	ใบเนื้อหา	หน้าที่ 1
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

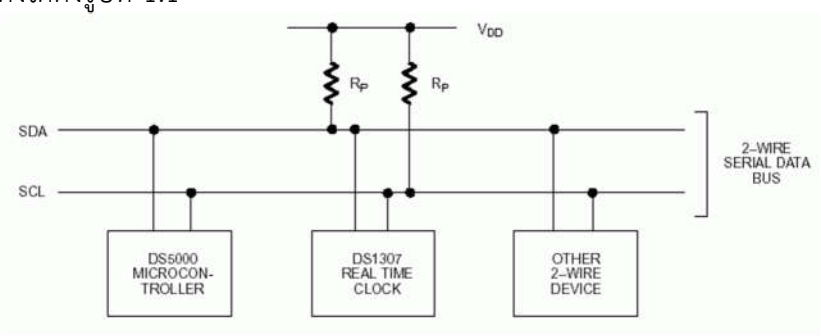
หน่วยที่ 8 การรับส่งข้อมูลอนุกรมแบบต่าง ๆ
การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

1. รูปแบบข้อมูลของการสื่อสารแบบ I2C

I2C Bus (I^2C) ย่อมาจาก Inter Integrate Circuit Bus (IIC) นิยมเรียกสั้นๆว่า I^2C (ไอ-แสคว-ซี-บัส) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) แบบ Half Duplex เพื่อใช้ติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอก ซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ Serial data (SDA) และสาย Serial clock (SCL) ซึ่งสามารถเชื่อมต่ออุปกรณ์จำนวนหลาย ๆ ตัว เข้าด้วยกันได้ ทำให้ MCU ใช้ขาพอร์ตเพียง 2 ขาเท่านั้น

1.1 ลักษณะการเชื่อมต่ออุปกรณ์แบบ I2C Bus

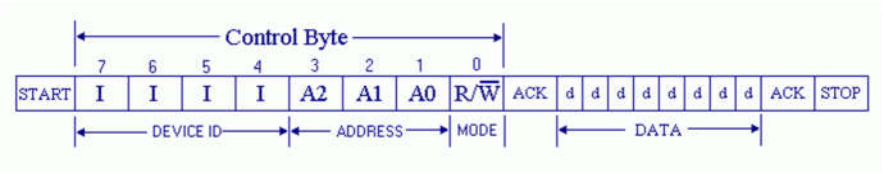
I2C Bus ใช้สายสัญญาณ 2 เส้น คือ SCL และ SDA สำหรับติดกับอุปกร์แบบ 2 ทิศทาง โดยที่ขาสัญญาณทั้ง 2 จะต้องต่อกับตัวต้านทานแบบ Pull up 2 - 10K เนื่องจากเอาต์พุตมีลักษณะเป็นแบบ Open Darin หรือเป็นแบบ Open Collector เพื่อให้ขาสัญญาณเอาต์พุตสามารถเชื่อมต่อกันได้หลายตัว ซึ่งลักษณะการเชื่อมต่ออุปกรณ์บนระบบบัส I2C แสดงได้ดังรูปที่ 1.1



รูปที่ 1.1 แสดงลักษณะการการเชื่อมต่ออุปกรณ์แบบ I2C Bus
(ที่มา : http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm)

1.2 การเขียน-อ่านข้อมูลกับอุปกรณ์แบบ I2C Bus

ลักษณะการเขียน-อ่านข้อมูลกับอุปกรณ์แบบ I2C Bus สามารถแสดงรูปแบบข้อมูลของสัญญาณการอ่าน-เขียนได้ดังรูปที่ 1.2



รูปที่ 1.2 แสดงลักษณะรูปแบบของสัญญาณการเขียน-อ่านข้อมูลบน I2C Bus
(ที่มา : http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm)

	ใบเนื้อหา	หน้าที่ 2
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

การรับ-ส่งข้อมูลในรูปแบบ I2C Bus อุปกรณ์มาสเตอร์หรือ MCU จะมีขั้นตอนและขบวนการติดต่อสื่อสารอุปกรณ์อื่น ๆ บน I2C Bus ดังนี้

1. ส่งสถานะเริ่มต้น (START Conditions) เพื่อแสดงการขอใช้บัส
2. ตามด้วยรหัสควบคุม (Control Byte) ซึ่งประกอบด้วยรหัสประจำตัวอุปกรณ์ Device ID , Device Address และ Mode ในการเขียนหรืออ่านข้อมูลดังรูปที่ 1.2
3. เมื่ออุปกรณ์รับทราบว่า MCU ต้องการจะติดต่อด้วยก็ต้องส่งสถานะรับรู้ (Acknowledge) หรือแจ้งให้ MCU รับรู้ว่าข้อมูลที่ได้ส่งมามีความถูกต้อง และให้เริ่มขบวนการต่อไป
4. และเมื่อสิ้นสุดการส่งข้อมูล MCU จะต้องส่งสถานะสิ้นสุด (STOP Conditions) เพื่อบอกกับอุปกรณ์ว่า สิ้นสุดการใช้งานระบบบัส

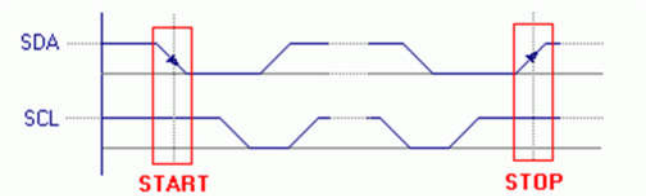
โดยการเขียน-อ่านข้อมูลกับอุปกรณ์แบบ I2C Bus ด้วย MCU จะมีการกำหนดสถานะของ I2C Bus ในรูปแบบต่าง ๆ ดังนี้

1. สถานะบัสว่าง คือเมื่อบัสไม่ได้ถูกใช้งาน ทั้งขา SCL และ SDA จะเป็นต้องถูกกำหนดให้เป็นลอจิก ‘1’ ทั้งคู่ด้วยอุปกรณ์มาสเตอร์ หรือ MCU
2. การกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ I2C Bus (START and STOP Conditions)

ลักษณะการกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ I2C Bus

2.1 เมื่อต้องการส่งข้อมูล MCU จะต้องส่งสถานะเริ่มต้น (START Conditions) คือให้ขา SDA เปลี่ยนจากลอจิก ‘1’ มาเป็นลอจิก ‘0’ ในขณะที่ขา SCL มีค่าเป็นลอจิก ‘1’

2.2 เมื่อสิ้นสุดการใช้งานบัส MCU จะต้องส่งสถานะสิ้นสุด (STOP Conditions) คือให้ขา SDA เปลี่ยนจากลอจิก ‘0’ มาเป็นลอจิก ‘1’ ในขณะที่ขา SCL มีค่าเป็นลอจิก ‘1’



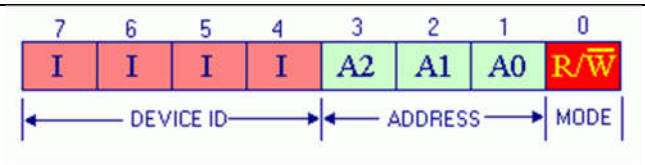
รูปที่ 1.3 แสดงรูปสัญญาณการกำหนดสถานะ START and STOP Conditions ของ I2C Bus
(ที่มา : http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm)

3. การส่งสัญญาณรหัสควบคุมของ I2C Bus (Control Byte)

การส่งสัญญาณรหัสควบคุมของ I2C Bus ประกอบด้วยรหัสประจำตัวของอุปกรณ์ (Device ID) จำนวน 7 บิต ที่ประกอบด้วยข้อมูลตำแหน่งบิตที่ 1-7 และบิต 0 จะเป็นบิตที่ทำหน้าที่ควบคุมการเขียนอ่านข้อมูลบนอุปกรณ์ที่ต่ออยู่บนระบบ I2C Bus ดังรูปที่ 1.4

	ใบเนื้อหา	หน้าที่ 3
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI



รูปที่ 1.4 แสดงรายละเอียดของสัญญาณรหัสควบคุมของ I2C Bus

(ที่มา : http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm)

จากรูปที่ 1.4 จะมีรายละเอียดการกำหนดข้อมูลบิตต่าง ๆ ดังนี้

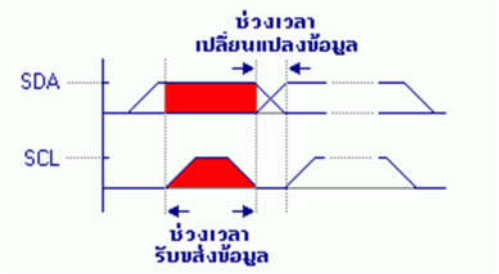
3.1 รหัสประจำตัวของอุปกรณ์ ประกอบด้วยรหัสประจำตัวจากผู้ผลิต Product ID จำนวน 4 บิต (ตำแหน่งบิต 4 - 7) ที่เปลี่ยนแปลงแก้ไขไม่ได้ และ Device Address จำนวน 3 บิต (ตำแหน่งบิต 1 - 3) ซึ่งผู้ใช้สามารถกำหนดเองได้ รวมแล้วเป็นรหัสจำนวน 7 บิต ที่ใช้ระบุตัวอุปกรณ์ที่ต่ออยู่บนบัส I2C และจะมีค่าซ้ำกันไม่ได้

3.2 บิตควบคุมการเขียนอ่าน (Mode) คือตำแหน่งบิต 0 เมื่อ MCU ต้องการเขียนข้อมูลไปยังอุปกรณ์ก็กำหนดให้บิตนี้มีค่าเป็นลอจิก ‘0’ และเมื่อต้องการอ่านข้อมูลจากอุปกรณ์ก็กำหนดให้บิตนี้มีค่าเป็นลอจิก ‘1’

4. ช่วงเวลารับส่งบิตข้อมูลของ I2C Bus ซึ่งจะมี 2 สภาวะดังนี้

4.1 สะภาวะการรับ-ส่งข้อมูล จะกระทำในขณะที่ขา SCL เป็นสัญญาณลอจิก ‘1’

4.2 สะภาวะการเปลี่ยนแปลงข้อมูล จะกระทำในขณะที่ขา SCL เป็นสัญญาณลอจิก ‘0’



รูปที่ 1.5 แสดงรูปแบบของสัญญาณการรับส่งบิตข้อมูลบน I2C Bus

(ที่มา : http://www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm)

2. รูปแบบข้อมูลของการสื่อสารแบบ SPI

SPI ย่อมาจาก Serial Peripheral Interface คือรูปของแบบการสื่อสารข้อมูลแบบอนุกรมแบบซิงโครนัสรูปแบบหนึ่ง ถูกพัฒนาขึ้นมาโดยโมโตโรล่าเพื่อใช้ในการการสื่อสารระยะใกล้ โดยเฉพาะในระบบสมองกลฝังตัว การสื่อสารอนุกรมแบบ SPI จะอาศัยสัญญาณนาฬิกาเป็นตัวกำหนดจังหวะการรับส่งข้อมูล สามารถส่งข้อมูลไปยังปลายทางและรับข้อมูลจากปลายทางกลับมาในครั้งเดียวกัน (Full Duplex) การสื่อสารอนุกรมแบบ SPI จะแบ่งอุปกรณ์ออกเป็น 2 ฝั่ง คือ Master เป็นตัวควบคุมการรับส่งข้อมูล และ Slave เป็นอุปกรณ์ที่รอรับคำสั่งจาก Master ในบัสการสื่อสารแบบอนุกรมแบบ SPI สามารถมี Slave ได้มากกว่า 1 ตัว

	ใบเนื้อหา	หน้าที่ 4
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

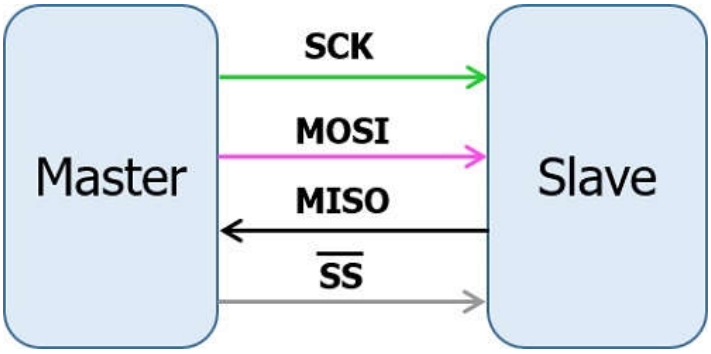
2.1 ขาสัญญาณและลักษณะการเชื่อมต่ออุปกรณ์ด้วยระบบบัส SPI

การสื่อสารอนุกรมแบบ SPI ใช้สายสัญญาณทั้งหมดจำนวน 4 เส้น มีรายละเอียดดังนี้

- SCK (Clock Data) ใช้สำหรับส่งสัญญาณนาฬิกาจาก Master ไปยัง Slave Shared
- MISO (Master In Slave Out) ใช้สำหรับรับข้อมูลจาก Slave Shared
- MOSI (Master Out Slave In) ใช้สำหรับส่งข้อมูลจาก Master ไปยัง Slave Shared
- SS/CS (Slave Select/Chip Select) ใช้สำหรับเลือก Slave ที่ต้องการใช้งาน Not Shared

สำหรับอุปกรณ์ที่ใช้การสื่อสารอนุกรมแบบ SPI บางชนิดอาจมีชื่อเรียกแตกต่างกันไป ตัวอย่างเช่น

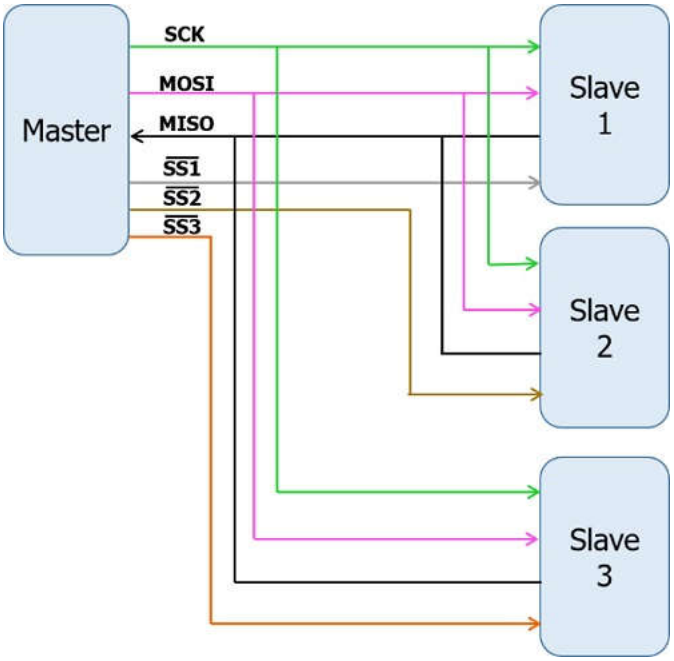
- Serial Clock
  * SCLK : SCK
- Master Output Slave Input
  * SIMO, MTSR
  * SDI, DI, DIN, SI - on Slave devices
  * SDO, DO, DOUT,SO - on Master devices
- Master Input Slave Output
  * SOMI, MRST
  * SDO, DO, DOUT, SO - on Slave devices
  * SDI, DI, DIN, SI - on Master devices
- Slave Select
  * SS, $\overline{SS}$, SSEL, CS, $\overline{CS}$, CE, nSS, /SS, SS#



รูปที่ 1.6 แสดงการเชื่อมต่อระบบด้วยการสื่อสารอนุกรมแบบ SPI แบบอุปกรณ์ Slave 1 ตัว
(ที่มา : https://www.thaieasyelec.com/article-wiki/embedded-electronics-application/09-espino32-spi.html)

	ใบเนื้อหา	หน้าที่ 5
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI



รูปที่ 1.7 แสดงการเชื่อมต่อระบบด้วยการสื่อสารอนุกรมแบบ SPI แบบอุปกรณ์ Slave หลายตัว

(ที่มา : https://www.thaieasyelec.com/article-wiki/embedded-electronics-application/09-espino32-spi.html)

จากรูปที่ 1.6 และรูปที่ 1.7 การเชื่อมต่อระบบด้วยการสื่อสารอนุกรมแบบ SPI แบบอุปกรณ์ Slave 1 ตัว และ การเชื่อมต่อระบบด้วยการสื่อสารอนุกรมแบบ SPI แบบอุปกรณ์ Slave หลายตัว จะประกอบไปด้วย 2 ส่วนคือ ส่วนอุปกรณ์หลัก (Master) และอุปกรณ์ย่อย (Slave) โดยระบบการสื่อสารอนุกรมแบบ SPI สามารถต่ออุปกรณ์ Slave ได้หลายอุปกรณ์ ภายในการสื่อสารอนุกรมแบบ SPI จะใช้สายสัญญาณ SCK MOSI และ MISO ร่วมกันแต่จะไม่ใช้สายสัญญาณ $\overline{SS}$ ร่วมกัน เวลาที่ Master ต้องการติดต่อสื่อสารกับ Slave จะส่งสัญญาณลอจิก LOW ไปยังอุปกรณ์ Slave ที่ต้องการสื่อสารเพื่อบ่งบอกว่าต้องการสื่อสารกับอุปกรณ์ Slave ตัวนั้น ทำให้สามารถเลือกสื่อสารกับอุปกรณ์ Slave ได้ถูกต้อง

2.2 รูปแบบสัญญาณในระบบบัส SPI

รูปแบบสัญญาณ SPI มี 4 รูปแบบ แตกต่างกันที่ขอบสัญญาณนาฬิกา (Clock Polarity) และเฟส (Phase) ดังนี้

2.1 เมื่อ CPHA=0 และ CPOL=0 สัญญาณนาฬิกา (Clock) ในสถานะปกติจะเป็น Low และจะรับ-ส่งข้อมูลที่ขอบขาขึ้นของสัญญาณนาฬิกา (Rising Edge Clock)

2.2 เมื่อ CPHA=0 และ CPOL=1 สัญญาณนาฬิกา (Clock) ในสถานะปกติจะเป็น High และจะรับ-ส่งข้อมูลที่ขอบขาลงของสัญญาณนาฬิกา (Falling Edge Clock)

2.3 เมื่อ CPHA=1 และ CPOL=0 สัญญาณนาฬิกา (Clock) ในสถานะปกติจะเป็น Low และจะรับ-ส่งข้อมูลที่ขอบขาลงของสัญญาณนาฬิกา (Falling Edge Clock)

	ใบเนื้อหา	หน้าที่ 6
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

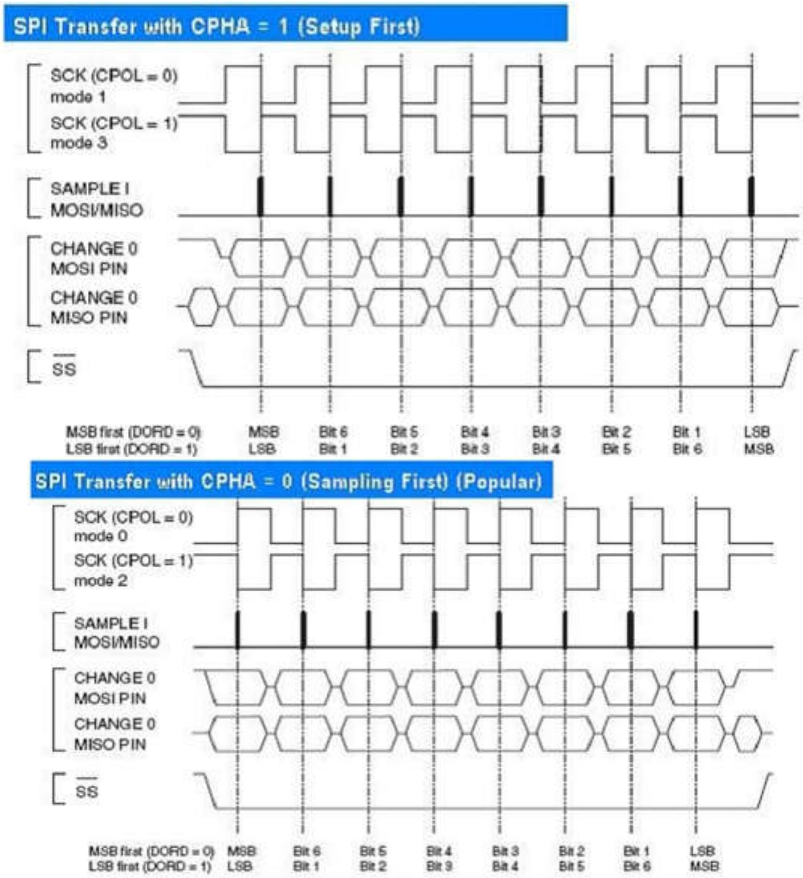
2.4 เมื่อ CPHA=1 และ CPOL=1 สัญญาณนาฬิกา (Clock) ในสถานะปกติจะเป็น High และจะรับ-ส่งข้อมูลที่ขอบขาขึ้นของสัญญาณนาฬิกา (Rising Edge Clock)

ดังนั้น Mode การรับส่งข้อมูลของระบบ SPI มี 4 โหมด ซึ่งสรุปการทำงานได้ดังรูปที่ 1.8

Mode	Clock Polarity (CPOL)	Clock Phase (CPHA)	Output Edge	Data Capture
SPI_MODE0	0	0	Falling	Rising
SPI_MODE1	0	1	Rising	Falling
SPI_MODE2	1	0	Rising	Falling
SPI_MODE3	1	1	Falling	Rising

รูปที่ 1.8 แสดงโหมดของการสื่อสารอนุกรมแบบ SPI

(ที่มา : https://www.thaieasyelec.com/article-wiki/embedded-electronics-application/09-espino32-spi.html)



รูปที่ 1.9 แสดงตัวอย่างรูปแบบของสัญญาณการสื่อสารอนุกรมแบบ SPI ในโหมด CPHA = 0 และ 1

(ที่มา : https://www.thaieasyelec.com/article-wiki/embedded-electronics-application/09-espino32-spi.html)

	ใบเนื้อหา	หน้าที่ 7
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

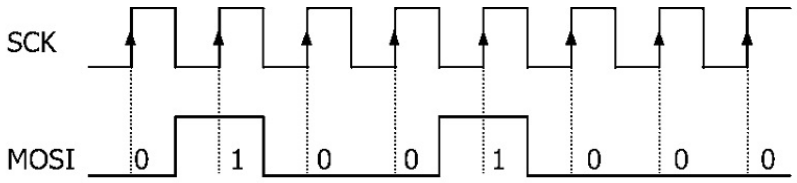
ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

2.3 การใช้งานการสื่อสารอนุกรมแบบ SPI

สำหรับการใช้งานการสื่อสารอนุกรมแบบ SPI จะไม่มีรูปแบบการสื่อสารที่เป็นมาตรฐาน จำเป็นต้องใช้รูปแบบของข้อมูลตามเอกสารของอุปกรณ์ที่ต้องการจะใช้งานร่วมกัน

ตัวอย่างการส่งข้อมูลแบบ SPI

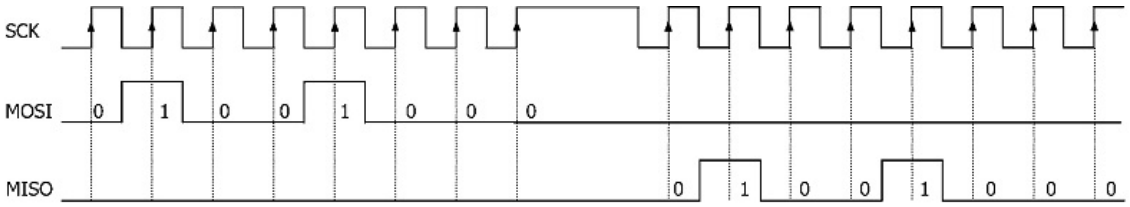
การส่งข้อมูลแบบ SPI จำเป็นต้องมีสัญญาณนาฬิกา (SCK) จากไมโครคอนโทรลเลอร์ และสายสัญญาณข้อมูลในการสื่อสาร ยกตัวอย่าง เช่น ให้ไมโครคอนโทรลเลอร์ส่งอักษร ‘H’ (B01001000) ออกไปผ่านช่องทางการสื่อสารอนุกรมแบบ SPI จะต้องใช้สายสัญญาณนาฬิกาที่คอยควบคุมจังหวะการรับส่งข้อมูล (ตัวอย่างนี้ใช้ SPI Mode 0) จะมีสัญญาณที่สายต่าง ๆ ดังรูป ตัวอย่างการส่งอักษร ‘H’ แบบ SPI



รูปที่ 1.10 ตัวอย่างสัญญาณการส่งอักษร ‘H’ แบบ SPI

ตัวอย่างการส่ง-รับข้อมูลแบบ SPI

การส่ง-รับข้อมูลแบบ SPI จะยังมีลักษณะเดียวกันกับการส่งข้อมูลแบบ SPI จะมีเอาต์พุตตอบกลับมาในสัญญาณนาฬิกาชุดถัดไป ยกตัวอย่าง เช่น ให้ไมโครคอนโทรลเลอร์ส่งอักษร ‘H’ (B01001000) ออกไปผ่านช่องทางการสื่อสารอนุกรมแบบ SPI และรับข้อมูลตัวอักษร ‘H’ กลับมา จะมีสัญญาณที่สายต่าง ๆ ดังรูป ตัวอย่างการส่งอักษร ‘H’ และรับข้อมูลอักษร ‘H’ ด้วยการสื่อสารอนุกรมแบบ SPI



รูปที่ 1.11 ตัวอย่างการส่งอักษร ‘H’ และรับข้อมูลอักษร ‘H’ ด้วยการสื่อสารอนุกรมแบบ SPI

	ใบเนื้อหา	หน้าที่ 8
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

3. รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต I2C ของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ AT89C51ED2 ที่ได้นำมาให้นักศึกษาใช้งานจะไม่มีโมดูลพิเศษ I2C ภายใน ถ้าต้องการติดต่อสื่อสารอุปกรณ์ภายนอกด้วยระบบบัส I2C จะต้องเขียนโปรแกรมให้ขาสัญญาณพอร์ตกำเนิดสัญญาณในรูปแบบ I2C ขึ้นมาแทน เรียกว่าการทำ Software I2C ส่วนไมโครคอนโทรลเลอร์ PIC16F887 และ ATMEGA32 จะมีโมดูลพิเศษ I2C อยู่ภายใน ซึ่งมีรีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต I2C ของไมโครคอนโทรลเลอร์ทั้งสองดังนี้

3.1 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต I2C ของ PIC16F887

ไมโครคอนโทรลเลอร์ PIC16F887 มีพอร์ตสำหรับการสื่อสารข้อมูล I2C จำนวน 1 พอร์ตที่ขา RC3 (SCL) และ RC4 (SDA) ซึ่งมีรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดการทำงานของพอร์ต I2C ดังนี้

SSPSTAT คือรีจิสเตอร์ที่ใช้แสดงสถานะและกำหนดลักษณะการทำงานของพอร์ต I2C โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.12

REGISTER 13-1: SSPSTAT: SSP STATUS REGISTER

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode
In I^2C Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high speed mode (400 kHz)

bit 6 **CKE:** SPI Clock Edge Select bit
CKP = 0:
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK
CKP = 1:
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit (I^2C mode only)
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

รูปที่ 1.12 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPATAT

	ใบเนื้อหา	หน้าที่ 9
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

bit 4 **P:** Stop bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a Stop bit has been detected last (this bit is '0' on Reset)
0 = Stop bit was not detected last

bit 3 **S:** Start bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a Start bit has been detected last (this bit is '0' on Reset)
0 = Start bit was not detected last

bit 2 **R/$\overline{W}$:** Read/Write bit information (I^2C mode only)
This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit, or not $\overline{ACK}$ bit.
In I^2C Slave mode:
1 = Read
0 = Write
In I^2C Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress
OR-ing this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSSP is in Idle mode.

bit 1 **UA:** Update Address bit (10-bit I^2C mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit
Receive (SPI and I^2C modes):
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
Transmit (I^2C mode only):
1 = Data transmit in progress (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is empty

รูปที่ 1.12 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPATAT (ต่อ)

SSPCON คือรีจิสเตอร์ที่ใช้ควบคุมการทำงานของพอร์ต I2C โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.13

REGISTER 13-2: SSPCON: SSP CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **WCOL:** Write Collision Detect bit
Master mode:
1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started
0 = No collision
Slave mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

รูปที่ 1.13 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPCON

	ใบเนื้อหา	หน้าที่ 10
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

bit 6 **SSPOV:** Receive Overflow Indicator bit

In SPI mode:

1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register (must be cleared in software).

0 = No overflow

In I^2C mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a "don't care" in Transmit mode (must be cleared in software).

0 = No overflow

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

In both modes, when enabled, these pins must be properly configured as input or output

In SPI mode:

1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

In I^2C mode:

1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit

In SPI mode:

1 = Idle state for clock is a high level

0 = Idle state for clock is a low level

In I^2C Slave mode:

SCK release control

1 = Enable clock

0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In I^2C Master mode:

Unused in this mode

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits

0000 = SPI Master mode, clock = Fosc/4
0001 = SPI Master mode, clock = Fosc/16
0010 = SPI Master mode, clock = Fosc/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0110 = I^2C Slave mode, 7-bit address
0111 = I^2C Slave mode, 10-bit address
1000 = I^2C Master mode, clock = Fosc / (4 * (SSPADD+1))
1001 = Load Mask function
1010 = Reserved
1011 = I^2C firmware controlled Master mode (Slave idle)
1100 = Reserved
1101 = Reserved
1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

รูปที่ 1.13 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPCON (ต่อ)

SSPCON2 คือรีจิสเตอร์ที่ใช้ควบคุมการทำงานของพอร์ต I2C โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.14

SSPBUF คือรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลที่เกิดจากการรับส่งข้อมูลที่พอร์ต I2C

SSPMSK คือรีจิสเตอร์ที่ใช้สำหรับการเปรียบเทียบแอดเดรสของการสื่อสารด้วยระบบบัส I2C โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.15

SSPADD คือรีจิสเตอร์ที่ใช้สำหรับการคำนวณหาค่าความเร็วในการสื่อสารด้วยระบบบัส I2C เมื่อกำหนดบิต SSPM<3:0> = 1000 ของรีจิสเตอร์ SSPCON

	ใบเนื้อหา	หน้าที่ 11
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

REGISTER 13-3: SSPCON2: SSP CONTROL REGISTER 2

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **GCEN:** General Call Enable bit (in I^2C Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (in I^2C Master mode only)
In Master Transmit mode:
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (in I^2C Master mode only)
In Master Receive mode:
Value transmitted when the user initiates an Acknowledge sequence at the end of a receive
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (in I^2C Master mode only)
In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

bit 3 **RCEN:** Receive Enable bit (in I^2C Master mode only)
1 = Enables Receive mode for I^2C
0 = Receive idle

bit 2 **PEN:** Stop Condition Enable bit (in I^2C Master mode only)
SCK Release Control:
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enabled bit (in I^2C Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enabled bit (in I^2C Master mode only)
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle

Note 1: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I^2C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

รูปที่ 1.14 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPCON2

	ใบเนื้อหา	หน้าที่ 12
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

REGISTER 13-4: SSPMSK: SSP MASK REGISTER(1)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0(2)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-1 **MSK<7:1>:** Mask bits
1 = The received address bit n is compared to SSPADD<n> to detect I²C address match
0 = The received address bit n is not used to detect I²C address match

bit 0 **MSK<0>:** Mask bit for I²C Slave mode, 10-bit Address(2)
I²C Slave mode, 10-bit Address (SSPM<3:0> = 0111):
1 = The received address bit 0 is compared to SSPADD<0> to detect I²C address match
0 = The received address bit 0 is not used to detect I²C address match

Note 1: When SSPCON bits SSPM<3:0> = 1001, any reads or writes to the SSPADD SFR address are accessed through the SSPMSK register.

2: In all other SSP modes, this bit has no effect.

รูปที่ 1.15 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPMSK

3.2 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต I2C ของ ATMEGA32

ไมโครคอนโทรลเลอร์ ATMEGA32 มีพอร์ตสำหรับการสื่อสารข้อมูล I2C จำนวน 1 พอร์ตที่ขา PC0 (SCL) และ PC1 (SDA) ซึ่งมีการความเร็วในการติดต่อสื่อสารตามสมการในรูปที่ 1.16 และมีรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดการทำงานของพอร์ต I2C ดังนี้

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

รูปที่ 1.16 แสดงสมการคำนวณหาค่าความเร็วของการสื่อสารด้วยระบบบัส I2C

TWBR คือรีจิสเตอร์ที่ใช้ในการกำหนดค่าเร็วในการสื่อสารด้วยระบบบัส I2C ดังรูปที่ 1.16

TWCR คือรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของการสื่อสารด้วยระบบบัส I2C โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.17

	ใบเนื้อหา	หน้าที่ 13
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TWINT: TWI Interrupt Flag**

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched.

The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI, so all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR), and TWI Data Register (TWDR) must be complete before clearing this flag.

- **Bit 6 – TWEA: TWI Enable Acknowledge Bit**

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met:

1. The device's own slave address has been received.
2. A general call has been received, while the TWGCE bit in the TWAR is set.
3. A data byte has been received in Master Receiver or Slave Receiver mode.

By writing the TWEA bit to zero, the device can be virtually disconnected from the Two-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

- **Bit 5 – TWSTA: TWI START Condition Bit**

The application writes the TWSTA bit to one when it desires to become a master on the Two-wire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO: TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the Two-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed slave mode and releases the SCL and SDA lines to a high impedance state.

รูปที่ 1.17 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ TWCR

	ใบเนื้อหา	หน้าที่ 14
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

- **Bit 3 – TWWC: TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN: TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit and will always read as zero.

- **Bit 0 – TWIE: TWI Interrupt Enable**

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

รูปที่ 1.17 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ TWCR (ต่อ)

TWSR คือรีจิสเตอร์ที่ใช้ในการแสดงสถานะการทำงานของการสื่อสารด้วยระบบบัส I2C และกำหนดปรีสเกลเลอร์ของความเร็วในการสื่อสาร โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.18

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- **Bits 7..3 – TWS: TWI Status**

These five bits reflect the status of the TWI logic and the Two-wire Serial Bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

รูปที่ 1.18 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ TWSR

- **Bits 1..0 – TWPS: TWI Prescaler Bits**

These bits can be read and written, and control the bit rate prescaler.

Table 73. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see “Bit Rate Generator Unit” on page 175. The value of TWPS1..0 is used in the equation.

รูปที่ 1.18 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ TWSR (ต่อ)

TWDR คือรีจิสเตอร์ที่ใช้ในการเก็บค่าข้อมูลของการรับส่งด้วยระบบบัส I2C

TWAR คือรีจิสเตอร์ที่ใช้ในการกำหนดตำแหน่งแอดเดรสของอุปกรณ์ ในกรณีที่ใช้ไมโครคอนโทรลเลอร์ ATMEGA32 ทำหน้าที่เป็นอุปกรณ์ตัวลูก (Slave) ในการสื่อสารด้วยระบบบัส I2C

4. รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต SPI ของไมโครคอนโทรลเลอร์

4.1 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต SPI ของ AT89C51ED2

ไมโครคอนโทรลเลอร์ AT89C51ED2 มีพอร์ตสำหรับการสื่อสารข้อมูล SPI จำนวน 1 พอร์ตที่ขา P1.1 (SS) , P1.5 (MISO) , P1.6 (SCK) และ P1.7 (MOSI) ซึ่งมีรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดการทำงานของพอร์ต SPI ดังนี้

SPCON คือรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของการสื่อสารด้วยระบบบัส SPI และกำหนดค่าความเร็วของการสื่อสาร โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.19

SPSTA คือรีจิสเตอร์ที่ใช้ในการแสดงสถานะ และควบคุมการทำงานของการสื่อสารด้วยระบบบัส SPI โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.20

SPDAT คือรีจิสเตอร์ที่ใช้ในการเก็บค่าข้อมูลของการรับส่งด้วยระบบบัส SPI

	ใบเนื้อหา	หน้าที่ 16
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

SPCON - Serial Peripheral Control Register (0C3H)

7	6	5	4	3	2	1	0
SPR2	SPEN	SSDIS	MSTR	CPOL	CPHA	SPR1	SPR0

Bit Number	Bit Mnemonic	Description
7	SPR2	**Serial Peripheral Rate 2** Bit with SPR1 and SPR0 define the clock rate.
6	SPEN	**Serial Peripheral Enable** Cleared to disable the SPI interface. Set to enable the SPI interface.
5	SSDIS	**$\overline{SS}$ Disable** Cleared to enable $\overline{SS}$ in both Master and Slave modes. Set to disable $\overline{SS}$ in both Master and Slave modes. In Slave mode, this bit has no effect if CPHA ='0'. When SSDIS is set, no MODF interrupt request is generated.
4	MSTR	**Serial Peripheral Master** Cleared to configure the SPI as a Slave. Set to configure the SPI as a Master.
3	CPOL	**Clock Polarity** Cleared to have the SCK set to '0' in idle state. Set to have the SCK set to '1' in idle low.
2	CPHA	**Clock Phase** Cleared to have the data sampled when the SCK leaves the idle state (see CPOL). Set to have the data sampled when the SCK returns to idle state (see CPOL).

Bit Number	Bit Mnemonic	Description			
		SPR2	**SPR1**	**SPR0**	**Serial Peripheral Rate**
1	SPR1	0	0	0	$F_{CLK\ PERIPH}/2$
		0	0	1	$F_{CLK\ PERIPH}/4$
		0	1	0	$F_{CLK\ PERIPH}/8$
0	SPR0	0	1	1	$F_{CLK\ PERIPH}/16$
		1	0	0	$F_{CLK\ PERIPH}/32$
		1	0	1	$F_{CLK\ PERIPH}/64$
		1	1	0	$F_{CLK\ PERIPH}/128$
		1	1	1	Invalid

Reset Value = 0001 0100b

รูปที่ 1.19 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SPCON

	ใบเนื้อหา	หน้าที่ 17
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

SPSTA - Serial Peripheral Status and Control register (0C4H)

7	6	5	4	3	2	1	0
SPIF	WCOL	SSERR	MODF	-	-	-	-

Bit Number	Bit Mnemonic	Description
7	SPIF	**Serial Peripheral Data Transfer Flag** Cleared by hardware to indicate data transfer is in progress or has been approved by a clearing sequence. Set by hardware to indicate that the data transfer has been completed.
6	WCOL	**Write Collision Flag** Cleared by hardware to indicate that no collision has occurred or has been approved by a clearing sequence. Set by hardware to indicate that a collision has been detected.
5	SSERR	**Synchronous Serial Slave Error Flag** Set by hardware when $\overline{SS}$ is de-asserted before the end of a received data. Cleared by disabling the SPI (clearing SPEN bit in SPCON).
4	MODF	**Mode Fault** Cleared by hardware to indicate that the $\overline{SS}$ pin is at appropriate logic level, or has been approved by a clearing sequence. Set by hardware to indicate that the $\overline{SS}$ pin is at inappropriate logic level.
3	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit
2	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.

Bit Number	Bit Mnemonic	Description
1	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
0	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.

Reset Value = 00X0 XXXXb

รูปที่ 1.20 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SPSTA

	ใบเนื้อหา	หน้าที่ 18
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

4.2 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต SPI ของ PIC16F887

ไมโครคอนโทรลเลอร์ PIC16F887 มีพอร์ตสำหรับการสื่อสารข้อมูล SPI จำนวน 1 พอร์ตที่ขา RA5 ($\overline{SS}$) , RC3 (SCK) , RC4 (SDI) และ RC5 (SDO) ซึ่งมีรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดการทำงานของพอร์ต SPI ดังนี้

SSPSTAT คือรีจิสเตอร์ที่ใช้แสดงสถานะและกำหนดลักษณะการทำงานของพอร์ต SPI โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.21

REGISTER 13-1: SSPSTAT: SSP STATUS REGISTER

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode
In I^2C Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high speed mode (400 kHz)

bit 6 **CKE:** SPI Clock Edge Select bit
CKP = 0:
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK
CKP = 1:
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit (I^2C mode only)
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a Stop bit has been detected last (this bit is '0' on Reset)
0 = Stop bit was not detected last

bit 3 **S:** Start bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a Start bit has been detected last (this bit is '0' on Reset)
0 = Start bit was not detected last

bit 2 **R/$\overline{W}$:** Read/Write bit information (I^2C mode only)
This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit, or not ACK bit.
In I^2C Slave mode:
1 = Read
0 = Write
In I^2C Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress
OR-ing this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSSP is in Idle mode.

bit 1 **UA:** Update Address bit (10-bit I^2C mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit
Receive (SPI and I^2C modes):
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
Transmit (I^2C mode only):
1 = Data transmit in progress (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is empty

รูปที่ 1.21 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPATAT

	ใบเนื้อหา	หน้าที่ 19
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

SSPCON คือรีจิสเตอร์ที่ใช้ควบคุมการทำงานของพอร์ต SPI โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.22

REGISTER 13-2: SSPCON: SSP CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **WCOL:** Write Collision Detect bit
Master mode:
1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started
0 = No collision
Slave mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit
In SPI mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register (must be cleared in software).
0 = No overflow
In I^2C mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a "don't care" in Transmit mode (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** Synchronous Serial Port Enable bit
In both modes, when enabled, these pins must be properly configured as input or output
In SPI mode:
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins
In I^2C mode:
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level
In I^2C Slave mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)
In I^2C Master mode:
Unused in this mode

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = Fosc/4
0001 = SPI Master mode, clock = Fosc/16
0010 = SPI Master mode, clock = Fosc/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0110 = I^2C Slave mode, 7-bit address
0111 = I^2C Slave mode, 10-bit address
1000 = I^2C Master mode, clock = Fosc / (4 * (SSPADD+1))
1001 = Load Mask function
1010 = Reserved
1011 = I^2C firmware controlled Master mode (Slave idle)
1100 = Reserved
1101 = Reserved
1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

รูปที่ 1.22 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SSPCON

	ใบเนื้อหา	หน้าที่ 20
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

SSPBUF คือรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลที่เกิดจากการรับส่งข้อมูลที่พอร์ต SPI

4.3 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต SPI ของ ATMEGA32

ไมโครคอนโทรลเลอร์ ATMEGA32 มีพอร์ตสำหรับการสื่อสารข้อมูล SPI จำนวน 1 พอร์ตที่ขา PB4 ($\overline{SS}$) , PB5 (MOSI) , PB6 (MISO) และ PB7 (SCK) ซึ่งมีรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดการทำงานของพอร์ต SPI ดังนี้

SPCR คือรีจิสเตอร์ที่ใช้ควบคุมการทำงานของพอร์ต SPI โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.23

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the global interrupt enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If $\overline{SS}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 67 and Figure 68 for an example. The CPOL functionality is summarized below:

Table 56. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

รูปที่ 1.23 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SPCR

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 67 and Figure 68 for an example. The CPHA functionality is summarized below:

Table 57. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the following table:

Table 58. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

รูปที่ 1.23 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SPCR (ต่อ)

SPSR คือรีจิสเตอร์ที่ใช้แสดงสถานะการทำงานของพอร์ต SPI และเพิ่มความเร็วในการติดต่อสื่อสารด้วยพอร์ต SPI อีก 2 เท่า โดยมีรายละเอียดการใช้งานของรีจิสเตอร์ดังรูปที่ 1.24

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

รูปที่ 1.24 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SPSR

	ใบเนื้อหา	หน้าที่ 22
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

- **Bit 6 – WCOL: Write COLlision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the ATmega32 and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 58). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the ATmega32 is also used for program memory and EEPROM downloading or uploading. See page 270 for SPI Serial Programming and Verification.

รูปที่ 1.24 แสดงรายละเอียดการทำงานของบิตต่าง ๆ ภายในรีจิสเตอร์ SPSR (ต่อ)

SPDR คือรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลที่เกิดจากการรับส่งข้อมูลที่พอร์ต SPI

5. การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ I2C ของไมโครคอนโทรลเลอร์

5.1 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ I2C ของ AT89C51ED2

การเขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ I2C ของ AT89C51ED2 จะเป็นการเขียนฟังก์ชันที่เรียกว่า Software I2C ดังนี้

```
sbit scl = P2^0;
sbit sda = P2^1;

void i2c_delay(){  //ฟังก์ชันหน่วงเวลาเพื่อสร้างสัญญาณนาฬิกาและรับส่งข้อมูลด้วยระบบบัส I2C
   unsigned char i;
   for(i=15;i>0;i--) _nop_();
}

void i2c_clk(){  //ฟังก์ชันสร้างสัญญาณนาฬิกาของระบบบัส I2C
   i2c_delay();
   scl = 1;
   i2c_delay();
   scl = 0;
}
```

	ใบเนื้อหา	หน้าที่ 23
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

```
void i2c_start(){  //ฟังก์ชันสร้างสัญญาณ start สำหรับการเริ่มขบวนการสื่อสารด้วยระบบบัส I2C
    if(scl) scl = 0;
    sda = 1;
    scl = 1;
    i2c_delay();
    sda = 0;
    i2c_delay();
    scl = 0;
}

void i2c_stop(){  //ฟังก์ชันสร้างสัญญาณ stop สำหรับสิ้นสุดขบวนการสื่อสารด้วยระบบบัส I2C
    if(scl) scl = 0;
    sda = 0;
    i2c_delay();
    scl = 1;
    i2c_delay();
    sda = 1;
}

bit i2c_wrdata(unsigned char dat){  //ฟังก์ชันสำหรับเขียนข้อมูลขนาด 8 บิตบนระบบบัส I2C
    bit data_bit;
    unsigned char i;
    for (i=0;i<8;i++){
        data_bit = dat & 0x80;
        sda = data_bit;
        i2c_clk();
        dat = dat<<1;
    }
    sda = 1;
    i2c_delay();
    scl = 1;
    i2c_delay();
    data_bit = sda;
    scl = 0;
    i2c_delay();
    return (data_bit);
}
```

	ใบเนื้อหา	หน้าที่ 24
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

```
unsigned char i2c_rddata(){  //ฟังก์ชันสำหรับอ่านข้อมูลขนาด 8 บิตบนระบบบัส I2C
    bit rd_bit;
    unsigned char i,dat;
    dat = 0x00;
    for (i=0;i<8;i++){
        i2c_delay();
        scl = 1;
        i2c_delay();
        rd_bit = sda;
        dat = dat<<1;
        dat = dat | rd_bit;
        scl = 0;
    }
    sda = 1;
    i2c_delay();
    i2c_clk();
    scl = 1;
    return (dat);
}
//ฟังก์ชันสำหรับเขียนข้อมูลให้แก่อุปกรณ์ที่ต่ออยู่บนระบบบัส I2C
void i2cWriteByte(unsigned char control,unsigned char addr,unsigned char dat){
    bit err = 0;
    do{
        i2c_start();
        err = i2c_wrdata(control);
        if(err == 0){
                err = i2c_wrdata(addr);
                if(err == 0){
                        err = i2c_wrdata(dat);
                        i2c_stop();
                }
        }
    }while(err);
}
```

	ใบเนื้อหา	หน้าที่ 25
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

```
//ฟังก์ชันสำหรับอ่านข้อมูลจากอุปกรณ์ที่ต่ออยู่บนระบบบัส I2C
unsigned char i2cReadByte(unsigned char control,unsigned char addr){
    bit err = 0;
    unsigned char dat;
    do{
        i2c_start();
        err = i2c_wrdata(control);
        if(err == 0){
                err = i2c_wrdata(addr);
                if(err == 0){
                        i2c_start();
                        err = i2c_wrdata(control+1);
                        if(err == 0){
                                dat = i2c_rddata();
                                i2c_stop();
                        }
                }
        }
    }while(err);
    return (dat);
 }
//ฟังก์ชันสำหรับกำหนดสถานะเริ่มต้นของขา scl และ sda
void init_i2c(){
    scl = 1;
    sda = 1;
}
```

	ใบเนื้อหา	หน้าที่ 26
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

5.2 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ I2C ของ PIC16F887

การเขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ I2C ของ PIC16F887 จะเป็นการเขียนฟังก์ชันที่เรียกว่า Hardware I2C ดังนี้

```
//ฟังก์ชันสำหรับกำหนดค่าเริ่มต้นของการสื่อสารด้วยระบบบัส I2C และกำหนดความเร็วในการสื่อสาร
void I2C_Init(long i2c_clk_freq){
    SSPCON  = 0x28;                          // configure MSSP module to work in I2C mode
    SSPADD  = (_XTAL_FREQ/(4 * i2c_clk_freq)) - 1;  // set I2C clock frequency
    SSPSTAT = 0;
}
//ฟังก์ชันสร้างสัญญาณ start ของการสื่อสารด้วย I2C
void I2C_Start(){
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));  // wait for MSSP module to be free
    SEN = 1;                                       // initiate start condition
}
//ฟังก์ชันสร้างสัญญาณ restart ของการสื่อสารด้วย I2C
void I2C_Repeated_Start(){
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));  // wait for MSSP module to be free
    RSEN = 1;                                      // initiate repeated start condition
}
//ฟังก์ชันสร้างสัญญาณ stop ของการสื่อสารด้วย I2C
void I2C_Stop(){
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));  // wait for MSSP module to be free
    PEN = 1;                                       // initiate stop condition
}
//ฟังก์ชันสำหรับส่งข้อมูล 8 บิต บนระบบบัส I2C
void I2C_Write(unsigned char i2c_data){
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));  // wait for MSSP module to be free
    SSPBUF = i2c_data;                             // update buffer
}
```

	ใบเนื้อหา	หน้าที่ 27
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

```
//ฟังก์ชันสำหรับการอ่านข้อมูล 8 บิต บนระบบัส I2C
unsigned char I2C_Read(unsigned char ack){
    unsigned char _data;
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));  // wait for MSSP module to be free (not busy)
    RCEN = 1;
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));  // wait for MSSP module to be free (not busy)
    _data = SSPBUF;  // read data from buffer
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));  // wait for MSSP module to be free (not busy)
    // send acknowledge pulse ? (depends on ack, if 1 send, otherwise don't send)
    if(ack) ACKDT = 0;
    else ACKDT = 1;
    ACKEN = 1;
    return _data;  // return data read
}
//ฟังก์ชันสำหรับเขียนข้อมูลขนาด 1 byte ไปยังอุปกรณ์และแอดเดรสที่กำหนด
void i2cWriteByte(unsigned char ctl,unsigned char addr,unsigned char dat){
    I2C_Start();        // start I2C
    I2C_Write(ctl);     // RTC chip address
    I2C_Write(addr);    // send register address
    I2C_Write(dat);     // send register address
    I2C_Stop();         // stop I2C
}
//ฟังก์ชันสำหรับอ่านข้อมูลขนาด 1 byte จากอุปกรณ์และแอดเดรสที่กำหนด
unsigned char i2cReadByte(unsigned char ctl,unsigned char addr){
    unsigned char dat;
    I2C_Start();        // start I2C
    I2C_Write(ctl);     // RTC chip address
    I2C_Write(addr);    // send register address
    I2C_Repeated_Start(); // restart I2C
    I2C_Write(ctl+1);   // initialize data read
    dat = I2C_Read(0);  // read data from addr
    I2C_Stop();         // stop I2C
    return dat;
}
```

	ใบเนื้อหา	หน้าที่ 28
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 8
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

5.3 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ I2C ของ ATMEGA32

การเขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ I2C ของ ATMEGA32 จะเป็นการเขียนฟังก์ชันที่เรียกว่า Hardware I2C ดังนี้

```
//ฟังก์ชันกำหนดค่าเริ่มต้นของการสื่อสารด้วยระบบบัส I2C และกำหนดความเร็วในการสื่อสาร
void i2c_init(uint32_t freq){
    TWBR =(uint32_t)(((uint32_t)(F_CPU/freq) - 16) / 8);
}
//ฟังก์ชันสำหรับเขียนข้อมูลขนาด 1 byte ไปยังอุปกรณ์และแอดเดรสที่กำหนด
void i2cWriteByte(char address, char reg, char data){
    TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);  // send a start bit on i2c bus
    while(!(TWCR & (1<<TWINT)));             // wait for confirmation of transmit
    TWDR = address;                          // load address of i2c device
    TWCR = (1<<TWINT) | (1<<TWEN);           // transmit
    while(!(TWCR & (1<<TWINT)));             // wait for confirmation of transmit
    TWDR = reg;
    TWCR = (1<<TWINT) | (1<<TWEN);           // transmit
    while(!(TWCR & (1<<TWINT)));             // wait for confirmation of transmit
    TWDR = data;
    TWCR = (1<<TWINT) | (1<<TWEN);           // transmit
    while(!(TWCR & (1<<TWINT)));             // wait for confirmation of transmit
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);     // stop bit
}
//ฟังก์ชันสำหรับอ่านข้อมูลขนาด 1 byte จากอุปกรณ์และแอดเดรสที่กำหนด
unsigned char i2cReadByte(char address, char reg){
    char read_data = 0;
    TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);  // send a start bit on i2c bus
    while(!(TWCR & (1<<TWINT)));             // wait for confirmation of transmit
    TWDR = address;                          // load address of i2c device
    TWCR = (1<<TWINT) | (1<<TWEN);           // transmit
    while(!(TWCR & (1<<TWINT)));             // wait for confirmation of transmit
    TWDR = reg;                              // send register number to read from
```

```
        TWCR = (1<<TWINT) | (1<<TWEN);          // transmit
        while(!(TWCR & (1<<TWINT)));            // wait for confirmation of transmit

        TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);     // send repeated start bit
        while(!(TWCR & (1<<TWINT)));            // wait for confirmation of transmit
        TWDR = address+1;                       // transmit address of i2c device with readbit set
        TWCR = (1<<TWINT)|(1<<TWEA)|(1<<TWEN);      // clear transmit interupt flag
        while(!(TWCR & (1<<TWINT)));            // wait for confirmation of transmit
        TWCR = (1<<TWINT) | (1<<TWEN);          // transmit, nack (last byte request)
        while(!(TWCR & (1<<TWINT)));            // wait for confirmation of transmit
        read_data = TWDR;                       // and grab the target data
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);     // send a stop bit on i2c bus
        return read_data;
    }
```

6. การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ SPI ของไมโครคอนโทรลเลอร์

6.1 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ SPI ของ AT89C51ED2

เนื่องจากการทำงานของพอร์ต SPI แบบ Hardware ของ AT89C51ED2 มีปัญหาทำให้ไม่สามารถสั่งงานให้ทำงานได้ ดังนั้น การติดต่ออุปกรณ์ด้วยพอร์ต SPI ของ AT89C51ED2 จึงกระทำแบบ Software SPI ดังนี้

```
#define ss   P1_1
#define mosi P1_7
#define miso P1_5
#define sck  P1_6
//ฟังก์ชันกำหนดค่าเริ่มต้นของการติดต่ออุปกรณ์ด้วยพอร์ต SPI
 void spi_init(){    //Software SPI      CPOL = 0 , CPHA = 0
        ss = 0;
        mosi = 0;
        miso = 1;
        sck = 0;
 }
```

```
//ฟังก์ชันสำหรับการเขียนค่าข้อมูล และรับข้อมูลจากพอร์ต SPI
unsigned char spi_write(char dat){
    signed char i;
    unsigned char datO=0;
    for(i=7;i>=0;i--){
        datO <<= 1;
        if(miso) datO |= 1;
        if(dat & (1<<i)) mosi = 1;
        else mosi = 0;
        sck = 1;
        sck = 0;
    }
    return datO;
}
```

6.2 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ SPI ของ PIC16F887

การเขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ SPI ของ PIC16F887 จะเป็นการเขียนฟังก์ชันที่เรียกว่า Hardware SPI ดังนี้

```
#define SS_TRIS TRISC0
#define CS_HIGH() (RC0 = 1)
#define CS_LOW()  (RC0 = 0)
//ฟังก์ชันกำหนดค่าเริ่มต้นของการติดต่ออุปกรณ์ด้วยพอร์ต SPI
void SPIInit(){
    TRISCbits.TRISC3 = 0; // Setting Serial Clock as Output
    TRISCbits.TRISC4 = 1; // Master Input Slave Output (MISO) - SDI as input
    TRISCbits.TRISC5 = 0; // Master Output Slave Input (MOSI) - SDO as output
    SS_TRIS = 0;          // Slave Select (SS) as output
    CS_HIGH();            // SS = HIGH
    //Mater Mode, CPOL = 0, CPHA = 0 ,Freq SPI = XTAL/64
    SSPSTAT = 0x00;
    SSPCON = 0b00100010;
}
```

```
//ฟังก์ชันสำหรับการเขียนค่าข้อมูล และรับข้อมูลจากพอร์ต SPI
unsigned char SPIWrite(unsigned char data){
    SSPBUF = data;
    while(!SSPSTATbits.BF); // Wait for transmission complete
    return (SSPBUF);
}
```

6.3 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ SPI ของ ATMEGA32

การเขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ SPI ของ ATMEGA32 จะเป็นการเขียนฟังก์ชันที่เรียกว่า Hardware SPI ดังนี้

```
#define SPI_PORT  PORTB
#define SPI_DDR   DDRB
#define SCK_P     PB7
#define MISO_P    PB6
#define MOSI_P    PB5
#define SS_P      PB4
#define CS_HIGH()  (SPI_PORT |= (1<<SS_P))
#define CS_LOW()  (SPI_PORT &= ~(1<<SS_P))
//ฟังก์ชันกำหนดค่าเริ่มต้นของการติดต่ออุปกรณ์ด้วยพอร์ต SPI
void SPIInit(){
    SPI_DDR |= (1<<MOSI_P)|(1<<SCK_P)|(1<<SS_P);
    SPI_PORT |= (1<<SS_P);
    CS_HIGH();
    //Mater Mode, CPOL = 0, CPHA = 0 ,Freq SPI = XTAL/16 = 1MHz
    SPCR|=(1<<SPE)|(1<<MSTR)|(1<<SPR0);
}
//ฟังก์ชันสำหรับการเขียนค่าข้อมูล และรับข้อมูลจากพอร์ต SPI
uint8_t SPIWrite(uint8_t data){
    SPDR = data;
    while(!(SPSR & (1<<SPIF))); // Wait for transmission complete
    return (SPDR);
}
```

	แบบฝึกหัด	**หน้าที่ 1**
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	**หน่วยที่ 8**
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

<u>คำสั่ง</u> จงตอบคำถามต่อไปนี้ให้ถูกต้อง

1. ให้นักศึกษาอธิบายลักษณะการสื่อสารด้วยพอร์ต I2C

.....

.....

.....

2. ให้นักศึกษาอธิบายลักษณะการสื่อสารด้วยพอร์ต SPI

.....

.....

.....

3. ให้นักศึกษาอธิบายการใช้งานสัญญาณ ACK และ NACK บนระบบบัส I2C

.....

.....

.....

4. ให้บอกข้อแตกต่างของการสื่อสารด้วยพอร์ต I2C และ SPI

.....

.....

.....

5. ไมโครคอนโทรลเลอร์ AT89C51ED2 สามารถติดต่ออุปกรณ์ภายนอกด้วยพอร์ต I2C และ SPI ได้หรือไม่ อย่างไร

.....

.....

.....

6. ไมโครคอนโทรลเลอร์ PIC16F887 มีรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานด้วยพอร์ต I2C จำนวนกี่ตัว อะไรบ้าง

.....

.....

.....

7. ไมโครคอนโทรลเลอร์ PIC16F887 มีรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานด้วยพอร์ต SPI จำนวนกี่ตัว อะไรบ้าง

.....

.....

.....

	แบบฝึกหัด	**หน้าที่ 2**
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	**หน่วยที่ 8**
	ชื่อหน่วย การรับส่งข้อมูลอนุกรมแบบต่าง ๆ	

ชื่อเรื่อง การรับส่งข้อมูลอนุกรมในรูปแบบ I2C และ SPI

8. ไมโครคอนโทรลเลอร์ ATMEGA32 มีรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานด้วยพอร์ต I2C จำนวนกี่ตัว อะไรบ้าง

.....

.....

.....

9. ไมโครคอนโทรลเลอร์ ATMEGA32 มีรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานด้วยพอร์ต SPI จำนวนกี่ตัว อะไรบ้าง

.....

.....

.....

10. ไมโครคอนโทรลเลอร์ PIC16F887 มีพอร์ต I2C และ SPI มีความเกี่ยวข้องกันอย่างไร

.....

.....

.....