อกับพิน PFW ของ CP2725AC54TE เพื่ออ่านสถานะ
16	IO1_3	Input	เชื่อมต่อกับพิน Power_Cap ของ CP2725AC54TE เพื่ออ่านสถานะ
21	A0	Input	รับระดับแรงดันเพื่อกำหนด Address bit-A1

22	SCL	Input	รับสัญญาณ Clock จาก BeagleBone
23	SDA	Input/Output	รับ-ส่งข้อมูลผ่านการสื่อสารแบบ I^2C
24	V_{DD}	Supply Voltage	เชื่อมต่อกับแหล่งจ่ายไฟ

3.1.4.7 U7 – NCP1117

รับกำลังงานจากพิน 5VA (5โวลต์/0.75แอมป์) ของ CP2725AC54TE เพื่อปรับแรงดันขาออกเป็น 3.3 โวลต์

นำแรงดัน Output ที่ได้มาใช้งาน ดังนี้

- จ่ายเป็นไฟเลี้ยงให้กับ PCA9515DP และ PCA9535PW
- ควบคุมสัญญาณ Enable ของ PCA9515DP
- Pull Up ผ่านตัวต้านทาน 10K โอห์ม เพื่อกำหนด Address bit A0 A1 A2 ให้กับ PCA9535PW
- Pull Up ผ่านตัวต้านทาน 10K โอห์มให้กับสัญญาณ Clock และ Data ทางด้าน Output ของ PCA9515DP
- Pull up ผ่านตัวต้านทาน 10K โอห์มให้กับสัญญาณที่พิน OTW, Fault, PFW, และ Power_Cap

3.1.4.8 U8 – Screw Terminals (Output Power)

ระดับแรงขาออกจากสูงสุดจาก CP2725AC54TE คือ 58 V_{DC}

กระแสไฟฟ้าสูงสุดที่ออกจาก CP2725AC54TE คือ 52.4 A_{DC}

เลือกใช้ Screw Terminals เป็น Connector Output Power ของบริษัท Keystone Electronics มีคุณสมบัติอัตราการทนกระแสไฟฟ้าสูงสุด 30A

จึงใช้เป็น Connector ทางด้าน V_{OUT}(+) จำนวน 2 ตัว และทางด้าน V_{OUT}(-) 2 ตัว สามารถรองรับกระแสไฟฟ้าขาออกได้สูงสุดรวม 60 แอมป์

3.1.4.9 อุปกรณ์พาสซิฟ

R1 : ตัวต้านทานปรับค่าได้ 1 เมกกะโอห์ม

หลักการออกแบบ : ระดับแรงดันที่พิน **Margin** มีความสัมพันธ์กับระดับแรงดันไฟฟ้าขาออก (Output Power) ดังนี้

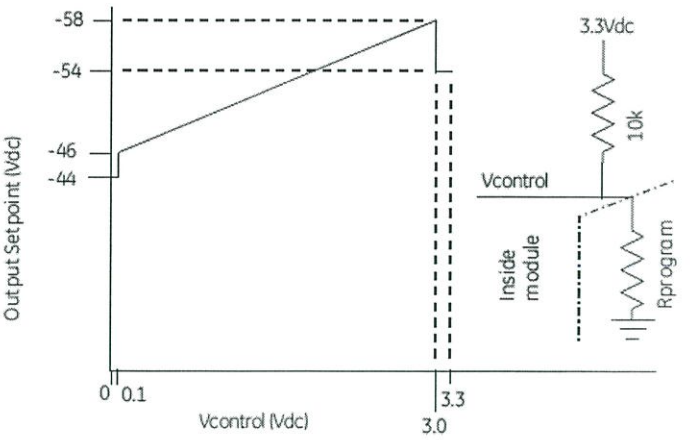
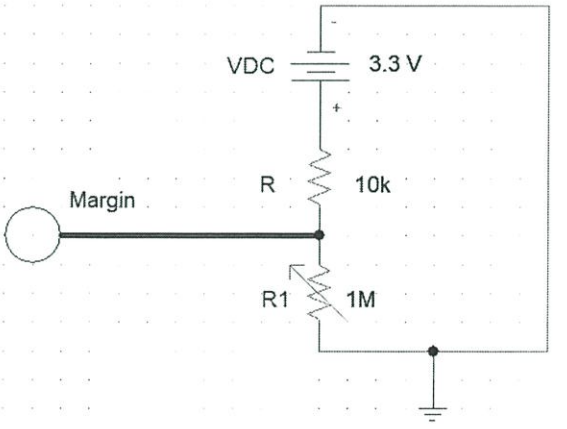


Figure 13. Diagram showing how output can be margined using Vcontrol adjustment.

ภาพที่ 3.7 ไดอะแกรมแสดงความสัมพันธ์ระหว่างแรงดันควบคุมและแรงดันเอาต์พุตหลัก

กรณีใช้งานในโหมดอนาลอก สามารถปรับแรงดันไฟฟ้าขาออกได้โดยการปรับค่าตัวต้านทาน R1 ที่เชื่อมต่ออยู่ระหว่างพิน Margin และพิน LOGIC_GRD ดังรูป



ภาพที่ 3.8 แสดงการต่อ R1 เพื่อปรับแรงดันควบคุม

ตามหลักการแบ่งแรงดัน (Voltage Dividers) สามารถคำนวณค่า R1 ได้ดังนี้

กรณีค่า R_1 = 1 เมกกะโอห์ม จะได้ $V_{Margin} = \frac{1M}{1M+10K} \times 3.3 \approx 3.267\ V.$

จะได้ $V_{DC} = 54\ V.$

กรณีนี้จะเกิดกระแสรั่วไหลเท่ากับ $I = \frac{V_{DC}}{R+R_1} = \frac{3.3}{10K+1M} = \frac{3.3}{1.01\ M} \approx 3.267\ \mu A$

กรณีค่า R1 = 0 โอห์ม จะได้ $V_{Margin} = \frac{0}{0+10K} \times 3.3 \approx 0\ V.$

จะได้ $V_{DC} = 44\ V.$

กรณีนี้จะเกิดกระแสรั่วไหลมากที่สุด เท่ากับ $I = \frac{V_{DC}}{R} = \frac{3.3}{10K} \approx 330\ \mu A$

ดังนั้น สามารถปรับค่าความต้านทาน R_1 ได้ตลอดช่วง 0 – 1 เมกกะโอห์ม เพื่อปรับระดับแรงดันที่พิน **Margin** และส่งผลไปยังระดับแรงดันไฟฟ้าขาออก (Output Power)

R2 : ตัวต้านทานกำหนด Unit Address

หลักการออกแบบ : เชื่อมต่อตัวต้านทาน R2 ค่าต่างๆ ระหว่างพิน **Unit addr** และพิน **Vout(-)** เพื่อกำหนด Address bit- A0 และ A1 ให้กับเพาเวอร์ซัพพลายทั้ง 4 หน่วย (Unit) ใน 1 ชั้น (Shelf)

R3 : ตัวต้านทาน 0 โอห์ม

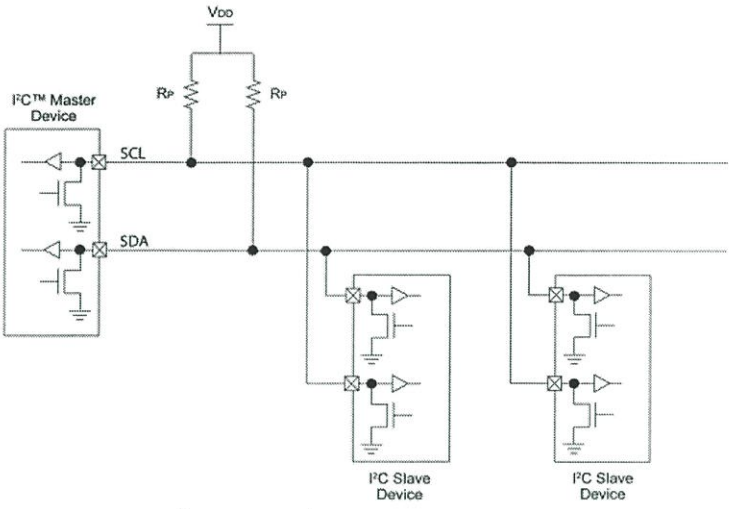
หลักการออกแบบ : ระดับแรงดันที่พิน **Shelf Addr** เทียบกับพิน **V_out(-)** จะเป็นตัวกำหนด Address bit- A2 ซึ่งใน 1 ระบบ I_2C สามารถกำหนดมีได้ 2 Shelf คือ Shelf 0 และ Shelf 1 ผู้ออกแบบจะให้ R3 = 0 โอห์มเป็นตัวเชื่อมต่อระหว่างพิน **Shelf Addr** และ **V_out(-)** ทำให้ระดับแรงดันระหว่างพิน **Shelf Addr** และพิน **V_out(-)** มีค่าประมาณ 0 โวลต์ จะได้ Address bit- A2 = 1

R4 : ตัวต้านทาน 0 โอห์ม

หลักการออกแบบ : เชื่อมต่อตัวต้านทาน R4 = 0 โอห์มระหว่างพิน **ON/OFF** และพิน **V_out(-)** เพื่อทำการ ON/OFF เพาเวอร์ซัพพลาย

R5, R6 : **ตัวต้านทาน 10K โอห์ม** (I^2C Bus Pull up resistor)

หลักการออกแบบ :



ภาพที่ 3.9 การเชื่อมต่อตัวต้านทาน Pull up

สามารถคำนวณหาค่าของตัวต้านทานได้ดังนี้

$$R_{p(min)} = \frac{(V_{CC} - V_{OL(max)})}{I_{OL}} = \frac{3.3 - 0.4}{3\ m} = 966.667 \text{ โอห์ม}$$

$$R_{p(max)} = \frac{t_r}{(0.8473 \times C_b)} = \frac{1\mu}{(0.8473 \times 400 \times 10^{-12})}$$

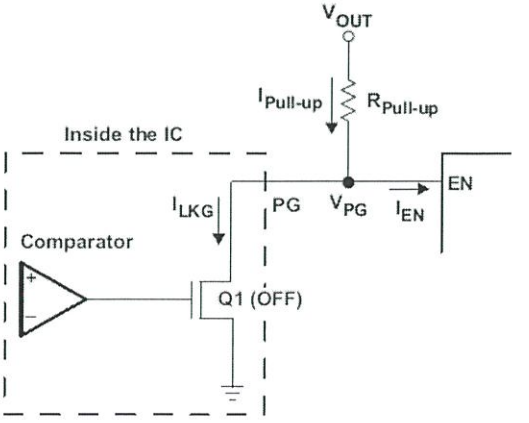
$$R_{p(max)} = 2950.548 \text{ โอห์ม}$$

สามารถใช้ค่าตัวต้านทานได้ตลอดช่วง 966.667 – 2950.548 โอห์ม

ผู้ออกแบบได้เลือกใช้ตัวต้านทานขนาด 1K โอห์ม

R7, R8 : **ตัวต้านทาน 10 กิโลโอห์ม** (Pull up resistor for open drain outputs)

หลักการออกแบบ : การคำนวณค่าตัวต้านทานที่ใช้ในการ Pull up

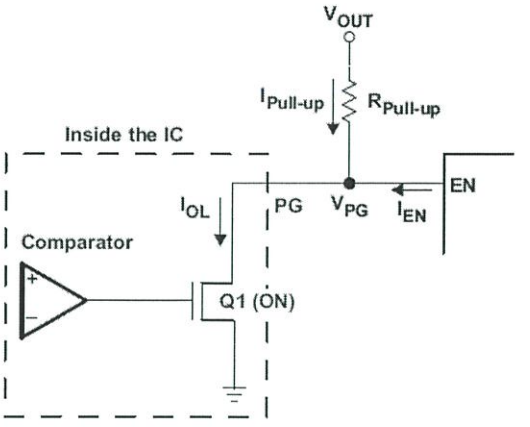


ภาพที่ 3.10 การเชื่อมต่อตัวต้านทานสำหรับเอาต์พุตแบบ Open Drain (MOSFET OFF)

จากรูป จะได้ $I_{Pull-up} = I_{EN} + I_{LKG} = 80\mu + 4(10\mu) = 120\mu A$

สามารถคำนวณหาค่าตัวต้านทานสูงสุด ได้จาก $R_{Pull\text{-}up,max} = \frac{V_{out} - V_{IH}}{I_{pull\text{-}up}} = \frac{3.3 - 2.145}{120\mu}$

$$R_{Pull\text{-}up,max} = 9625 \text{ Ohm.}$$



ภาพที่ 3.11 การเชื่อมต่อตัวต้านทานสำหรับเอาต์พุตแบบ Open Drain (MOSFET ON)

จากรูป จะได้ $I_{Pull-up} = I_{OL} - I_{EN} = 5m - 80\mu = 4.92 \text{ mA}$

สามารถคำนวณหาค่าตัวต้านทานต่ำสุด ได้จาก $R_{Pull\text{-}up,min} = \frac{V_{out}}{I_{pull\text{-}up}} = \frac{3.3}{4.92\ m}$

$$R_{Pull\text{-}up,min} = 670.732 \text{ โอห์ม}$$

สามารถเลือกใช้ค่าตัวต้านทานได้ตลอดช่วง 670.732 ถึง 9625 โอห์ม

ผู้ออกแบบได้เลือกใช้ตัวต้านทานขนาด 10K โอห์ม

R9 : ตัวต้านทาน 0 โอห์ม

หลักการออกแบบ :

เลือกใส่ R9-0.75A หรือ R9-2A เพียงตัวใดตัวหนึ่งบน PCB ที่มีการลงอุปกรณ์ IRM-10-5 เพื่อเลือกแหล่งจ่ายไฟให้กับบอร์ด BeagleBone

เลือกใส่ R9-0.75A เพื่อใช้พอร์ท 5VA (5 โวลต์/0.75แอมป์) ของเพาเวอร์ซัพพลายที่เชื่อมต่ออยู่กับ PCB นั้น เป็นแหล่งจ่ายไฟให้กับบอร์ด BeagleBone

เลือกใส่ R9-2A เพื่อใช้แหล่งจ่ายไฟขนาด (5 โวลต์/2 แอมป์) จาก IRM-10-5 เป็นแหล่งจ่ายไฟให้กับบอร์ด BeagleBone

ผู้ออกแบบได้เลือกใช้ตัวต้านทาน 0 โอห์ม ขนาด 0805 มีอัตราทนกระแสไฟฟ้าสูงสุด 2.5 แอมป์

R10, R11, R12 : ตัวต้านทาน 10 กิโลโอห์ม

หลักการออกแบบ :

เลือกใส่ R10-0 หรือ R10-1 บน PCB เพียงตัวใดตัวหนึ่งเพื่อกำหนด Address bit- A0 ให้กับ PCA9535PW

เลือกใส่ R11-0 หรือ R11-1 บน PCB เพียงตัวใดตัวหนึ่งเพื่อกำหนด Address bit- A1 ให้กับ PCA9535PW

เลือกใส่ R12-0 หรือ R12-1 บน PCB เพียงตัวใดตัวหนึ่งเพื่อกำหนด Address bit- A2 ให้กับ PCA9535PW

เลือกใส่ R10-0 เพื่อกำหนด Address bit-A0 ของ PCA9535PW ให้เท่ากับ 0

เลือกใส่ R10-1 เพื่อกำหนด Address bit-A0 ของ PCA9535PW ให้เท่ากับ 1

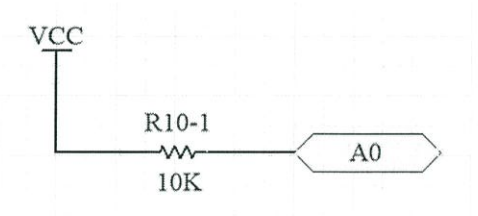
เลือกใส่ R11-0 เพื่อกำหนด Address bit-A1 ของ PCA9535PW ให้เท่ากับ 0

เลือกใส่ R11-1 เพื่อกำหนด Address bit-A1 ของ PCA9535PW ให้เท่ากับ 1

เลือกใส่ R12-0 เพื่อกำหนด Address bit-A2 ของ PCA9535PW ให้เท่ากับ 0

เลือกใส่ R12-1 เพื่อกำหนด Address bit-A2 ของ PCA9535PW ให้เท่ากับ 1

การเลือกขนาดตัวต้านทาน



ภาพที่ 3.12 การต่อตัวต้านทาน Pull up

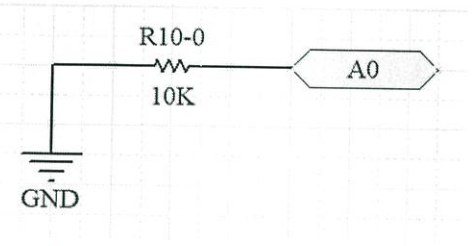
ค่าความต้านทานสูงสุด $R_{\text{Pull-up, max}} = \frac{V_{cc} - V_{IH(min)}}{I_{\text{input leakage}}} = \frac{3.3 - (0.7)(3.3)}{1\mu} = 990\text{K Ohms}$

กรณีที่พิน A0 เกิดการลัดวงจรภายใน จะกำหนดให้กระแสลัดวงจรนี้มีค่าไม่เกิน 1 mA จะได้

ค่าความต้านทานต่ำสุด $R_{\text{Pull-up, min}} = \frac{V_{cc}}{I_{\text{Short Circuit}}} = \frac{3.3 - (0.7)(3.3)}{1\text{m}} = 990 \text{ Ohms}$

สามารถใช้ค่าตัวต้านทาน Pull up ได้ตลอดช่วง 990 ถึง 990K โอห์ม

ผู้ออกแบบได้เลือกใช้ค่าตัวต้านทาน Pull up ขนาด 10K โอห์ม



ภาพที่ 3.13 การต่อตัวต้านทาน Pull down

ค่าความต้านทานสูงสุด $R_{\text{Pull-down, max}} = \frac{V_{IL(max)}}{|I_{\text{short Circuit}}|} = \frac{(0.3)(3.3)}{|-1\mu|} = 990\text{K Ohms.}$

กรณีที่พิน A0 เกิดการลัดวงจรภายในแล้วไปเชื่อมต่อกับ V_{cc} จะทำให้เกิดกระแสไฟฟ้าลัดวงจร กำหนดให้กระแสลัดวงจรนี้ให้มีค่าไม่เกิน 1 mA จะได้

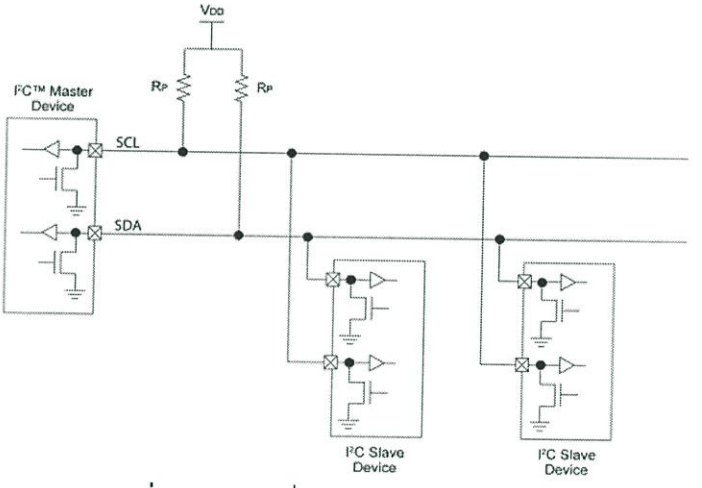
ค่าความต้านทานต่ำสุด $R_{\text{Pull-up, min}} = \frac{V_{cc}}{I_{\text{Short Circuit}}} = \frac{3.3}{1\text{m}} = 3.3\text{K Ohms}$

สามารถใช้ค่าตัวต้านทาน Pull down ได้ตลอดช่วง 3.3K ถึง 990K โอห์ม

ผู้ออกแบบได้เลือกใช้ค่าตัวต้านทาน Pull down ขนาด 10K โอห์ม

R13, R14 : **ตัวต้านทาน 1 กิโลโอห์ม** (I²C Bus Pull up resistor)

หลักการออกแบบ : การคำนวณค่าตัวต้านทาน



ภาพที่ 3.14 การเชื่อมต่อตัวต้านทาน Pull up

$$R_{p(min)} = \frac{(V_{CC} - V_{OL(max)})}{I_{OL}} = \frac{3.3 - 0.4}{3\text{ m}} = 966.667\text{ Ohms}$$

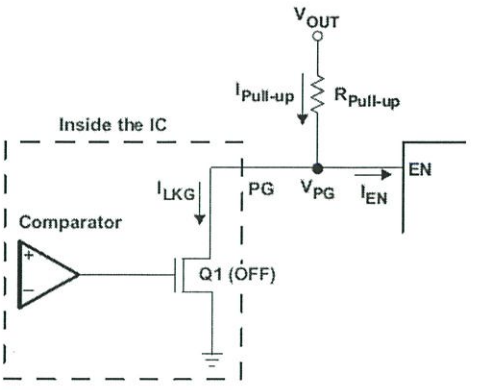
$$R_{p(max)} = \frac{t_r}{(0.8473 \times C_b)} = \frac{1\mu}{(0.8473 \times 400 \times 10^{-12})} = 2950.548\text{ Ohms}$$

สามารถใช้ค่าตัวต้านทานได้ตลอดช่วง 966.667 ถึง 2950.548 โอห์ม

ผู้ออกแบบได้เลือกใช้ตัวต้านทานขนาด 1K โอห์ม

R15, R16, R17, R18 : **ตัวต้านทานขนาด 10 กิโลโอห์ม**

หลักการออกแบบ : การคำนวณค่าตัวต้านทานที่ใช้ในการ Pull up



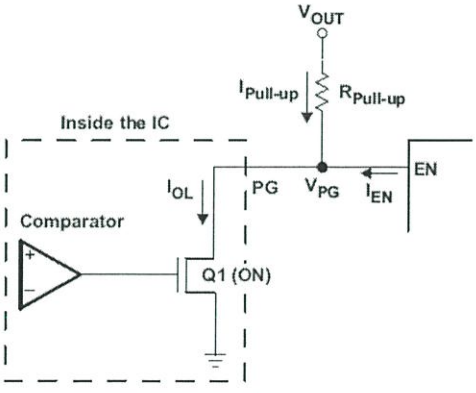
ภาพที่ 3.15 การเชื่อมต่อตัวต้านทานสำหรับเอาต์พุตแบบ Open Drain (MOSFET OFF)

จากรูป จะได้ $I_{Pull-up} = I_{LKG} + I_{EN} = 10\mu + 1\mu = 11\mu$

สามารถคำนวณหาค่าตัวต้านทานสูงสุด ได้จาก

$$R_{Pull\text{-}up,max} = \frac{V_{out} - V_{IH(min)}}{I_{pull\text{-}up}} = \frac{3.3 - (0.7 \times 3.3)}{11\mu}$$

$$R_{Pull\text{-}up,max} = 90 \text{ กิโลโอห์ม}$$



ภาพที่ 3.16 การเชื่อมต่อตัวต้านทานสำหรับเอาต์พุตแบบ Open Drain (MOSFET ON)

จากรูป จะได้ $I_{Pull-up} = I_{OL} - I_{EN} = 5m - 1\mu = 4.999$ มิลลิแอมป์

สามารถคำนวณหาค่าตัวต้านทานต่ำสุด ได้จาก $R_{Pull\text{-}up,min} = \frac{V_{OUT}}{I_{pull\text{-}up}} = \frac{3.3}{4.999m}$

$$R_{Pull\text{-}up,min} = 660.132 \text{ โอห์ม}$$

สามารถเลือกใช้ค่าตัวต้านทานได้ตลอดช่วง 660.132 ถึง 90 กิโลโอห์ม

ผู้ออกแบบได้เลือกใช้ตัวต้านทานขนาด 10 กิโลโอห์ม

R19 : ตัวต้านทานขนาด 0 โอห์ม

หลักการออกแบบ : พิน Enable ของ PCA9515DP มีคุณสมบัติ Active High โดยเมื่อพิน Enable ได้รับแรงดันไฟเลี้ยง ตัวไอซี PCA9515DP จะทำงาน และจะหยุดทำงานเมื่อพิน Enable ไม่ได้รับแรงดันไฟเลี้ยง ผู้ออกแบบจึงใช้ตัวต้านทานขนาด 0 โอห์ม เป็นตัวเชื่อมต่อระหว่างพิน Enable และแรงดันไฟเลี้ยง

C1 : ตัวเก็บประจุขนาด 47 uF

หน้าที่ : เพื่อเก็บหรือสำรองประจุไฟฟ้า (คำแนะนำจากดาต้าชีทของ IRM-10)

หลักการเลือกอุปกรณ์ : เลือกใช้ตัวเก็บประจุชนิดแทนทาลัม มีพิกัดแรงดัน 25 โวลต์

ตำแหน่งการวางอุปกรณ์ที่เหมาะสม : ใกล้แรงดันเอาท์พุตของ IRM-10 ให้มากที่สุดเท่าที่เป็นไปได้

C3, C5, C8, C9, C12, C14 : ตัวเก็บประจุขนาด 10 uF

หน้าที่ : เพื่อเก็บหรือสำรองประจุไฟฟ้า

หลักการเลือกอุปกรณ์ : เลือกใช้ตัวเก็บประจุชนิดแทนทาลัม มีขั้ว มีพิกัดแรงดัน 25 โวลต์

ตำแหน่งการวางอุปกรณ์ที่เหมาะสม : C5, C8, C9, C12, C14 ไม่ควรห่างจากขาไฟเลี้ยงของอุปกรณ์เกิน 2 นิ้ว

C2, C4, C6, C7, C10, C11, C13 : ตัวเก็บประจุขนาด 0.1 ไมโครฟารัด

หน้าที่ : ดูดซับสัญญาณรบกวนความถี่สูงลงกราวด์

หลักการเลือกอุปกรณ์ : เลือกใช้ตัวเก็บประจุชนิดเซรามิค ไม่มีขั้ว มีพิกัดแรงดัน 25 โวลต์

ตำแหน่งการวางอุปกรณ์ที่เหมาะสม : C6, C7, C10, C13 ด้านหนึ่งควรวางให้ติดกับขาไฟเลี้ยงของตัวอุปกรณ์ให้มากที่สุด อีกด้านหนึ่งควรเชื่อมต่อกับกราวด์เพลน โดยให้มีระยะทางสั้นที่สุด

FB1, FB2, FB3, FB4 : Ferrite Bead

หน้าที่ : กำจัดสัญญาณรบกวนในระบบ

ป้องกันสัญญาณรบกวนความถี่สูง

ป้องกันสัญญาณรบกวนจากภายใน ซึ่งอาจแพร่ไปยังส่วนอื่นๆของระบบ

หลักการเลือกอุปกรณ์ : เลือกใช้ Ferrite Bead ที่มีพิกัดกระแสตามตัวที่มีกระแสไหลผ่านสูงสุด ซึ่งได้แก่ FB1 มีกระแสไฟฟ้าไหลผ่านสูงสุดที่ 2 แอมป์ ผู้ออกแบบได้เลือกใช้พิกัดกระแส 2.5 แอมป์

F1 : ฟิวส์

หน้าที่ : ป้องกันการลัดวงจรของ IRM-10

หลักการเลือกอุปกรณ์ : การเลือกใช้ฟิวส์ที่มีขนาดถูกต้อง ผู้ใช้ต้องทราบขนาดกระแสปกติในวงจรที่อุณหภูมิห้อง (25 °C) เสียก่อน เมื่อทราบขนาดกระแสดังกล่าวแล้ว จึงเลือกใช้ฟิวส์ที่มีค่าพิกัดกระแส 135% ของค่ากระแสนั้น โดยให้เลือกฟิวส์ขนาดมาตรฐานถัดไปจากค่ากระแสที่คำนวณได้ โดยระบบที่เดิน

ปกติซึ่งมีกระแสในระบบ 250mA ที่อุณหภูมิห้อง ควรเลือกใช้ฟิวส์ขนาด 500 mA (ค่า 135% ของกระแสดังกล่าวคือ 1.35 x 0.25A = 0.3375 A ซึ่งขนาดมาตรฐานของฟิวส์ถัดไปที่ควรเลือกใช้ก็คือขนาด 500mA)

- Current Rating : 500 mA

- Voltage Rating AC : 250 V_{AC}

3.1.5 หลักการออกแบบ PCB

3.1.5.1 การคำนวณความกว้างของตัวนำ

1. หาพื้นที่ตัดขวางจาก

   Area [mils^2] = (Current[Amps]/(k*(Temp_Rise^b[deg.C]))$^{1/c}$

2. หาความกว้างจาก

   Width [mils] = Area[mils^2] / (Thickness[oz]*1.378[mils/oz])

สำหรับมาตรฐาน IPC-2221 ; เลเยอร์ภายใน : k = 0.024, b = 0.44, c = 0.725

สำหรับมาตรฐาน IPC-2221 ; เลเยอร์ภายนอก : k = 0.048, b = 0.44, c = 0.725

3.1.5.2 การกำหนดระยะห่างระหว่างตัวนำ

	Internal layers		External conductors, uncoated		External conductors coated	
Vpk,V	mm	inch	mm	inch	mm	inch
15	0.05	0.002	0.1	0.004	0.05	0.002
30	0.05	0.002	0.1	0.004	0.05	0.002
50	0.1	0.004	0.6	0.024	0.13	0.006
100	0.1	0.004	0.6	0.024	0.13	0.006
150	0.2	0.008	0.6	0.024	0.4	0.016
170	0.2	0.008	1.25	0.05	0.4	0.016
250	0.2	0.008	1.25	0.05	0.4	0.016
300	0.2	0.008	1.25	0.05	0.4	0.016
500	0.25	0.01	2.5	0.1	0.8	0.032
1000	1.5	0.06	5	0.2	2.33	0.092
2000	4	0.158	10	0.4	5.38	0.22
3000	6.5	0.256	15	0.6	8.43	0.34
4000	9	0.355	20	0.79	11.48	0.46
5000	11.5	0.453	25	0.99	14.53	0.58

ภาพที่ 3.17 การกำหนดระยะห่างระหว่างตัวนำ

การออกแบบความกว้างของตัวนำและระยะห่างระหว่างตัวนำจะแบ่งออกเป็น 4 ส่วน ได้แก่

1) แรงดันไฟฟ้ากระแสตรงเอาท์พุทสูงสุด 58 V_{DC} กระแสไฟตรงเอาท์พุท สูงสุด 52.4 A_{DC} บน Bottom Layer (เลเยอร์ภายนอก)

กำหนดให้ความหนาของเส้นทองแดง (Copper Thickness) = 1 mm = 28.818 oz/ft^2

กำหนดอุณหภูมิเพิ่มขึ้น (Temperature Rise) 10 °C

คำนวณหาความกว้างของเส้นทองแดง

1. Area = $(52.4 / 0.048 * 10^{0.44})^{1/0.725}$ = 3833.297 mils^2
2. Width = 3833.297 / (28.818 * 1.378) = 96.528 mils

จะได้ขนาดความกว้างของเส้นทองแดงน้อยที่สุด เท่ากับ 96.528 mil หรือ 2.452 mm.

จากภาพที่ 3.17 จะได้ระยะห่างระหว่างตัวนำน้อยที่สุด เท่ากับ 0.6 mm

2) แรงดันไฟฟ้ากระแสสลับอินพุท 230 V_{rms} หรือเท่ากับ 325.22 V_{Peak} กระแสไฟสลับอินพุต สูงสุด 13.25 A_{AC} บน Bottom Layer (เลเยอร์ภายนอก)

กำหนดให้ความหนาของเส้นทองแดง (Copper Thickness) = 2 oz/ft^2

กำหนดอุณหภูมิเพิ่มขึ้น (Temperature Rise) 10 °C

คำนวณหาความกว้างของเส้นทองแดง

1. Area = $(13.25 / 0.048 * 10^{0.44})^{1/0.725}$ = 575.396 mils^2
2. Width = 575.396 / (2 * 1.378) = 208.779 mils

จะได้ขนาดความกว้างของเส้นทองแดงน้อยที่สุด เท่ากับ 208.779 mil หรือ 5.30 mm

จากภาพที่ 3.17 จะได้ระยะห่างระหว่างตัวนำน้อยที่สุด เท่ากับ 0.8 มิลลิเมตร

3) แรงดันไฟฟ้ากระแสตรงเอาท์พุต 5 V_{DC} กระแสไฟตรงเอาท์พุตสูงสุด 2 A_{DC} (เพาเวอร์เอาท์พุตจาก IRM-10) บน Bottom Layer และ Top Layer (เลเยอร์ภายนอก)

กำหนดให้ความหนาของเส้นทองแดง (Copper Thickness) = 2 oz/ft^2

กำหนดอุณหภูมิเพิ่มขึ้น (Temperature Rise) 10 °C

คำนวณหาความกว้างของเส้นทองแดง

1. Area = $(2 / 0.048 * 10^{0.44})^{1/0.725}$ = 42.393 mils^2
2. Width = 42.393 / (2 * 1.378) = 15.382 mils

จะได้ขนาดความกว้างของเส้นทองแดงน้อยที่สุด เท่ากับ 15.382 mil หรือ 0.391 mm

จากภาพที่ 3.17 จะได้ระยะห่างระหว่างตัวนำน้อยที่สุด เท่ากับ 0.05 mm

4) ณ ตำแหน่งอื่นๆ บน Bottom Layer และ Top Layer (เลเยอร์ภายนอก) จะเป็นแรงดันไฟฟ้าต่ำกระแสไฟฟ้าต่ำ

ผู้ออกแบบได้กำหนดค่าต่างๆ ดังนี้

ความหนาของเส้นทองแดง (Copper Thickness) = 2 oz/ft^2

ขนาดความกว้างของเส้นทองแดงน้อยที่สุด เท่ากับ 10 mil หรือ 0.254 mm

ระยะห่างระหว่างตัวนำน้อยที่สุด เท่ากับ 10 mil หรือ 0.254 mm ซึ่งเพียงพอต่อคุณสมบัติทางไฟฟ้า

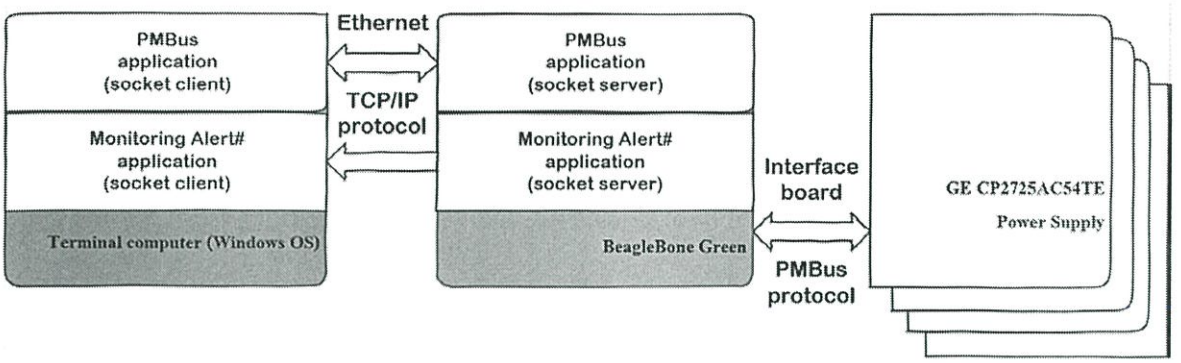
3.2 การออกแบบส่วนต่างๆ ด้านซอฟต์แวร์

3.2.1 การออกแบบระบบ

3.2.1.1 ภาพรวมของระบบ

ระบบที่ออกแบบขึ้นคือระบบควบคุมและตรวจสอบการทำงานของเพาเวอร์ซัพพลายฟ้ากระสแตรงรุ่น CP2725AC54TE ที่ใช้กับเครื่องทดสอบผลิตภัณฑ์ในกระบวนการผลิตของโรงงานด้วยโปรแกรมคอมพิวเตอร์ ระบบนี้ใช้เพื่อควบคุมการทำงานของเพาเวอร์ซัพพลายแต่ละตัวได้อย่างอิสระโดยไม่ต้องใช้ power distribution switch และสามารถแจ้งเตือนผู้ใช้งานให้ทราบถึงการเปลี่ยนแปลงสถานะการทำงานหรือข้อผิดพลาดระหว่างการทำงานของเพาเวอร์ซัพพลายได้ เพื่อให้ผู้ใช้งานสามารถตรวจสอบหรือแก้ไขข้อผิดพลาดระหว่างการทำงานได้อย่างเหมาะสม ภาพที่ 3.1 แสดงส่วนประกอบของระบบควบคุมและตรวจสอบการทำงานของเพาเวอร์ซัพพลายฟ้ากระแสตรงรุ่น CP2725AC54TE

การทำงานของระบบมีสองส่วน ส่วนแรกคือโปรแกรม PMBus ใช้สำหรับควบคุมการทำงานของเพาเวอร์ซัพพลาย โดยให้ผู้ใช้งานพิมพ์คำสั่งเข้าไปในโปรแกรมที่สร้างขึ้นบนคอมพิวเตอร์ จากนั้นคำสั่งจะส่งผ่านเน็ตเวิร์กไปยัง BeagleBone บนบอร์ดจะรับข้อมูลจากผู้ใช้งานมาประมวลผลให้เป็นโปรโตคอล PMBus จากนั้น BeagleBone จะส่งคำสั่ง PMBus ผ่านบัส I^{2}C ไปควบคุมการทำงานเพาเวอร์ซัพพลาย CP2725AC54TE ส่วนที่สองคือโปรแกรม Monitoring Alert# ใช้เพื่อแสดงผลเมื่อเพาเวอร์ซัพพลายมีการเปลี่ยนสถานะการทำงาน โปรแกรมจะตรวจสอบสถานะโลจิกที่ขาสัญญาณ Alert ของเพาเวอร์ซัพพลาย CP2725AC54TE ถ้าโลจิกเปลี่ยนสถานะเป็น ‘0’ โปรแกรมอ่านสถานะการทำงานของเพาเวอร์ซัพพลายทุกตัวบนบัสแล้วส่งผลที่อ่านได้ไปที่คอมพิวเตอร์ของผู้ใช้งาน ทำให้ผู้ใช้งานทราบถึงการเปลี่ยนแปลงสถานะการทำงานหรือข้อผิดพลาดในระหว่างการทำงานของเพาเวอร์ซัพพลาย CP2725AC54TE



ภาพที่ 3.18 block diagram ของระบบควบคุมการทำงานเพาเวอร์ซัพพลาย Ge รุ่น CP2725AC54TE

3.2.1.2 หลักการออกแบบ

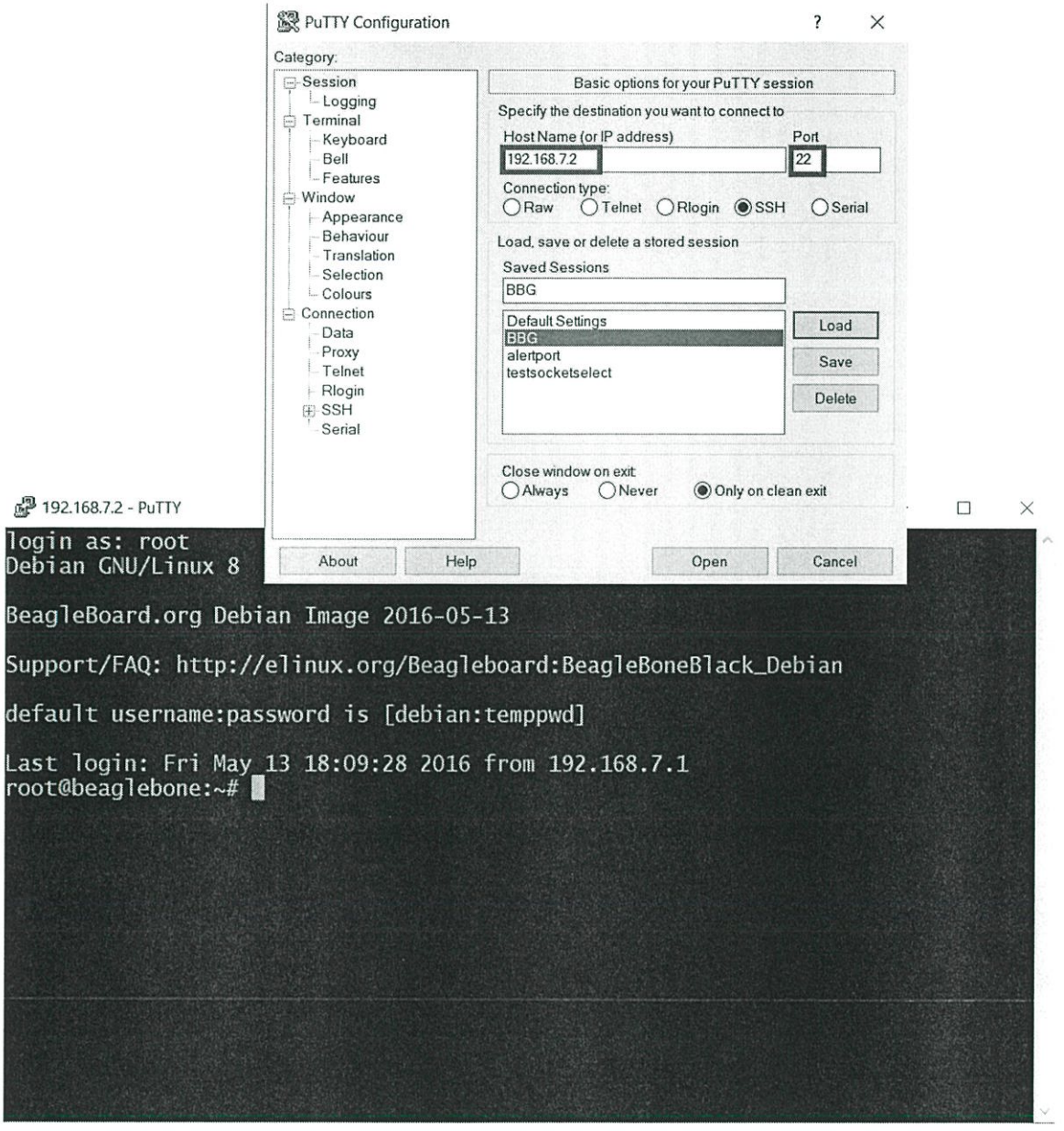
การออกแบบโปรแกรมใช้หลักการ server/client ในการออกแบบ ฝั่งเซิฟเวอร์จะมีเซิฟเวอร์สองตัวทำงานอยู่ตลอดเวลาบนบอร์ด BeagleBone ได้แก่ เซิฟเวอร์ PMBus ใช้สื่อสารกับผู้ใช้งานและควบคุมการสื่อสารทั้งหมดกับเพาเวอร์ซัพพลาย, เซิฟเวอร์ Monitoring Alert# ใช้ส่งข้อมูลที่อ่านได้จากเครื่องจ่ายไปที่ผู้ใช้งาน โดยโปรแกรมฝั่งเซิฟเวอร์จะเขียนด้วยภาษา C

ในส่วนของ PMBus client จะเขียนเป็นโปรแกรมรับค่าจากคีบอร์ดโดยกำหนดรูปแบบการป้อนข้อมูลไว้ เมื่อผู้ใช้ป้อนข้อมูลตามรูปแบบได้ถูกต้องจึงส่งข้อมูลนั้นไปที่เซิฟเวอร์ และส่วนของ Monitoring Alert# client อาจใช้เป็น telnet หรือจะเขียนไคลเอนท์เพื่อรับข้อมูลที่ส่งมาจากเซิฟเวอร์ก็ได้ โปรแกรมฝั่งไคลเอนท์จะเขียนด้วยภาษา visual C++

3.2.2 อุปกรณ์ที่ใช้ในการทดลอง

3.2.2.1 BeagleBone

ในการสร้างเครือข่ายการสื่อสารผ่าน ethernet ระหว่างบอร์ด BeagleBone กับคอมพิวเตอร์จะใช้สาย USB จะทำให้บอร์ด BeagleBone ใช้โหมดเครือข่ายพิเศษเรียกว่า Internet-over-USB เพื่อสร้างเครือข่ายเสมือน (virtual network) ระหว่างบอร์ด BeagleBone กับคอมพิวเตอร์และยังใช้เป็นสายส่งพลังงานให้กับบอร์ด BeagleBone (การเชื่อมต่อด้วยสาย USB ใช้เพื่ออำนวยความสะดวกในการทดลอง เนื่องจากการใช้สาย ethernet จะมีความซับซ้อนในการตั้งค่าและยังต้องมีวงจรภายนอกจ่ายไฟเลี้ยงให้บอร์ด BeagleBone ทำให้เกิดความล่าช้าในการทดลอง) เมื่อเชื่อมต่อสายเรียบร้อยจึงทดสอบการเชื่อมต่อโดยใช้โปรแกรม PuTTy เชื่อมต่อไปที่เซิฟเวอร์ secure shell (SSH) ของ BeagleBone โดยใส่ IP address 192.168.7.2 port 22 ในโปรแกรม เมื่อปรากฏคอนโซลดังภาพที่ 3.2 แสดงว่า BeagleBone กับคอมพิวเตอร์เชื่อมต่อกันผ่านเน็ตเวิร์กอย่างสมบูรณ์



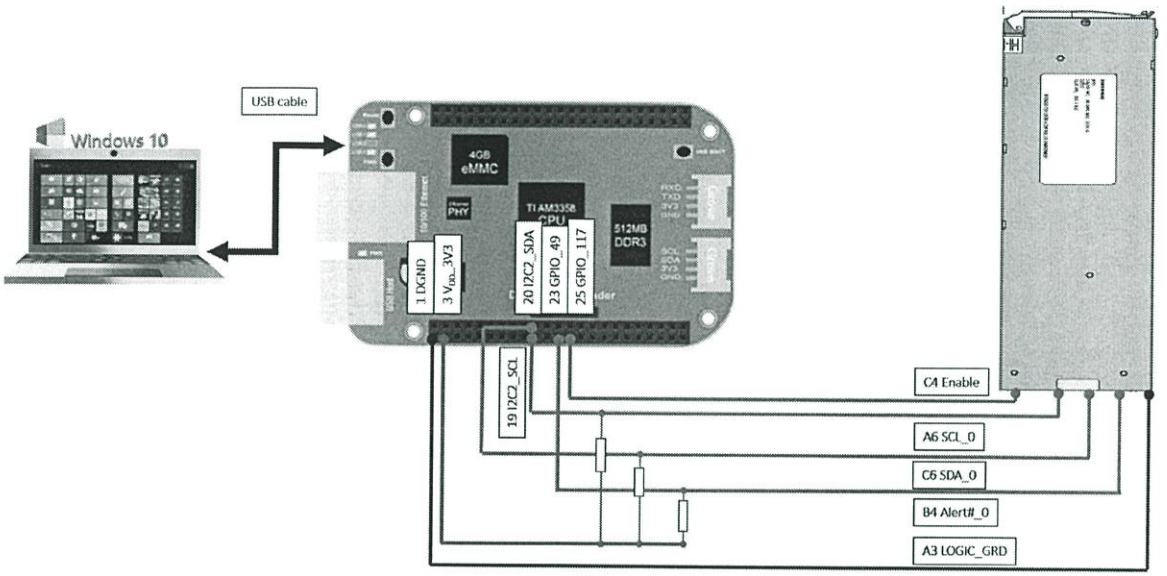
ภาพที่ 3.19 ทดสอบการเชื่อมต่อบอร์ด BeagleBone กับคอมพิวเตอร์

3.2.2.2 เพาเวอร์ซัพพลาย CP2725AC54TE

ในการใช้งานบนบัส I^2C เพาเวอร์ซัพพลายจะต้องมีแอดเดรส เมื่อต่อตัวต้านทาน 30k ระหว่างขา UNIT_ADDR กับ LOGIC_GRD ของเพาเวอร์ซัพพลายและต่อขา SHELF_ADDR กับขา LOGIC_GRD จะทำให้แอดเดรสของไมโครคอนโทรลเลอร์ภายในเพาเวอร์ซัพพลายมีค่าเป็น 0x44

การเชื่อมต่ออุปกรณ์ทั้งสองบนบัส I^2C จะใช้ขาสัญญาณ SCL_0, SDA_0, Alert#_0, Enable และ LOGIC_GRD ของเพาเวอร์ซัพพลายต่อเข้ากับขา 19, 20, 23, 25 และขา 1 บน header P9 ของ BealgeBone ตามลำดับและทุกสัญญาณยกเว้น Enable และ LOGIC_GRD ต้องต่อกับไฟ V_{DD} 3.3v ผ่าน

ตัวต้านทานพูลอัพ ภาพที่ 3.3 แสดงการเชื่อมต่อฮาร์ดแวร์เพื่อใช้ในการทดลอง ทดสอบการเชื่อมต่อกับเพาเวอร์ซัพพลายด้วยคำสั่ง I2C-tools ของระบบปฏิบัติการ Linux โดยใช้ SSH ใน BeagleBone จากนั้นพิมพ์คำสั่ง i2cdetect -r 2 ถ้าการเชื่อมต่อเป็นปกติจะปรากฏเอ้าพุทตามภาพที่ 3.4 จะเห็นว่ามีแอดเดรส 0x44 ปรากฏอยู่ในตารางตรงตามที่กำหนดไว้



ภาพที่ 3.20 การเชื่อมต่อฮาร์ดแวร์สำหรับใช้ในการทดลอง

```
root@beaglebone:~# i2cdetect -r 2
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-2 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- 44 -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- 74 -- -- --
root@beaglebone:~# i2cset 2 0x44 0x01 0x00 bp
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will write to device file /dev/i2c-2, chip address 0x44, data address
0x01, data 0x00, mode byte.
PEC checking enabled.
Continue? [Y/n] y
root@beaglebone:~#
```

ภาพที่ 3.21 คอนโซลเอ้าพุทเมื่อใช้คำสั่ง i2cdetect และ i2cset ตามลำดับ

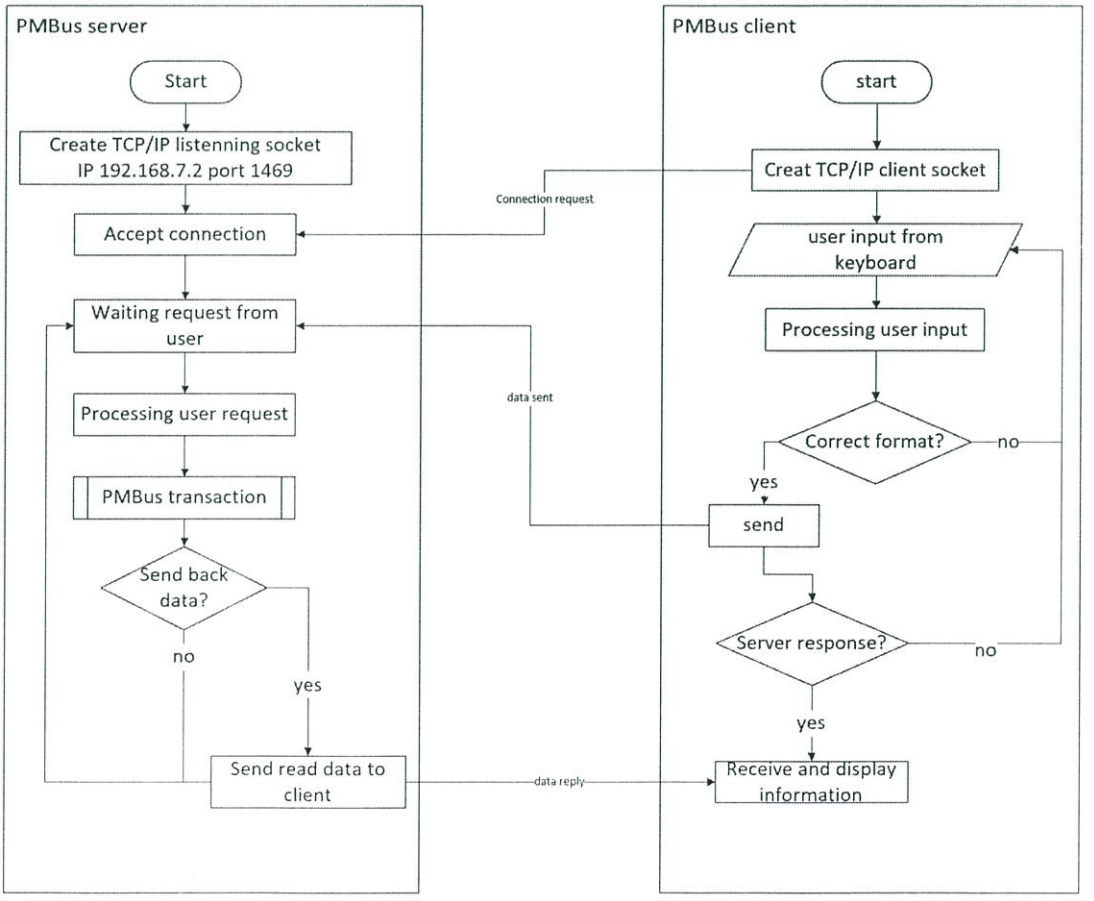
3.2.3 การเขียนโปรแกรม

3.2.3.1 การเขียนโปรแกรม PMBus

โปรแกรมด้านไคลเอนท์เขียนด้วยภาษา C++ คอมไพล์ด้วย Microsoft Visual C++ v14.11.25325 ลักษณะโปรแกรมเป็นแบบ command line interface ให้ผู้ใช้ป้อนคำสั่งแล้วส่งไปที่โปรแกรมเซิฟเวอร์ ภาพที่ 3.5 (ฝั่งขวา) แสดง flow chart ของโปรแกรม PMBus ไคลเอนท์ที่เขียนขึ้น

ขั้นตอนการเขียนโปรแกรมไคลเอนท์

1. เริ่มโปรแกรมจะสร้างซ๊อคเก็ตด้วย WinSock API และเชื่อมต่อไปยังเซิฟเวอร์ที่ IP 192.168.7.2 port 1469
2. เมื่อเชื่อมต่อกับเซิฟเวอร์สำเร็จ โปรแกรมจะรอรับคำสั่งจากผู้ใช้งานผ่านคีบอร์ด
3. ข้อมูลจะถูกตรวจสอบว่าตรงกับรูปแบบคำสั่งที่กำหนดไว้หรือไม่ ถ้าข้อมูลอยู่ในรูปแบบที่กำหนดไว้จะถูกส่งไปที่โปรแกรมเซิฟเวอร์ ถ้าข้อมูลไม่อยู่ในรูปแบบที่กำหนด โปรแกรมจะวนกลับไปรับคำสั่งใหม่
4. จากนั้นถ้าเซิฟเวอร์มีการส่งข้อมูลกลับมา โปรแกรมจะแสดงผลข้อมูลที่ได้รับจากเซิฟเวอร์แล้ววนกลับไปที่ข้อ 2



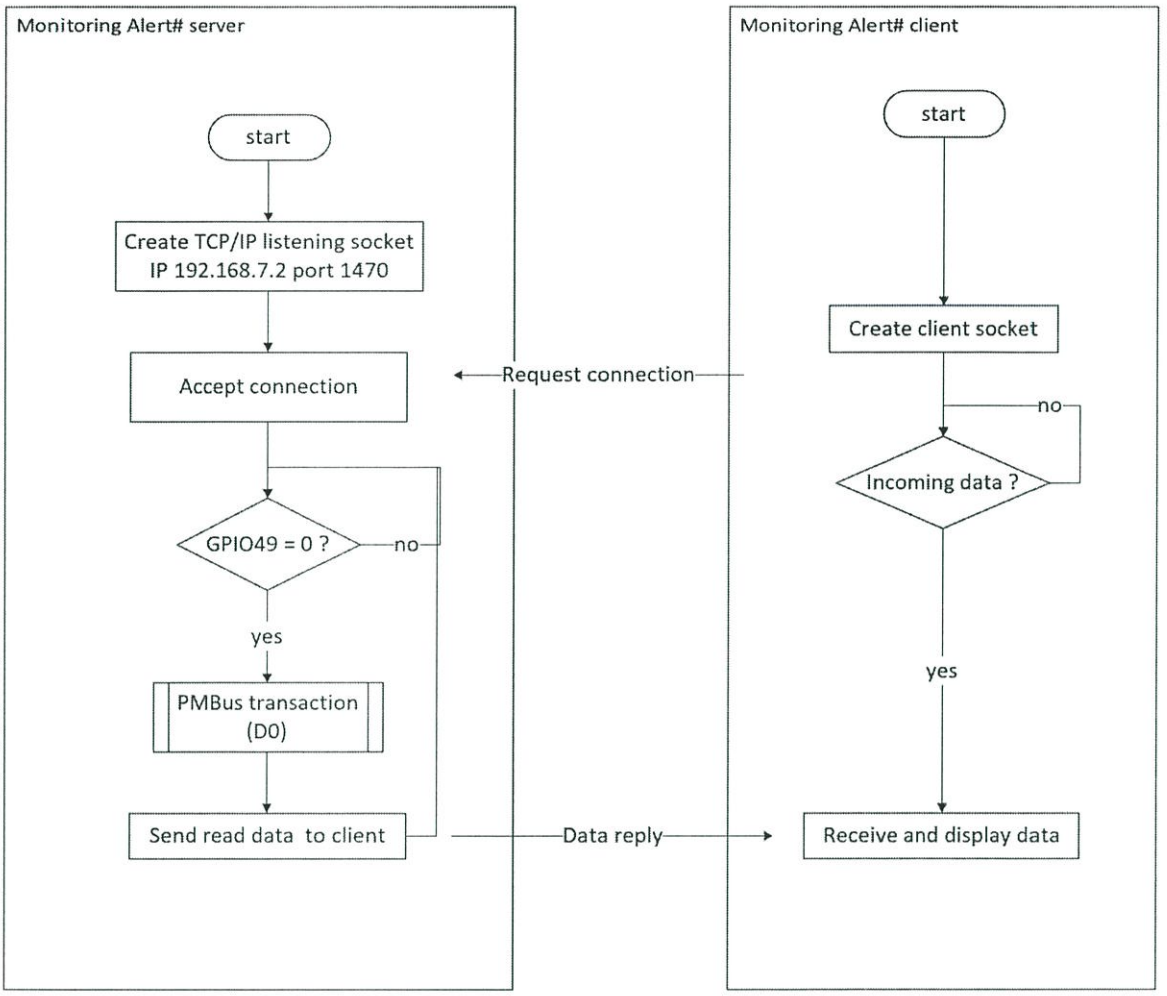
ภาพที่ 3.22 แสดง flow chart ของโปรแกรม PMBus

โปรแกรมด้านเซิฟเวอร์ที่สร้างขึ้นจะเขียนด้วยภาษา C คอมไพล์ด้วย gcc โปรแกรมทำงานบนบอร์ด BeagleBone ภาพที่ 3.9 (ฝั่งซ้าย) แสดง flow chart ของโปรแกรมเซิฟเวอร์ที่เขียนขึ้น
ขั้นตอนการเขียนโปรแกรมเซิฟเวอร์

1. สร้างซ๊อคเกต TCP/IP เพื่อรอรับการเชื่อมต่อที่ IP 192.168.7.2 port 1469
2. เมื่อเซิฟเวอร์ตอบรับการเชื่อมต่อ โปรแกรมจะอยู่ในสถานะรอรับข้อมูลจากไคลเอนท์ที่กำลังเชื่อมต่ออยู่
3. นำข้อมูลที่ได้รับจากไคลเอนท์ไปประมวลผล เพื่อเรียกใช้โปรโตคอล PMBus ที่สอดคล้องกับข้อมูลที่ได้รับเพื่อสื่อสารกับเพาเวอร์ซัพพลาย
4. ถ้าไม่มีข้อมูลที่ต้องส่งกลับไปไคลเอนท์ โปรแกรมจะวนกลับไปรอรับข้อมูลจากไคลเอนท์

3.2.3.2 การเขียนโปรแกรม Monitoring Alert#

โปรแกรมด้านเซิฟเวอร์จะเขียนด้วยภาษา C และทำงานอยู่บน BeagleBone เช่นเดียวกับโปรแกรม PMBus แต่เซิฟเวอร์จะใช้พอร์ต 1470 หลักการทำงานคือ โปรแกรมจะวนลูปอ่านค่าสถานะโลจิกที่ขา GPIO 49 ของ BeagleBone ถ้ามีการเปลี่ยนระดับสัญญาณจาก ‘1’ เป็น ‘0’ โปรแกรมจะสั่งอ่านสถานะการทำงานจากเพาเวอร์ซัพพลาย CP2725AC54TE ทุกตัวบนบัสด้วยโปรโตคอล PMBus แล้วส่งข้อมูลที่อ่านได้ไปแสดงผลที่ฝั่งไคลเอนท์ ภาพที่ 3.6 แสดง flow chart ของโปรแกรม Monitoring Alert#



ภาพที่ 3.23 flow chart ของโปรแกรม monitoring alert

บทที่ 4
ผลการวิจัย

4.1 ผลการวิจัยทางฮาร์ดแวร์

หลังจากทำการออกแบบวงจรและ PCB จะทำการทดสอบการใช้งาน โดยแบ่งออกเป็น 3 ส่วนคือ

4.1.1 ทดสอบการส่งผ่านกำลังงานทางด้านอินพุตและเอาต์พุต

โดยกำลังไฟฟ้าทางด้านอินพุตจะส่งจากเต้ารับ AC ผ่าน Trace บน PCB เข้าสู่เพาเวอร์ซัพพลาย และกำลังไฟฟ้าทางด้านเอาต์พุตจะส่งจากเพาเวอร์ซัพพลายผ่านเส้น Trace บน PCB และออกสู่เอาต์พุตคอนเนคเตอร์

โดยในการทดสอบจะทำการดึงกระแสไฟฟ้าผ่าน DC ELECTRONIC LOAD ยี่ห้อ Chroma รุ่น 63106 สามารถใช้งานได้ที่กำลังไฟฟ้าสูงสุด 600 วัตต์ ดังนั้นที่แรงดันไฟฟ้า 54 V สามารถดึงกระแสได้สูงสุด 11.11 แอมป์

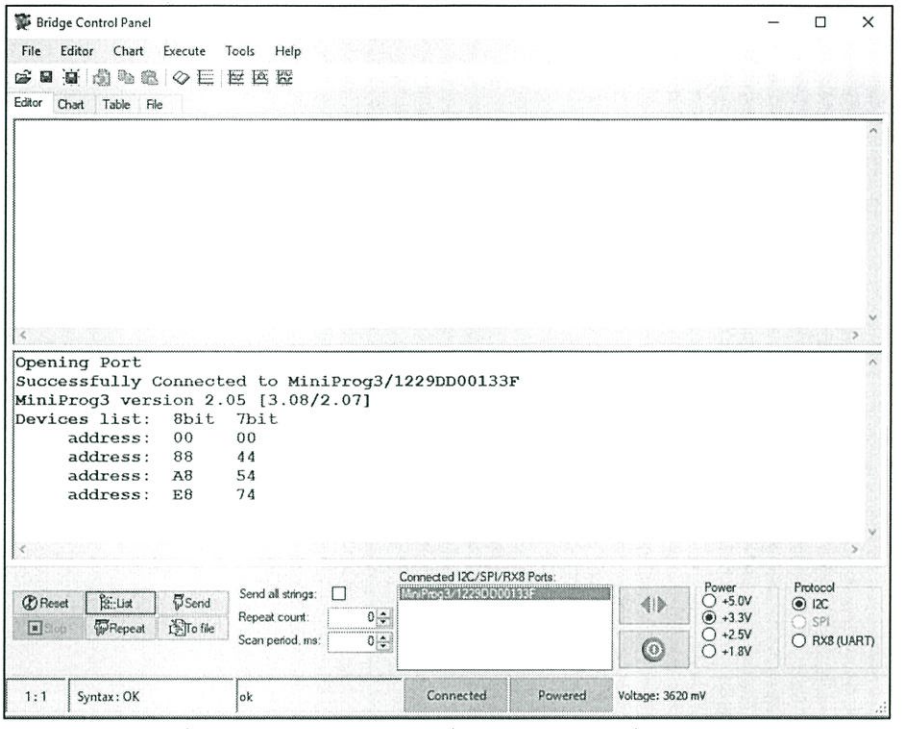
ทำการทดสอบดังนี้

ตารางที่ 4.1 ผลการทดสอบการดึงกระแสผ่านบอร์ดอินเตอร์เฟซ

แรงดันเอาต์พุต	กระแสเอาต์พุต	กำลังไฟเอาต์พุต
54 V	5 A	270 W
54 V	10 A	540 W

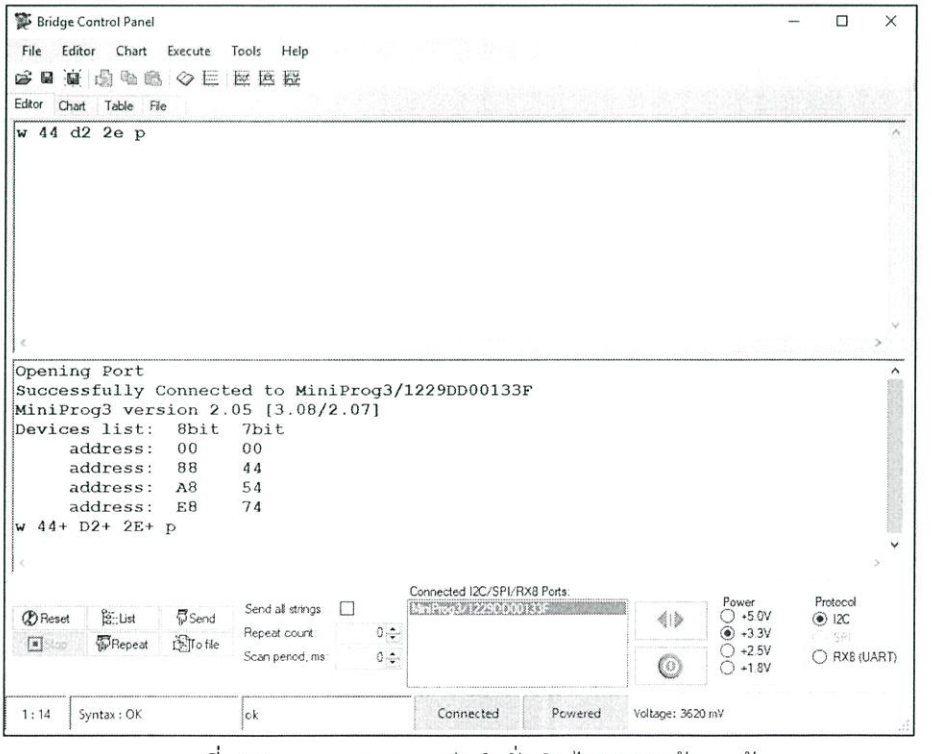
4.1.2 ทดสอบการสื่อสารผ่านโหมด PMBUS

1. ใช้อุปกรณ์ MiniProg3 ทำหน้าที่เป็น Master เชื่อมต่อผ่านโปรแกรม Bridge Control Panel ตรวจหา Slave ที่ต่ออยู่บนบัส ซึ่งเพาเวอร์ซัพพลายแต่ละตัวทำหน้าที่เป็น Slave



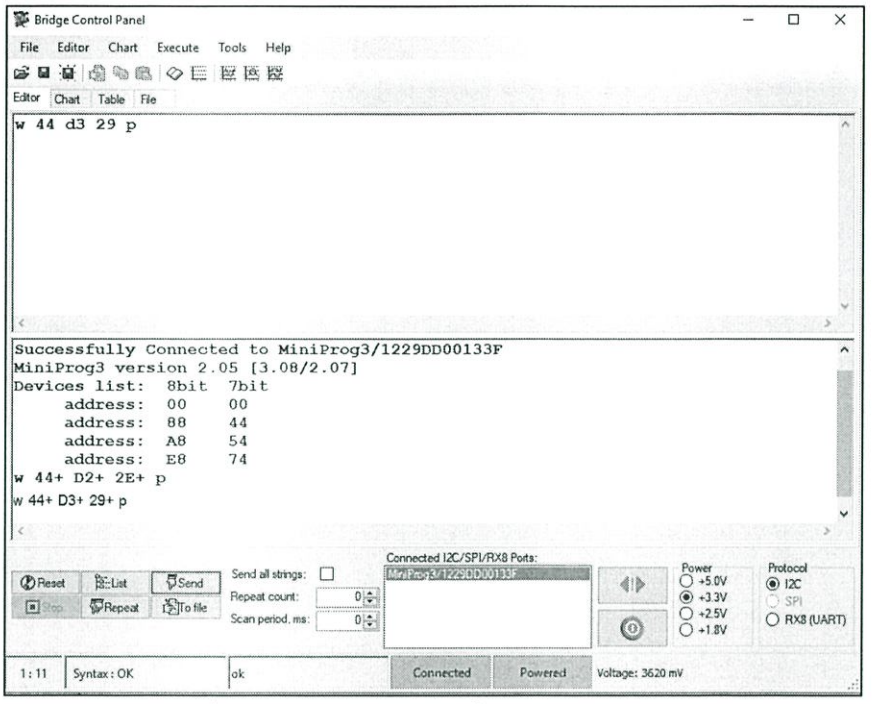
ภาพที่ 4.1 ผลการตรวจสอบที่อยู่ของ Slave ที่ต่ออยู่บนบัส

2. ทดสอบส่งคำสั่งเพื่อเปิดไฟ LED ด้านหน้า



ภาพที่ 4.2 ผลการทดสอบส่งคำสั่งเปิดไฟ LED ด้านหน้า

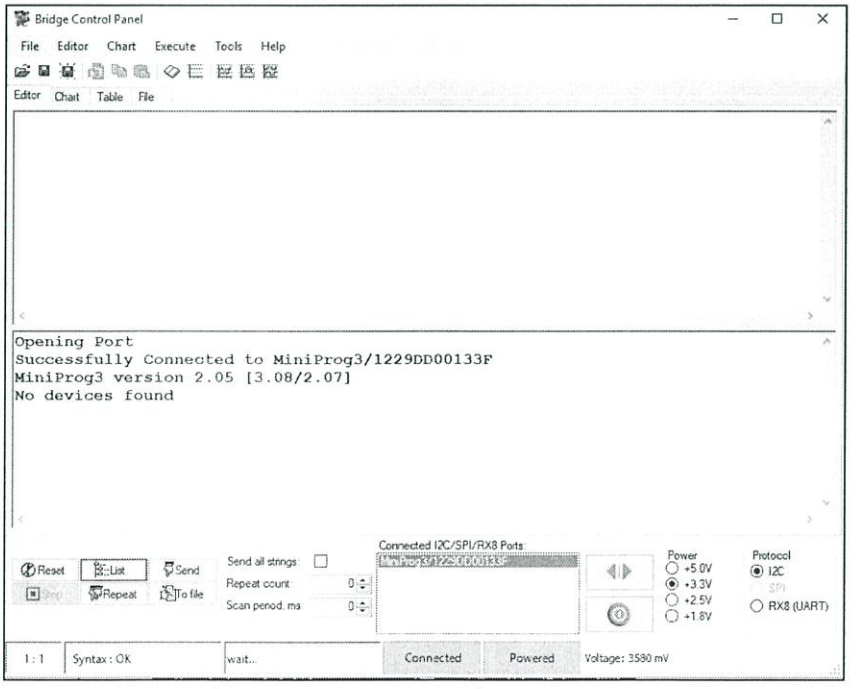
3. ทดสอบส่งคำสั่ง เพื่อปิดไฟ LED ด้านหน้า



ภาพที่ 4.3 การทดสอบส่งคำสั่งปิดไฟ LED ด้านหน้า

4.1.3 ทดสอบการสื่อสารผ่านโหมดAnalog(ใช้ไอซี PCA9535 แปลงสัญญาณ I2C เป็น GPIO)

1. ใช้อุปกรณ์ MiniProg3 ทำหน้าที่เป็น Master เชื่อมต่อผ่านโปรแกรม Bridge Control Panel ตรวจหา Slave ที่ต่ออยู่บนบัส ซึ่งไอซี PCA9535 แต่ละตัวทำหน้าที่เป็น Slave



ภาพที่ 4.4 การทดสอบส่งคำสั่งผ่านโหมด Analog

จากการทดลองพบว่า Master ไม่สามารถตรวจพบตัวอุปกรณ์ที่ติดตั้งอยู่บนบัส I^2C

4.2 ผลการวิจัยทางซอฟต์แวร์

4.2.1 วิธีการที่ใช้ทดสอบการทำงานของโปรแกรม

การทดสอบการทำงานของโปรแกรม PMBus ใช้วิธีการป้อนคำสั่งให้โปรแกรมทำงานตามฟังก์ชันต่างๆแล้วเปรียบเทียบเอ้าพุทที่ได้จากโปรแกรมกับเครื่องมือวัดทางไฟฟ้าในกรณีที่เอ้าพุทเป็นสัญญาณไฟฟ้า ในกรณีที่เป็นข้อมูลสถานะที่ได้อ่านได้จากเพาเวอร์ซัพพลายจะนำไปตรวจสอบกับดาต้าชีทของเพาเวอร์ซัพพลาย

การทดสอบการทำงานของโปรแกรม Alert ใช้วิธีทำให้ขาสัญญาณ Alert# ของเพาเวอร์ซัพพลายเปลี่ยนระดับสัญญาณจาก 'HI' เป็น 'LO' ตามเงื่อนไขที่ดาต้าชีทของเพาเวอร์ซัพพลาย CP2725AC54TE ระบุไว้ แล้วตรวจสอบความถูกต้องของค่าสถานะที่อ่านได้เปรียบเทียบกับข้อมูลที่ระบุไว้ในดาต้าชีท

4.2.2 ผลการทดลอง

ผลทดสอบการทำงานของโปรแกรม PMBus แสดงในตารางที่ 4.1-4.2 และผลทดสอบโปรแกรม alert แสดงในตารางที่ 4.3

ตารางที่ 4.2 ผลการทดสอบเอ้าพุทของโปรแกรม PMBus เทียบกับผลที่ได้จากเครื่องมือวัด

กรณีทดสอบ	อินพุท (โปรแกรม)	เอ้าพุท (โปรแกรม)	เครื่องมือวัด (Chroma 63106)	ผลทดสอบ
เปิดเอ้าพุท	on psu1	54.08 V	54.11 V	ผ่าน
ปิดเอ้าพุท	off psu1	0 V	5.07 V	ผ่าน*
อ่านค่าอินพุท (Load 10A)	input psu1	V_{IN} : 221 V P_{IN} : 589 W	P_{IN} = 568.97 W**	ผ่าน
กำหนดแรงดันเอ้าพุท	vout psu1 51.23 vout psu1 42 vout psu1 58	51.30 V 42.06 V 58.07 V	51.39 V 42.13 V 58.16 V	ผ่าน ผ่าน ผ่าน
อ่านสถานะการทำงาน (output on, Load 10 A)	status psu1	Status2: 0x50 Status1: 0x01 Alarm2: 0x00 Alarm1: 0x00		ผ่าน

		V_{OUT} : 54.06 V I_{OUT} : 10.00 A Temp.: 27 'C	53.96 V 9.97 A	
อ่านสถานะการทำงาน (output off)	status psu1	Status2: 0x51 Status1: 0x04 Alarm2: 0x00 Alarm1: 0x00 V_{OUT} : 0 V I_{OUT} : 0 A Temp.: 27 'C	 4.88 V 0 A	ผ่าน

*

**ได้จากการคำนวณซึ่งวัด V_{OUT} ได้ 53.96 V, I_{OUT} = 9.97 V, ใช้ค่าประสิทธิภาพ = 94.5% และไม่รวม idle power

ตารางที่ 4.3 ผลการทดสอบเอ้าพุทของโปรแกรม PMBus เทียบกับข้อมูลดาต้าชีท CP2725AC54TE

กรณีทดสอบ	อินพุท (โปรแกรม)	เอ้าพุท (โปรแกรม)	ข้อมูลจากดาต้าชีท	ผลการทดสอบ
อ่านค่าอินพุท (output on, no load)	input psu1	V_{IN} : 223 V P_{IN} : 17 W	Typ idle Power 16 W	ผ่าน
ทดสอบไฟ LED หน้าเครื่อง -on -off	 frontled psu1 on frontled psu1 off	 Status1: 0x03* Status1: 0x01*	 Bit 2 = '1' Bit 2 = '0'	 ผ่าน ผ่าน

*อ่านสถานะการทำงานในขณะที่เปิดเอ้าพุท

ตารางที่ 4.4 ผลการทดสอบโปรแกรม alert

กรณีทดสอบ	**อินพุท**	**เอ้าพุท (โปรแกรม)**	**ผลการทดสอบ**
hot plug PSU module		โปรแกรมรายงานสถานะของเพาเวอร์ซัพพลาย**	ผ่าน
disconnect PSU module		โปรแกรมรายงานสถานะของเพาเวอร์ซัพพลาย**	ผ่าน
Invalid instruction	0x02 0x19*	โปรแกรมรายงานสถานะของเพาเวอร์ซัพพลายอย่างต่อเนื่อง**	ผ่าน
แก้ invalid instruction	clrfault psu1 (ป้อนคำสั่งในโปรแกรม PMBus)	โปรแกรมหยุดรายงานสถานะของเพาเวอร์ซัพพลาย**	ผ่าน

*ใช้คำสั่ง i2cset 2 0x44 0x02 0x19 bp บน BeagleBone

**ภาพการแสดงผลของโปรแกรมแสดงในภาคผนวก

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

เนื่องจากในปัจจุบันอุปกรณ์ที่ทำหน้าที่กระจายและเปิด-ปิดแรงดันไฟที่จ่ายให้กับ UUT (Unit Under Test) ที่ชื่อว่า PDU (Power Distribution Unit) ซึ่งติดตั้งอยู่ในตู้ทดสอบ (Tester) มีราคาค่อนข้างสูง เพื่อลดค่าใช้จ่าย ทางบริษัทได้มีโครงงานให้พัฒนาระบบควบคุมเพาเวอร์ซัพพลายผ่านโครงข่าย โดยทำการออกแบบระบบใหม่ทั้งหมด ในการออกแบบระบบจะแบ่งออกเป็น 2 ส่วน คือ ฮาร์ดแวร์และซอฟต์แวร์ ในส่วนของการออกแบบทางด้านฮาร์ดแวร์ มีขั้นตอนการจัดทำ ดังนี้

1. ศึกษาการใช้งานเพาเวอร์ซัพพลายและบอร์ดบีเกิ้ลโบน
2. เลือกสัญญาณที่จะนำมาใช้งาน
3. เลือกอุปกรณ์เสริมที่จะนำมาใช้บนบอร์ดอินเตอร์เฟซ
4. ออกแบบระบบและออกแบบวงจรของบอร์ดอินเตอร์เฟซ
5. สร้าง Footprint และ 3D Model ของอุปกรณ์บนบอร์ดอินเตอร์เฟซ
6. ออกแบบลายวงจร PCB
7. จัดประกอบอุปกรณ์และทดสอบใช้งาน

ในส่วนของการออกแบบทางด้านซอฟต์แวร์ มีขั้นตอนการจัดทำ ดังนี้

1. เขียนโปรแกรมควบคุมการทำงานเพาเวอร์สัพพลายด้วยโปรโตคอล PMBus บนบอร์ด BeagleBone

2. เขียนโปรแกรม TCP/IP ให้สามารถสั่งงานเพาเวอร์สัพพลายจากคอมพิวเตอร์ที่อยู่บนโครงข่ายเดียวกับบอร์ด BeagleBone ได้

3. เขียนโปรแกรมตรวจสอบสถานการณ์ทำงานเพาเวอร์สัพพลายบนบอร์ด BeagleBone ซึ่งหากมีการเปลี่ยนแปลงสถานะให้แสดงผลไปที่คอมพิวเตอร์ของผู้ใช้งาน

จากการทดสอบใช้งานระบบพบว่าสามารถใช้งานได้ในระดับหนึ่ง ซึ่งโครงงานนี้จะต้องถูกพัฒนาต่อเพื่อจะสามารถนำไปใช้งานได้จริงและเป็นที่ยอมรับของลูกค้าต่อไป

5.2 ข้อเสนอแนะ

ในการพัฒนาต่ออาจเลือกใช้งานเฉพาะในโหมด PMBus เพื่อลดความซับซ้อนของระบบเนื่องจากการใช้งานในโหมด PMBus เพียงอย่างเดียวก็สามารถใช้งานได้ครอบคลุมทุกฟังก์ชัน

ในการพัฒนาต่ออาจเลือกใช้คอนโทรลเลอร์ประเภทอื่นที่มีความเหมาะสมมากกว่าบอร์ด Beaglebone ต่อการติดตั้งและใช้งานจริง

เพื่อให้เป็นไปตามมาตรฐานอุตสาหกรรมควรใช้แท่น Bus Bar ในการส่งผ่านกำลังงานเอาต์พุตหลัก

เอกสารอ้างอิง

[1] General Electric Company. GE CP2725AC54TE Compact Power Line High Efficiency Rectifier datasheet.;2016.

[2] Dr. Derek Molloy. Exploring BeagleBone. Indianapolis: John Wiley&Sons,Inc; 2015.

[3] นายสุเมศ แซ่ตั๊น. การศึกษาลินุกซ์เคอร์เนล ด้วย Raspberry Pi: พื้นฐานการทำงานของลินุกซ์ดีไวซ์ไดร์เวอร์ (ตอนที่ 1). แหล่งที่มา :http://embeddedmaker.com/article/basic_linux_device_driver/ . 18 กันยายน 2560

[4] NXP Semiconductors. UM10204 I2C-bus specification and user manual. Rev6.;2014

[5] System Management Interface Forum. System Management Bus(SMBus) Specification. Version 3.0.; 2014.

[6] System Management Interface Forum. Power System Mangement Protocol Specification Part II - Command Language. Revision 1.1; 2007.

[7] IBM International Technical Support Organization. TCP/IP Tutorial and Technical Overview.; 2006

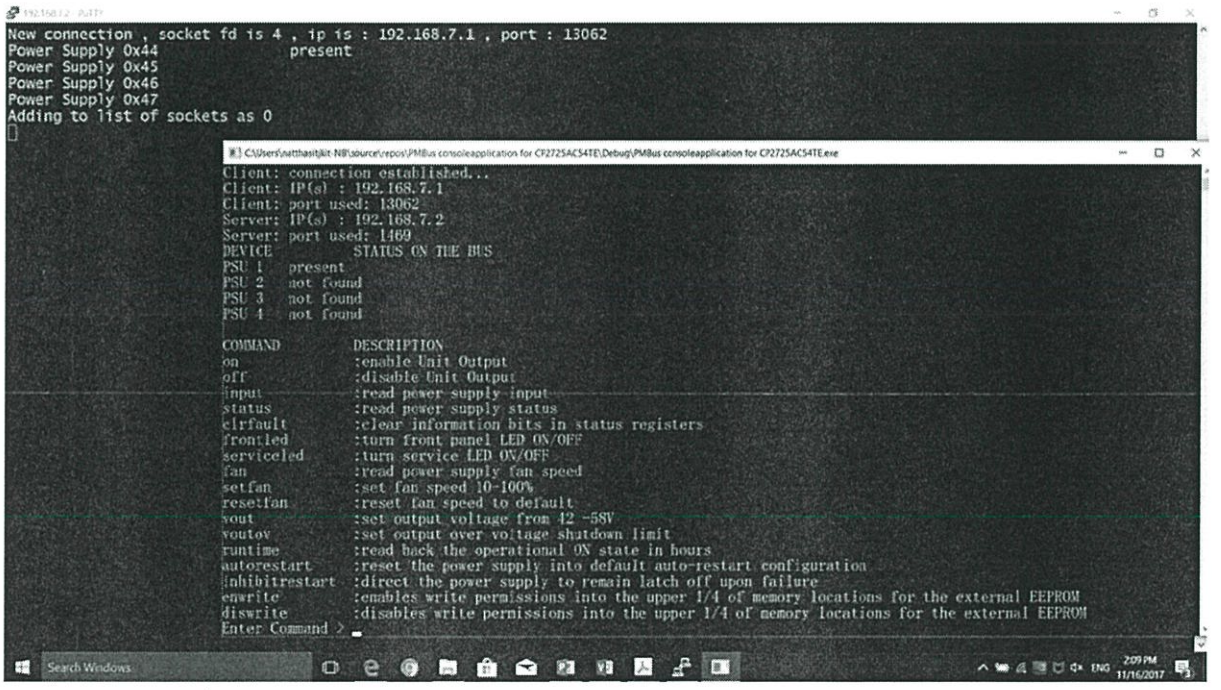
ภาคผนวก ก.

การใช้งานโปรแกรมที่ออกแบบ

ก.1 การใช้งานโปรแกรม PMBus

เมื่อพิมพ์คำสั่ง ./testserver บน BeagleBone โปรแกรม PMBus(server) จะเริ่มทำงานและพร้อมรับการเชื่อมต่อจากคอมพิวเตอร์เครื่องอื่นที่จะเชื่อมต่อเข้ามา บนคอมพิวเตอร์ของผู้ใช้งานจะต้องรันโปรแกรม PMBus consoleapplication for CP2725AC54TE.exe ซึ่งอยู่ในโฟลเดอร์ solution ของ visual studio

เมื่อคอมพิวเตอร์เชื่อมต่อกับเซิฟเวอร์บน BeagleBone ผู้ใช้งานจะสามารถพิมพ์คำสั่งเพื่อควบคุมการทำงานของเพาเวอร์ซัพพลายได้ดังรูป ก-1 หน้าต่างคอนโซลด้านล่างขวาคือโปรแกรมสำหรับผู้ใช้งาน



ภาพที่ ก-1 แสดงโปรแกรมควบคุมการทำงานเพาเวอร์ซัพพลาย CP2725AC54TE

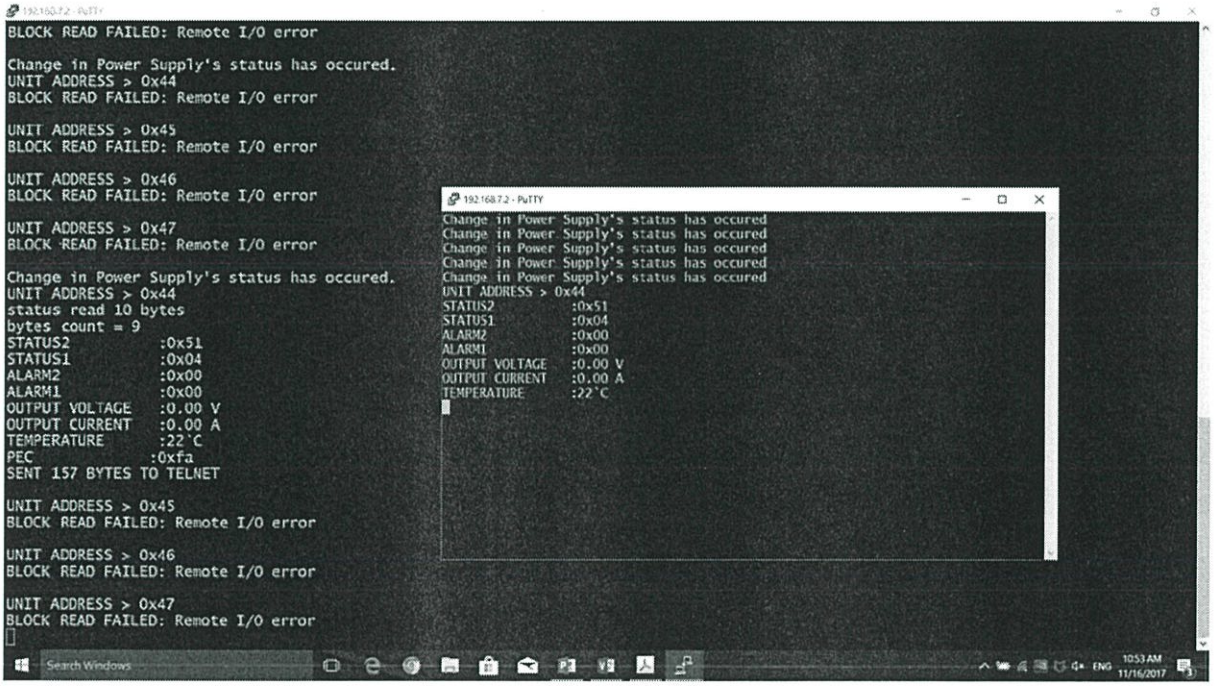
โดยผู้ใช้งานจะสามารถพิมพ์คำสั่งต่างๆเพื่อใช้งานเพาเวอร์ซัพพลายได้ตามตารางที่ ก-1 คอลัมน์ที่หนึ่งคือรูปแบบคำสั่งที่ต้องพิมพ์ใส่โปรแกรม เช่น ต้องการเปิดเอ้าพุทของเพาเวอร์ซัพพลายตัวที่ 2 จะต้องพิมพ์ on psu2

ตารางที่ ก-1 อธิบายรูปแบบคำสั่งและหน้าที่ของคำสั่งที่ใช้ควบคุมเพาเวอร์ซัพพลาย

Command format	Command description
on [PSU-NAME] off [PSU-NAME]	Output ON Output OFF
clrfault [PSU-NAME]	Clear PSU Status register
vout [PSU-NAME] [VALUE]	Set Vout
voutov [PSU-NAME] [VALUE]	Set OV fault limit
status [PSU-NAME]	Read PSU Status&alarm register, Vout, Iout, T
frontled [PSU-NAME] on	Turn on front panel LEDs
frontled [PSU-NAME] off	Turn off front panel LEDs
serviceled [PSU-NAME] on	Turn on Service LED
serviceled [PSU-NAME] off	Turn off Service LED
enwrite [PSU-NAME]	Enable EEPROM write
diswrite [PSU-NAME]	Disable EEPROM write
inhibitrestart [PSU-NAME]	Latch upon failure
autorestart [PSU-NAME]	Hiccup
input [PSU-NAME]	Read Vin and Pin
runtime [PSU-NAME]	Accumulated ON state
setfan [PSU-NAME] [VALUE]	Fan speed control
resetfan [PSU-NAME]	Stop fan control
fan [PSU-NAME]	Read fan speed
input [PSU-NAME]	Read Vin and Pin

ก.2 การใช้งานโปรแกรม Monitoring Alert

ผู้ใช้งานอาจเปิดโปรแกรมนี้เพื่อเอาไว้ตรวจสอบว่าเพาเวอร์ซัพพลายทำงานเป็นปกติหรือไม่ โดยต้องรันโปรแกรม CP2725AC54TE Alert consoleapplication.exe ซึ่งอยู่ในโฟลเดอร์ solution ของ visual studio เพื่อเชื่อมต่อไปที่เซิฟเวอร์บน BeagleBone โปรแกรมจะแสดงข้อความบนหน้าจอก็ต่อเมื่อเกิดการเปลี่ยนสถานการณ์ทำงาน(หรือการเปลี่ยนแปลงในสเตตัสบิทของเพาเวอร์ซัพลาย) ภาพที่ ก-2 แสดงตัวอย่างโปรแกรมเมื่อ hot-plug เพาเวอร์ซัพพลายเข้าไปในระบบ



ภาพที่ ก-2 แสดงตัวอย่างโปรแกรมเมื่อ hot-plug เพาเวอร์ซัพพลายเข้าไปในระบบ