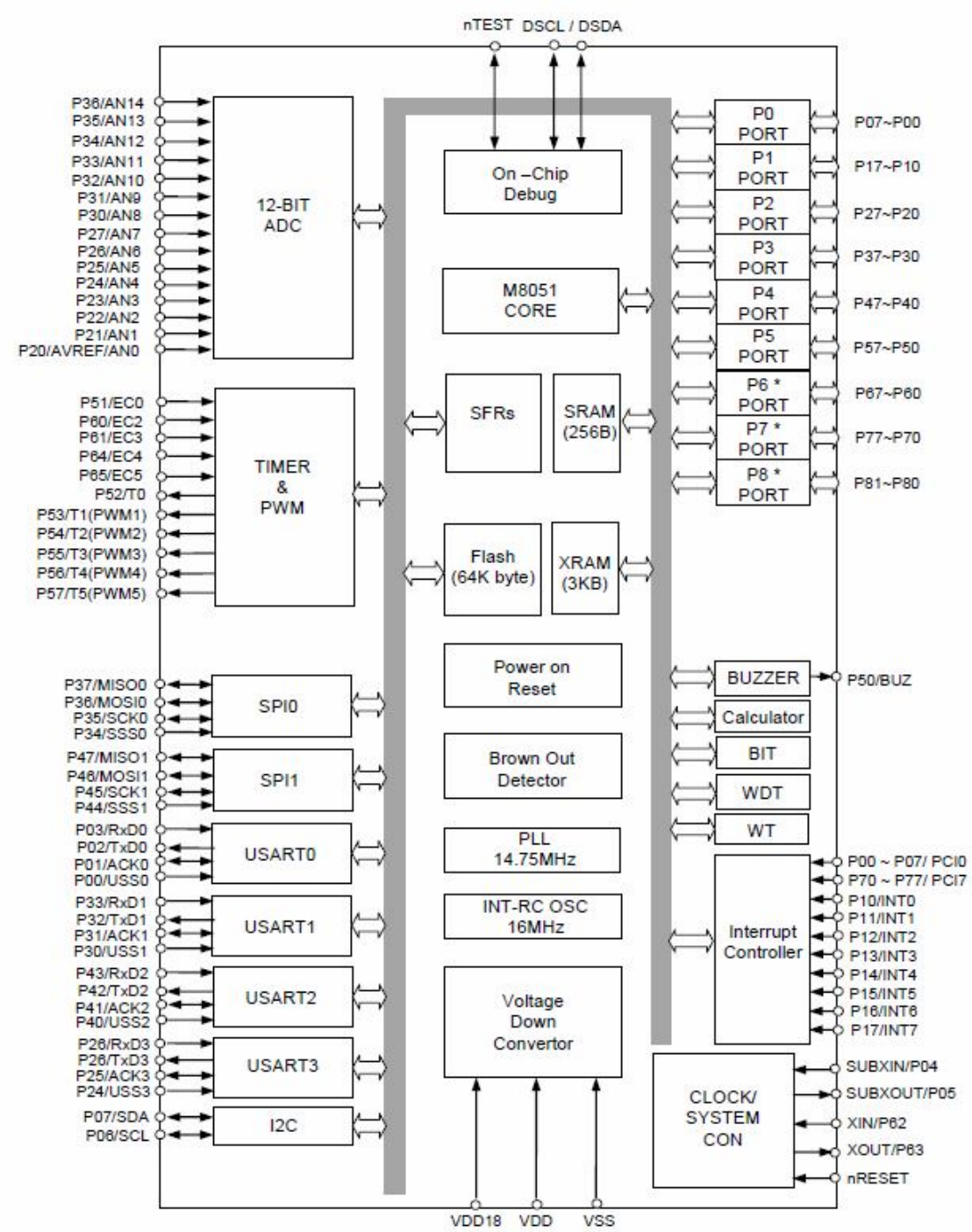
Z8051

Z8051 เป็นไมโครคอนโทรลเลอร์ชิพเดี่ยวแบบ CISC ขนาด 8บิต ซึ่งใช้สถาปัตยกรรมการประมวลผลของ MCS51 ตามแบบฉบับของ Intel เหมือนเดิม แต่ทาง Zilog ได้มีการปรับเปลี่ยนระบบการทำงานของ Peripheral I/O ต่างๆใหม่หมด ทั้ง I/O Port, UART และ Timer/Counter เรียกว่า ยกเครื่อง Peripheral I/O กันใหม่หมด แต่ยังใช้วิธีการ สำหรับใช้ควบคุมและเข้าถึง Peripheral I/O ต่างๆผ่านทาง SFR (Special Function Register) เหมือน Intel เช่นเดิม ดังนั้นรูปแบบการเขียนโปรแกรมสั่งงานจึงเหมือน MCS51 เดิมจาก Intel ทุกประการ โดยทาง Zilog ได้ทำการปรับปรุงพัฒนาขีดความสามารถด้านต่างๆของ Z8051 ให้มีขีดความสามารถสูงขึ้นกว่า MCS51 มาตรฐานในหลายๆด้าน ทั้งด้านความเร็วการประมวลผล มีการเพิ่มหน่วยคำนวณคณิตศาสตร์สำหรับคูณและหารเลขจำนวนเต็ม 32บิต และการจัดการด้านพลังงาน รวมทั้งได้บรรจุอุปกรณ์ Peripheral I/O พิเศษแบบต่างๆ เช่น GPIO Port, Pin Pull-Up, Pin Debounce, Pin Change Interrupt, USART, SPI, I2C, ADC, Timer/Counter/Capture/PWM ,Buzzer Control และความสามารถในการรองรับการ Interrupt ได้ดีกว่า MCS51 มาตรฐาน ทำให้การเขียนโปรแกรมควบคุมสั่งงานฮาร์ดแวร์ทำได้ง่ายมากขึ้น ลดความซับซ้อนในการพัฒนาโปรแกรมได้เป็นอย่างมาก

ซึ่งในปัจจุบันทาง Zilog Inc. ได้ผลิตชิพตระกูล Z8051 นี้ออกมาให้เลือกใช้จำนวน 5 เบอร์ โดยแต่ละเบอร์จะมีคุณสมบัติพื้นฐานที่เหมือนกัน แต่จะมีความแตกต่างกันในเรื่องของรูปร่างตัวถัง จำนวนขาสัญญาณ I/O ใช้งาน ขนาดของหน่วยความจำ และจำนวนทรัพยากรต่างๆที่บรรจุไว้ภายในตัว ซึ่งจะมีมากน้อยไม่เท่ากัน ทั้งนี้เพื่อให้ผู้ใช้สามารถเลือก MCU ไปออกแบบใช้งานให้มีความเหมาะสมกับงาน เช่น Z51F0410, Z51F0811, Z51F3220, Z51F3221 และ Z51F6412

ตัวอย่างคุณสมบัติของ Z8051 เบอร์ Z51F6412



โครงสร้างของ **MCU** ตระกูล **Z8051** เบอร์ **Z51F6412**

คุณสมบัติของ Z8051 เบอร์ Z51F6412

- เป็น MCUตระกูล Z8051 Run ความถี่สูงสุด 16 MHz(ประมวลผล 125nS/1 Machine Cycle) จาก Internal Oscillator ภายใน
- 64KByte Flash / 3KByte XRAM / 256 Byte IRAM
- ใช้สถาปัตยกรรมการประมวลผลแบบ MCS51 (2 Clock / 1 Machine Cycle)
- Internal Oscillator 16MHz(ผิดพลาดไม่เกิน +/-2%) ตั้งหาร 2,4,8,16 ได้จากโปรแกรม
- มี 66 Bit GPIO Port สามารถโปรแกรมเป็น Peripheral I/O แบบต่างๆได้ เช่น
    - 15 Channel 12Bit ADC
    - 4 Channel UART
    - 2 Channel SPI
    - 1 Channel I2C
    - 2 Channel 8 Bit Timer/Counter(T0,T1) สามารถใช้รวมกันเป็น 16 Bit 1 ช่องได้
    - 4 Channel 16 Bit Timer/Counter/PWM(T2,T3,T4,T5)
    - 8 Bit External Interrupt Trigger(INT0...INT7)
    - 16 Bit Pin Change Interrupt Trigger(P0,P7)
    - Internal Pin Pull-Up ในทุกๆ Pin เลือก Enable/Disable ได้อิสระทุก Pin
    - Internal Pin Debounce ในทุกๆ Pin เลือก Enable/Disable ได้อิสระทุก Pin
    - 1 Channel Buzzer Drive
    - รองรับการ Interrupt จากอุปกรณ์ต่างๆ 32 แหล่ง 32 Vector
    - มีวงจร Calculator สำหรับคำนวณแบบคูณและหารเลขจำนวนเต็มขนาด 32บิต
        - คูณเลขจำนวนเต็ม 16Bit x 16Bit โดยใช้เวลา 1 Cycle Clock
        - หารเลขจำนวนเต็ม 32Bit / 16Bit โดยใช้เวลา 32 Cycle Clock
- Watch Timer และ Watch Dog Timer
- Power-ON Reset ที่ 1.4V
- Programmable Brown-Out Detect(1.6V, 2.5V, 3.6V และ 4.2V)

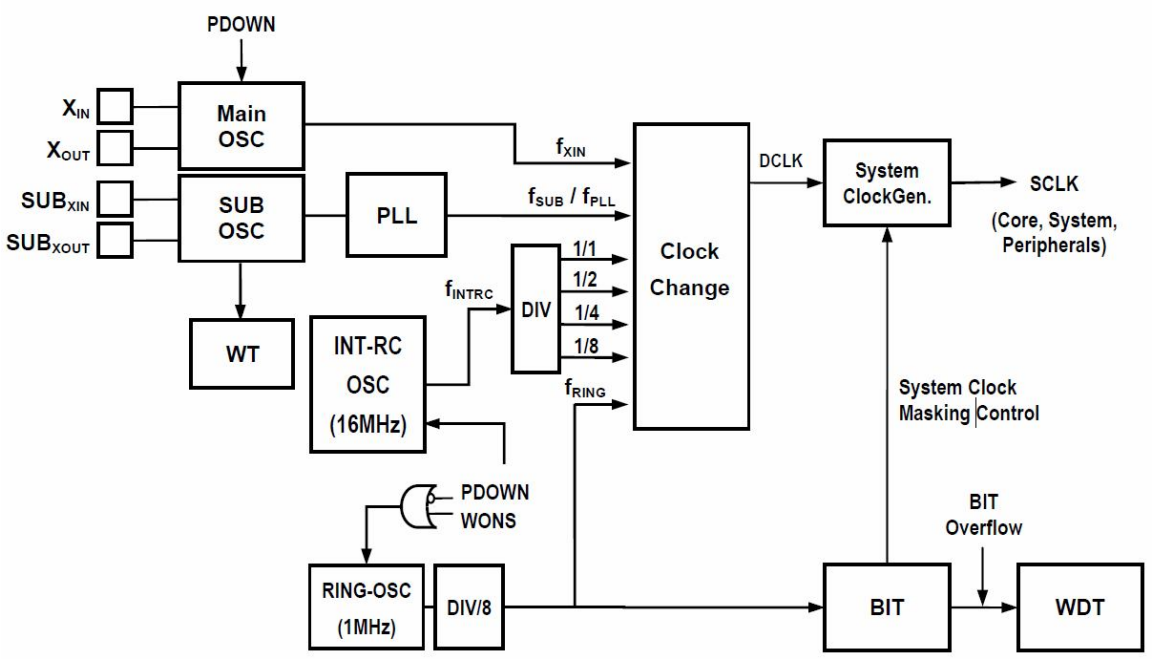
ความโดดเด่นของ Z8051 เบอร์ Z51F6412

- **Port I/O** ได้รับการพัฒนาปรับปรุงให้สามารถทำหน้าที่เป็นได้ทั้ง Input / Output แบบ GPIO ทั่วๆไป และทำหน้าที่เชื่อมต่อกับโมดูล Peripheral I/O ต่างๆที่บรรจุไว้ภายในตัว MCU โดยเมื่อใช้งานเป็น GPIO ทั่วๆไป Z8051 ก็ยังมีความโดดเด่นเป็นอย่างมาก คือ มีการบรรจุวงจรสนับสนุนต่างๆเพื่อให้ผู้ใช้งานได้เลือกใช้ตามความต้องการในทุกๆ Pin Port โดยอิสระ
    - มีส่วนควบคุมทิศทาง(Port Direction) สำหรับกำหนดหน้าที่การใช้งานว่าจะใช้เป็น Input หรือ Output ในทุกๆ Pin Port โดยอิสระ
    - สามารถกำหนดคุณสมบัติของ Port เป็นแบบ Output ปรกติ(Push-Pull) หรือแบบ Open-Drain ได้เมื่อทำหน้าที่เป็น Output ในทุกๆ Pin Port โดยอิสระ
    - สามารถ เปิด-ปิด วงจร Internal Pull-Up ของแต่ละ Pin Port ได้เมื่อใช้งานเป็น Input
    - สามารถ เปิด-ปิด วงจร Debounce ของแต่ละ Pin Port ได้เมื่อใช้งานเป็น Input เพื่อเชื่อมต่อกับอุปกรณ์ประเภทหน้าสัมผัสต่างๆ
- **ด้านการประมวลผล** สามารถประมวลผลได้อย่างรวดเร็ว โดยใช้สัญญาณนาฬิกาเพียง 2 Cycle ในการประมวลผล 1 Machine Cycle ซึ่ง Z8051 สามารถ Run ได้สูงสุดที่ความถี่ 16MHz จึงทำให้ Z8051 สามารถประมวลผลได้เร็วถึง 8MIPS หรือใช้เวลาในการประมวลผลคำสั่งเพียง 125nS ต่อ 1 Machine Cycle ที่ 16MHz
- **ด้านการคำนวณ** มีวงจรช่วยคำนวณทางคณิตศาสตร์ซึ่งเป็น Hardware สำหรับช่วยคำนวณเกี่ยวกับการคูณและหารเลขจำนวนเต็มขนาด 32บิต บรรจุไว้ภายใน ทำให้ลดความยุ่งยากในการเขียนโปรแกรมเกี่ยวกับการคำนวณและใช้เวลาประมวลผลรวดเร็วมากยิ่งขึ้น
    - คูณเลขจำนวนเต็ม 16บิต x 16บิต โดยใช้เวลาเพียง 1 Cycle
    - หารเลขจำนวนเต็ม 32บิต/ 16บิต โดยใช้เวลาเพียง 32 Cycle
- **ด้านพลังงาน** ส่วนประมวลผลได้รับการออกแบบให้ปฏิบัติงานที่แรงดันคงที่ +1.8V โดยมีวงจร Regulate บรรจุไว้ภายในชิพ ซึ่งภายนอกก็ยังคงป้อนแหล่งจ่าย 2.0V - 5.5V ให้กับชิพตามปรกติ โดยจะมีวงจรปรับระดับโลจิกระหว่าง 1.8V กับ +VDD ภายนอกบรรจุไว้ เพื่อให้หน่วยประมวลผลเชื่อมต่อกับอุปกรณ์ภายนอกโดยไม่เกิดปัญหา ลดปัญหา MCU หยุดทำงานอันเนื่องจากสาเหตุของแหล่งจ่ายกระเพื่อมเมื่ออุปกรณ์ Peripheral I/O และ Output ต่างๆ รอบนอกทำงานและเปลี่ยนแปลงค่าอย่างรวดเร็ว

- **ด้านของสัญญาณนาฬิกา Clock** สามารถเลือกใช้งานแหล่งกำเนิดสัญญาณนาฬิกาได้จากหลายแหล่งทั้งจาก แหล่งกำเนิดภายใน และ แหล่งกำเนิดภายนอก และ ยังสามารถสั่งเปลี่ยนแปลงสลับแหล่งกำเนิดสัญญาณนาฬิกาได้จากคำสั่งในโปรแกรมด้วย จึงทำให้ผู้ใช้สามารถที่จะเลือกรูปแบบการทำงานให้เหมาะสมกับการใช้งานมากที่สุด ทั้งเหตุผลจากเรื่องการประหยัดพลังงาน ความเร็วในการทำงาน ต้นทุน หรือ ความแม่นยำเที่ยงตรง
- **มี Timer Counter ให้ใช้มากถึง 6 ช่อง** โดยเป็น Timer/Counter ขนาด 8บิต 2 ช่อง และ 16 บิต อีก 4 ช่อง ซึ่ง Timer/Counter 8 บิต สามารถโปรแกรมให้ทำงานร่วมกันเป็น Timer/Counter ขนาด 16บิต 1 ช่องแทนได้ด้วย
- **มีวงจรสร้าง PWM ขนาด 16บิต จำนวน 4 ช่อง** และสามารถโปรแกรม Timer 8 บิต 2 ช่องร่วมกันให้เป็น 16บิตเพื่อสร้าง PWM ขนาด 10บิต เพิ่มอีก 1 ช่อง สามารถโปรแกรมสร้างสัญญาณ PWM ที่สามารถปรับค่า Duty Cycle มีความละเอียดสูงถึง 125nS ต่อ Step ที่ Period 8.192mS หรือ สร้างสัญญาณ PWM ที่ปรับค่า Duty Cycle ละเอียด 500nS ที่ Period 20mS สำหรับ Control RC Servo ได้มากถึง 4 ช่อง
- **มีวงจร UART จำนวน 4 ช่อง** มากพอสำหรับรองรับการสื่อสารกับอุปกรณ์รอบนอกต่างๆ
- **มีวงจร SPI จำนวน 2 ช่อง** สำหรับเชื่อมต่อกับอุปกรณ์สนันสนุนต่างๆที่เป็นSPI
- **มีวงจร I2C Bus จำนวน 1 ช่อง** สำหรับเชื่อมต่อกับอุปกรณ์สนับสนุนต่างๆที่เป็น I2C
- **มีวงจร ADC ความละเอียด 12บิต** จำนวน 15 ช่อง บรรจุไว้ สามารถเลือก เปิด-ปิด การทำงานแยกเป็นช่องๆได้โดยอิสระ และสามารถกำหนดการ Trigger ให้ ADC เริ่มต้นแปลงค่าจากอุปกรณ์อื่นๆ เช่น Timer Overflow, External Interrupt หรือ Pin Change ได้
- **มีวงจรควบคุม Buzzer** โดยสามารถกำหนดค่าความถี่ในการกำเนิดเสียงได้เอง โดยกำหนดเงื่อนไขต่างๆให้กับรีจิสเตอร์แล้ววงจรจะสร้างสัญญาณควบคุมการกำเนิดเสียงของ Buzzer ให้เอง ทำให้การเขียนโปรแกรมสร้างความถี่เสียงต่างๆเป็นเรื่องง่ายและสะดวกมากยิ่งขึ้น
- **ด้านการ Interrupt** สามารถกำหนดเงื่อนไขการ Interrupt ได้จากอุปกรณ์หลายๆแหล่ง รองรับการ Interrupt 32 แหล่งและแยกการบริการ Interrupt ได้มากถึง 32 Vector Address
- **มีระบบตรวจสอบการรีเซ็ต** สามารถเลือกกำหนดระบบการรีเซ็ตและตรวจสอบแหล่งของสัญญาณรีเซ็ตได้ทำให้ผู้ใช้สามารถตรวจสอบได้ว่า MCU ถูกรีเซ็ตจาก External Reset หรือ Power-On Reset หรือ Brown-Out Reset หรือ Watch-Dog Overflow Reset หรือ OCD Reset เพื่อเลือกจัดการเงื่อนไขให้ MCU ทำงานต่อหลังจากถูกรีเซ็ตได้อย่างถูกต้อง

- **สนับสนุนโหมดประหยัดพลังงาน Power Down Mode 3 ระดับ**
  - IDLE Mode หยุดการทำงานของส่วนประมวลผลแต่ Peripheral I/O ยังทำงานต่อ
  - STOP Mode1 หยุดการทำงานของส่วนประมวลผลและ Peripheral I/O บางส่วน
  - STOP Mode2 หยุดการทำงานของส่วนประมวลผลและส่วนกำเนิดสัญญาณนาฬิกาทั้งหมดยกเว้น Sub Clock และ Peripheral I/O บางส่วน
- **ด้านการพัฒนาโปรแกรม** มีระบบ Debugger บรรจุไว้ภายในชิพ ทำให้ผู้ใช้สามารถตรวจสอบค่ารีจิสเตอร์ และ หน่วยความจำ ต่างๆในขณะที่ MCU ทำงานอยู่ จึงทำให้การตรวจสอบความถูกต้องและค้นหาข้อผิดพลาดต่างๆทำได้อย่างง่ายดาย ผู้ใช้สามารถเข้าถึงอุปกรณ์ต่างๆ สามารถสั่งหยุดการทำงาน ปรับแต่ง แก้ไขค่าให้กับรีจิสเตอร์และหน่วยความจำเพื่อทดสอบการทำงานของโปรแกรม โดยไม่ต้องใช้การแก้ไข ลบ และ โปรแกรม Code ซ้ำใหม่บ่อยๆให้ยุ่งยากเสียเวลา ทำให้การพัฒนาโปรแกรมมีความสะดวกมากยิ่งขึ้น
  - สามารถเข้าถึง Peripheral I/O ต่างๆได้
  - สามารถเข้าถึงหน่วยความจำ IRAM ได้
  - สามารถตรวจสอบค่า Program Counter(PC) ได้
  - สามารถหยุดการทำงานของ MCU เพื่อตรวจสอบค่าต่างๆได้
  - สามารถควบคุมให้ MCU ทำงานทีละคำสั่ง เพื่อตรวจสอบผลการทำงานได้
  - ขาสัญญาณที่ใช้ในการ Debug จะแยกอิสระออกจาก Pin Port ทำให้ไม่ต้องเสียขาสัญญาณ I/O ในการ Program และ Debug

รู้จักระบบสัญญาณนาฬิกาของ Z8051



สำหรับระบบสัญญาณนาฬิกาของ Z8051 นั้นสามารถเลือกกำหนดให้ MCU เลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาได้จาก 4 แหล่งกำเนิด และสามารถสั่ง สลับเปลี่ยนแหล่งกำเนิดของสัญญาณนาฬิกาให้กับ MCU โดยค่าเริ่มต้นของ MCU หลังการรีเซ็ตทุกครั้งนั้น MCU จะเริ่มต้นทำงานจากแหล่งกำเนิดสัญญาณนาฬิกาภายใน Internal RC Oscillator 16MHz ผ่านวงจรหารความถี่ 1/2 เสมอ ดังนั้น MCU จะเริ่มต้นทำงานจากสัญญาณนาฬิกาค่าความถี่ 8.00 MHz เสมอตอนเริ่มต้น ซึ่งหลังจากพ้นสภาวะการรีเซ็ตแล้ว ผู้ใช้สามารถสั่งเปลี่ยนแปลงแหล่งกำเนิดสัญญาณนาฬิกาให้กับ MCU ได้ตลอดเวลาจากเงื่อนไขของคำสั่งในโปรแกรม

ซึ่งการที่จะพิจารณาว่าจะเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาจากแหล่งใดนั้น อาจต้องคำนึงถึงเหตุผลหลายๆประการ ทั้งด้านต้นทุน แหล่งกำเนิดพลังงาน ความเที่ยงตรงแม่นยำ สัญญาณรบกวนในระบบ หรือ หลายๆองค์ประกอบรวมกัน เช่น ถ้าใช้ Main Clock จะมีความเที่ยงตรงแม่นยำสูง แต่อาจเป็นแหล่งกำเนิดสัญญาณรบกวนในระบบได้ และใช้พลังงานมาก เพิ่มต้นทุนอุปกรณ์ ถ้าใช้ Sub Clock มีความเที่ยงตรงแม่นยำสูง สัญญาณรบกวนในระบบต่ำ กินกำลังไฟต่ำกว่าแบบอื่นๆ แต่ใช้เวลาในการ Start Up นานกว่า Main Clock ถ้าใช้ Internal RC ต้นทุนต่ำ กินกำลังไฟมาก ทำงานได้เร็ว แต่มีค่าความผิดพลาดคาดเคลื่อนของความถี่ของสัญญาณนาฬิกาที่อาจเกิดจากสิ่งแวดล้อมในการใช้งาน เช่น อุณหภูมิเปลี่ยนแปลงก็อาจทำให้สัญญาณนาฬิกาเกิดความเปลี่ยนแปลงตามไปด้วยได้เช่นเดียวกัน

สำหรับรีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของสัญญาณนาฬิกา จะมีรีจิสเตอร์สำหรับใช้กำหนดการทำงานระบบอยู่ด้วยกัน 2 ชุด สำหรับเลือกแหล่งกำเนิดสัญญาณนาฬิกาให้กับ MCU และกำหนดค่าการคูณความถี่ให้กับวงจร Phase Lock Loop ในกรณีที่เลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาจาก Sub Clock Oscillator โดยมีรายละเอียดดังนี้คือ

รีจิสเตอร์ SCCR (System and Clock Control Register)

STOP1	DIV1	DIV0	CBYS	ISTOP	XSTOP	CS1	CS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- **STOP1** ใช้กำหนดการทำงานของสัญญาณนาฬิกาใน STOP Mode(PCON=0x03) แต่ถ้า MCU ไม่ถูกกำหนดให้ทำงานใน STOP Mode บิตนี้จะไม่มีผล
    - เมื่อ PCON=0x03 และบิตนี้เป็น “1” จะเป็นการเลือก STOP Mode ในโหมด STOP1
    - เมื่อ PCON=0x03 และบิตนี้เป็น “0” จะเป็นการเลือก STOP Mode ในโหมด STOP2
- **DIV1:DIV0** ใช้กำหนดการหารความถี่ของสัญญาณนาฬิกาจาก Internal RC (16MHz) แต่ถ้าเลือกใช้สัญญาณนาฬิกาจากแหล่งอื่นที่ไม่ใช่ Internal RC จะไม่มีผลต่อการทำงาน
    - 0:0 เป็นการกำหนดให้ Internal RC/1 จะได้ความถี่ระบบ 16MHz
    - 0:1 เป็นการกำหนดให้ Internal RC/2 จะได้ความถี่ระบบ 8MHz
    - 1:0 เป็นการกำหนดให้ Internal RC/4 จะได้ความถี่ระบบ 4MHz
    - 1:1 เป็นการกำหนดให้ Internal RC/8 จะได้ความถี่ระบบ 2MHz
- **CBYS** ใช้กำหนดเงื่อนไขการเปลี่ยนแปลงแหล่งกำเนิดสัญญาณนาฬิกาให้กับ MCU
    - เมื่อกำหนดเป็น “0” เป็นการกำหนดให้สัญญาณนาฬิกาถูกเปลี่ยนโดย Hardware ซึ่งหมายถึง เมื่อมีการสั่งเปลี่ยนแปลงค่าในบิต CS1:CS0 เพื่อเลือกแหล่งกำเนิดของสัญญาณนาฬิกาใหม่ สัญญาณนาฬิกาจะไม่ถูกสลับในทันที แต่สัญญาณนาฬิกาจะถูกเปลี่ยนหลังจาก MCU เข้าทำงานใน STOP Mode และสัญญาณนาฬิกาจะถูกสลับตามค่าที่เลือกไว้ในบิตของ CS1:CS0 เมื่อ MCU จบการทำงานจาก STOP Mode แล้ว โดย MCU จะทำงานตามแหล่งกำเนิดสัญญาณนาฬิกาแหล่งใหม่เมื่อ MCU เริ่มต้น Wakeup จาก STOP Mode แล้ว
    - เมื่อกำหนดเป็น “1” เป็นการกำหนดให้สัญญาณนาฬิกาถูกเปลี่ยนโดย Software

- **ISTOP** ใช้กำหนดการทำงานของวงจรกำเนิดสัญญาณนาฬิกาจาก Internal RC (16MHz)
  - เมื่อกำหนดเป็น “0” เป็นการเปิดการทำงาน(Enable)ของ Internal RC
  - เมื่อกำหนดเป็น “1” เป็นการปิดการทำงาน(Disable)ของ Internal RC
- **XSTOP** ใช้กำหนดการทำงานของวงจรกำเนิดสัญญาณนาฬิกาจาก Crystal Oscillator แต่ถ้ามีการกำหนดให้บิต XINENA ในรีจิสเตอร์ Configuration ของ FUSE_CON เป็น “0” ซึ่งเป็นการสั่งปิดการทำงานของ Main Clock Oscillator บิตนี้จะถูกกำหนดให้มีค่าเป็น “1” โดยอัตโนมัติ
  - เมื่อกำหนดเป็น “0” เป็นการเปิดการทำงาน(Enable)ของ X-TAL Oscillator
  - เมื่อกำหนดเป็น “1” เป็นการปิดการทำงาน(Disable)ของ X-TAL Oscillator
- **CS1:CS0** ใช้สำหรับเลือกแหล่งกำเนิดสัญญาณนาฬิกาให้กับ MCU
  - 0:0 เป็นการเลือกใช้สัญญาณนาฬิกาจาก Internal RC(16MHz)
  - 0:1 เป็นการเลือกใช้สัญญาณนาฬิกาจาก Main Clock Oscillator(1-10MHz)
  - 1:0 เป็นการเลือกใช้สัญญาณนาฬิกาจาก Sub Clock Oscillator(32.768KHz+PLL)
  - 1:1เป็นการเลือกใช้สัญญาณนาฬิกาจาก Ring Oscillator(125KHz)

รีจิสเตอร์ PLLCR (Phase Locked Loop Control Register)

PLLSTAT	PLLCKS	VDConSUB	PLLFB		PLLPD		PLLEN
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

รีจิสเตอร์ PLLCR ใช้สำหรับกำหนดการทำงานให้กับวงจร Phase Lock Loop สำหรับคูณความถี่ 32.768KHz ของ Sub Clock Oscillator โดยขนาดของความถี่ Output ของสัญญาณ PLL จะมีค่าเท่ากับ (32.768KHz x PLLFB) / PLLPD

- **PLLSTAT** เป็นแฟลกแสดงสถานะของ Phase Lock Loop บิตนี้อ่านได้อย่างเดียว
  - เมื่อบิตนี้เป็น “0” หมายถึงความถี่ Output ของวงจรมีค่า 32.768KHz(By Pass)
  - เมื่อบิตนี้เป็น “1” หมายถึงความถี่ Output ของวงจร ถูกคูณความถี่จากวงจรแล้ว
- **PLLCKS** ใช้กำหนดการทำงานของวงจรคูณความถี่ของ Phase Lock Loop
  - เมื่อกำหนดบิตนี้เป็น “0” หมายถึง ปิดการทำงานของวงจรคูณความถี่
  - เมื่อกำหนดบิตนี้เป็น “1” หมายถึง เปิดการทำงานของวงจรคูณความถี่

- **VDConSUB** ใช้กำหนดพิกัดกระแสที่จ่ายให้วงจรคูณความถี่ ซึ่งค่าตัวเลือกนี้จะมีผลมากเมื่อต้องใช้งานวงจรในงานที่ต้องประหยัดพลังงานมากๆ
    - เมื่อกำหนดให้บิตนี้เป็น "0" จะเป็นการจำกัดกระแสที่จะจ่ายให้กับวงจรคูณความถี่ โดยกรณีนี้กระแสจะถูกจำกัดอยู่ระหว่าง 0.1mA-1mA
    - เมื่อกำหนดให้บิตนี้เป็น "1" จะเป็นการเปิดให้วงจรสามารถใช้กระแสได้เต็มที่ ซึ่งตามปรกติวงจรอาจจะกินกระแส 1mA-10mA
- **PLLFB** ใช้กำหนดค่า PLL Feedback Divider (FBdiv) ให้กับวงจร Phase Lock Loop
    - 0:0 กำหนดให้ FBdiv มีค่า 674(ไม่สามารถใช้ค่านี้ได้)
    - 0:1 กำหนดให้ FBdiv มีค่า 562(ไม่สามารถใช้ค่านี้ได้)
    - 1:0 กำหนดให้ FBdiv มีค่า 450
    - 1:1 กำหนดให้ FBdiv มีค่า 338
- **PLLPD** ใช้กำหนดค่า PLL Post Divider(M) ให้กับวงจร Phase Lock Loop
    - 0:0 กำหนดค่า M = 1
    - 0:1 กำหนดค่า M = 2
    - 1:0 กำหนดค่า M = 4
    - 1:1 กำหนดค่า M = 8
- **PLLEN** ใช้กำหนดการทำงานของวงจร Phase Lock Loop
    - เมื่อกำหนดให้บิตนี้เป็น "0" เป็นการปิดการทำงาน (Disable) การคูณความถี่ของวงจรคูณความถี่ PLL
    - เมื่อกำหนดให้บิตนี้เป็น "1" เป็นการเปิดการทำงาน (Enable) การคูณความถี่ของวงจรคูณความถี่ PLL

หมายเหตุ ค่าที่แนะนำสำหรับกำหนดค่าให้กับวงจร Phase Lock Loop

Fvco = (32.768KHz x 450) / 1 = 14.7456MHz

Fvco = (32.768KHz x 338) / 1 = 11.075584 MHz

การเลือกใช้สัญญาณนาฬิกาจาก Main Clock Oscillator

แหล่งกำเนิดสัญญาณนาฬิกาแบบนี้สามารถทำงานได้อย่างรวดเร็ว มีความเที่ยงตรงแม่นยำสูงมาก แต่ก็ต้องแลกด้วยต้นทุนเรื่องอุปกรณ์และการกินพลังงานที่สูงกว่าการเลือกใช้ Sub Clock Oscillator และอาจมีสัญญาณรบกวนเกิดขึ้นในระบบได้ด้วย

```
#include <z51f6412.h>                      // Z8051 : Z51F6412

void main(void)
{
  char ch;
  int i;

  /* Config System Clock = External XTAL 10.00 MHz */
  //PLLCR = 0,0,0,00,00,0
  // 0xxxxxxx : PLL Output Status
  // x0xxxxxx : PLL Output Bypass
  // xx0xxxxx : Power PLL = Default
  // xxx00xxx : FBDiv = Default
  // xxxxx00x : PLL M = Default
  // xxxxxxx0 : PLL Disable
  PLLCR = 0x00;                            // Disable PLL

  // SCCR = 0,00,1,1,0,01
  // 0xxxxxxx : Stop Mode = Mode 2
  // x00xxxxx : Clock Divide 1
  // xxx1xxxx : Clock Change By Software
  // xxxx1xxx : RC Oscillator Disable
  // xxxxx0xx : XTAL Oscillator Enable
  // xxxxxx01 : System Clock Source = Main Clock(1~10 MHz)
  SCCR = 0x10;                             // Enable Main XTAL-10MHz

  for(i=0; i<32000; i++);                  // Delay for XTAL
Stabilization
  SCCR = 0x19;                             // Select Clock = XTAL-10MHz
&
                                           // Stop Internal RC
  /* Now System Clock = 10.00MHz */
  .
  .
  .
}
```

ตัวอย่างโปรแกรม สำหรับกำหนด MCU ทำงานจาก Main Clock Oscillator

การเลือกใช้สัญญาณนาฬิกาจาก Sub Clock Oscillator

แหล่งกำเนิดสัญญาณนาฬิกาแบบนี้มีความเที่ยงตรงแม่นยำสูงเช่นเดียวกันกับการเลือกใช้ Main Clock Oscillator แต่มีสัญญาณรบกวนน้อยกว่า และการเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาจาก Sub Clock Oscillator นี้จะมีข้อดีกว่าแบบอื่นๆ คือ เรื่องประหยัดพลังงาน ซึ่งสามารถประหยัดพลังงานได้มากกว่าการเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาจากแหล่งอื่นๆทั้งหมด แต่ก็ต้องแลกด้วยต้นทุนเรื่องอุปกรณ์และการเริ่มต้นทำงานที่ช้ากว่าแบบอื่นๆ

```
#include <z51f6412.h>               // Z8051 : Z51F6412

void main(void)
{
  char ch;
  //int i;

  /* Config System Clock = 32.768KHz+PLL(14.7456MHz) */
  //PLLCR = 0,1,0,10,00,1
  // 0xxxxxxx : PLL Status Read Only
  // x1xxxxxx : PLL Output Enable
  // xx0xxxxx : Power PLL = Default
  // xxx10xxx : FBDiv = 450
  // xxxxx00x : PLL M = 1
  // xxxxxxx1 : PLL Enable
  PLLCR = 0x51;                     // Enable PLL

  // SCCR = 0,00,1,1,0,10
  // 0xxxxxxx : Stop Mode = Mode 2
  // x00xxxxx : Clock Divide 1
  // xxx1xxxx : Clock Change By Software
  // xxxx1xxx : RC Oscillator Disable
  // xxxxx0xx : XTAL Oscillator Enable
  // xxxxxx10 : System Clock Source = 32.768KHz
  SCCR = 0x1A;                      // Run XTAL-32.768KHz+PLL
  while((PLLCR & 0x80) != 0x80);    // Wait PLL Output Lock
  /* Now System Clock = 14.7456MHz */
  .
  .
  .
}
```

ตัวอย่างโปรแกรม สำหรับกำหนด MCU ทำงานจาก Sub Clock Oscillator + PLL

การเลือกใช้สัญญาณนาฬิกาจาก Internal RC Oscillator

แหล่งกำเนิดสัญญาณนาฬิกาแบบนี้มีความเที่ยงตรงแม่นยำต่ำกว่าการเลือกใช้สัญญาณนาฬิกาจาก Crystal Oscillator แต่ประหยัดต้นทุนได้มากที่สุด เพราะไม่ต้องสูญเสียขาสัญญาณ I/O ในการเชื่อมต่อกับ Crystal ภายนอก แต่สัญญาณนาฬิกาที่ได้อาจมีความผิดพลาดคาดเคลื่อนเนื่องจากอุณหภูมิใช้งานที่เปลี่ยนแปลงไปได้เช่นเดียวกัน และกินพลังงานมากกว่าแบบอื่นๆ

แต่อย่างไรก็ตามในการใช้งานทั่วๆไป สัญญาณนาฬิกาที่ได้จากแหล่งกำเนิดนี้ ถือว่ามีระดับความเที่ยงตรงเพียงพอกับการใช้งาน เพราะค่าความคาดเคลื่อนไม่เกิน +/-2% เท่านั้น การสื่อสารข้อมูลต่างๆทั้ง SPI, I2C, USART ต่างๆ ยังสามารถใช้งานได้อย่างถูกต้องเช่นเดิม

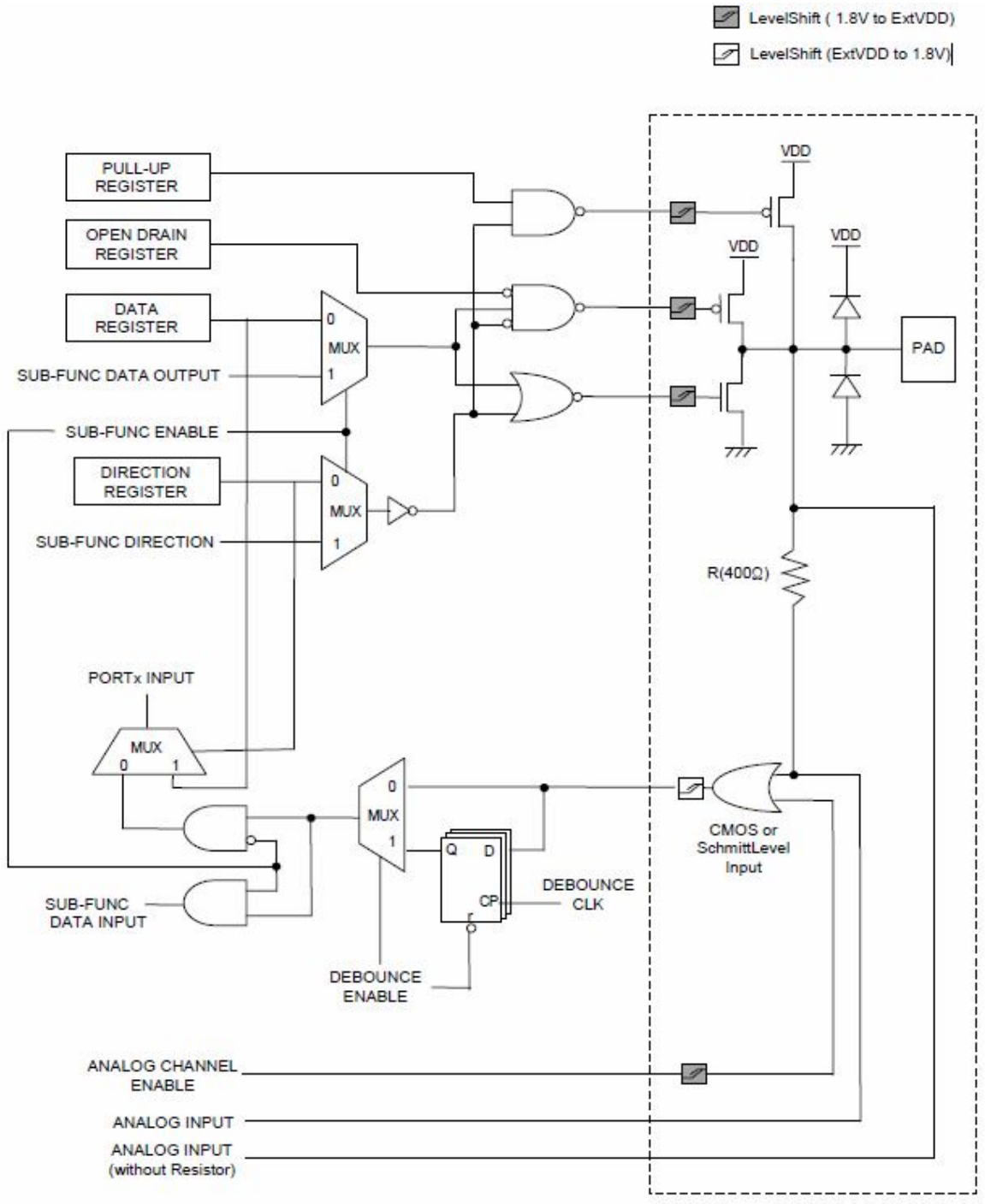
```
#include <z51f6412.h>                // Z8051 : Z51F6412

void main(void)
{
  /* Config System Clock = Internal RC 8.00 MHz */
  //PLLCR = 0,0,0,00,00,0
  // 0xxxxxxx : PLL Output Status
  // x0xxxxxx : PLL Output Bypass
  // xx0xxxxx : Power PLL = Default
  // xxx00xxx : FBDiv = Default
  // xxxxx00x : PLL M = Default
  // xxxxxxx0 : PLL Disable
  PLLCR = 0x00;                      // Disable PLL

  // SCCR = 0,01,0,0,1,00
  // 0xxxxxxx : Stop Mode = Mode 2
  // x01xxxxx : INTRC Clock Divide = INTRC(16MHz)/2 = 8MHz
  // xxx0xxxx : Clock Change By Hardware
  // xxxx0xxx : RC Oscillator Enable
  // xxxxx1xx : XTAL Oscillator Disable
  // xxxxxx00 : System Clock Source = INTRC(16MHz)
  SCCR = 0x24;                       // INT-RC 8MHz
  /* Now System Clock = 8.00MHz */
  .
  .
  .
}
```

ตัวอย่างโปรแกรม สำหรับกำหนด MCU ทำงานจาก Internal RC ค่า 8.00 MHz

รู้จัก GPIO Port ของ Z8051



ระบบ Port I/O ของ Z8051 จะมีวงจรแปลงระดับโลจิกสัญญาณจาก +1.8V ให้เป็น +VDD และแปลงระดับโลจิกจาก +VDD ให้เป็น +1.8V บรรจุไว้ในทุกๆ Pin โดย +VDD ภายนอกสามารถใช้กับแหล่งจ่ายขนาด +2.0V ถึง +5.5V ได้โดยไม่มีปัญหา

โดยระบบ GPIO ของ Z8051 จะแบ่งออกเป็นชุดๆ เรียกว่า พอร์ต (Port) โดยแต่ละพอร์ตจะมีขนาด 8 บิต แต่ละบิตของพอร์ตจะเรียกว่า พิน (Pin) สามารถกำหนดการทำงานแยกกันได้โดยอิสระ โดยในแต่ละ Port จะมีรีสเตอร์ขนาด 8 บิต สำหรับกำหนด ควบคุมการทำงาน และ อ่าน เขียน ข้อมูลของแต่ละ Pin ใน Port โดยข้อมูลใน บิต0 ของรีจิสเตอร์ก็จะใช้ควบคุมสั่งงานเข้าถึง Pin0 ของ Port และทำนองเดียวกัน ค่าใน บิต1 ของรีจิสเตอร์ก็จะใช้ควบคุมสั่งงานเข้าถึง Pin1 ของ Port เรียงลำดับกันไปทั้ง 8บิต ตามลำดับ

โดย GPIO Port จะมีรีจิสเตอร์จำนวน 5 ชุด สำหรับเข้าถึงและกำหนดการทำงานของ Port โดยทุกๆ Port สามารถเลือกเข้าถึงและกำหนดการทำงานแยกอิสระเป็น Pin ได้ตามต้องการดังนี้

- **PxIO** ใช้กำหนดหน้าที่ Input / Output ของแต่ละ Port (Port Direction)
- **Px** ใช้สำหรับ อ่าน เขียน ข้อมูลกับแต่ละ Port (Port Data)
- **PxPU** ใช้กำหนดการ Pull-Up สำหรับ Input Port(Port Pull-Up)
- **PxDB** ให้กำหนดการ Debounce สำหรับ Input Port(Port Debounce)
- **PxOD** ใช้กำหนดรูปแบบ Output ของ Port ให้เป็นแบบ Open-Drain

นอกจากรีจิสเตอร์ทั้ง 5 ชุดดังที่กล่าวมาข้างต้นแล้ว ยังมีรีจิสเตอร์พิเศษอีก 4 ชุด สำหรับกำหนดการทำงานให้กับเฉพาะบาง Port คือ

- **PCI0(P0 Pin Change Interrupt Enable Register)** ใช้กำหนดการทำงานของ Port P0 ให้ทำหน้าที่ตรวจจับการเปลี่ยนแปลงของ Input แบบ Pin Change Interrupt
- **PCI7(P7 Pin Change Interrupt Enable Register)** ใช้กำหนดการทำงานของ Port P7 ให้ทำหน้าที่ตรวจจับการเปลี่ยนแปลงของ Input แบบ Pin Change Interrupt
- **PSR0(Port Select Register0)** ใช้เลือกการทำงานของ Port P2 ว่าจะให้เป็นแบบ Analog(ADC) หรือแบบ Digital (Port P2)
- **PSR1(Port Select Register1)** ใช้เลือกการทำงานของ Port P3 ว่าจะให้เป็นแบบ Analog(ADC) หรือแบบ Digital (Port P3)

รีจิสเตอร์ PxIO (Px Direction Register)

Bit	DxIO7	PxIO6	PxIO5	PxIO4	PxIO3	PxIO2	PxIO1	PxIO0
Pin	Px.7	Px.6	Px.5	Px.4	Px.3	Px.2	Px.1	Px.0

รีจิสเตอร์ **PxIO(P0IO,P1IO,P2IO,P3IO,P4IO,P5IO,P6IO,P7IO,P8IO)** ทำหน้าที่กำหนดหน้าที่ของพอร์ต (Port Direction) โดยทุกๆ Pin ของทุกๆ Port สามารถกำหนดหน้าที่การทำงานให้เป็นได้ทั้ง Input หรือ Output ได้ตามต้องการ

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะทำให้ Pin นั้นจะทำหน้าที่เป็น Input
- บิตใดถูกกำหนดให้มีค่าเป็น “1” จะทำให้ Pin นั้นจะทำหน้าที่เป็น Output

รีจิสเตอร์ Px (Px Data Register)

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Pin	Px.7	Px.6	Px.5	Px.4	Px.3	Px.2	Px.1	Px.0

รีจิสเตอร์ **Px(P0,P1,P2,P3,P4,P5,P6,P7,P8)** ทำหน้าที่เป็นตัวกลางในการ อ่าน เขียนข้อมูลกับพอร์ต (Port Data) โดยถ้าอ่านค่ารีจิสเตอร์นี้จะเป็นการอ่านค่าสถานะจาก Pin Port ในขณะนั้นมายังรีจิสเตอร์ ถ้าพอร์ตถูกกำหนดเป็น Input ค่าที่อ่านได้จะเป็นค่าจาก Pin ภายนอก แต่ถ้าพอร์ตถูกกำหนดเป็น Output ค่าที่อ่านได้จะเป็นค่า Port Latch แทน และถ้าเขียนค่าไปยังรีจิสเตอร์นี้จะเป็นการกำหนดค่า Output ให้กับ Pin Port นั้นๆ

รีจิสเตอร์ PxPU (Px Pull-up Resistor Selection Register)

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Pin	Px.7	Px.6	Px.5	Px.4	Px.3	Px.2	Px.1	Px.0

รีจิสเตอร์ **PxPU(P0PU,P1PU,P2PU,P3PU,P4PU,P5PU,P6PU,P7PU,P8PU)** ทำหน้าที่กำหนดการ Pull-Up ให้กับ Pin Port (Port Pull-Up) โดยทุกๆ Pin ของทุกๆ Port ที่ถูกกำหนดให้ทำหน้าที่เป็น Input สามารถกำหนดให้มีการ Pull-Up ภายใน เพื่อคงค่าสถานะโลจิกให้กับแต่ละ Pin ที่เป็น Input ได้โดยอิสระ

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะเป็นการสั่งปิด(Disable) การ Pull-Up ของ Pin นั้นๆ
- บิตใดถูกกำหนดให้มีค่าเป็น “1”จะเป็นการสั่งเปิด(Enable) การ Pull-Up ของ Pin นั้นๆ

รีจิสเตอร์ PxDB (Px Debounce Enable Register)

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Pin	Px.7	Px.6	Px.5	Px.4	Px.3	Px.2	Px.1	Px.0

รีจิสเตอร์ **PxDB(P0DB,P1DB,P2DB,P3DB,P4DB,P5DB,P6DB,P7DB,P8DB)** ทำหน้าที่กำหนดการ Debounce ให้กับ Pin Port โดยทุกๆ Pin ของทุกๆ Port ที่ถูกกำหนดให้ทำหน้าที่เป็น Input สามารถกำหนดให้มีการ Debounce ภายใน เพื่อแก้ปัญหาการ Bounce ที่เกิดจากการเชื่อมต่อ Input กับ อุปกรณ์จำพวกสวิตช์ที่เป็นหน้าสัมผัสต่างๆ โดยวงจร Debounce ของ Z8051 จะทำการ Debounce สัญญาณเป็นเวลาประมาณ 5uS โดยอัตโนมัติถ้า Pin นั้นๆถูกสั่งเปิดการทำงานของวงจร Debounce ไว้ ดังนันถ้า Pin ใดต่อกับ Input ที่เป็นหน้าสัมผัส ผู้ใช้ควรสั่งเปิด(Enable) การทำงานของวงจร Debounce ให้กับ Pin นั้นๆไว้เสมอ แต่ถ้า Pin ใดใช้กับ Input ที่เป็น Signal Logic ต่างๆ ผู้ใช้ก็สามารถเลือกสั่ง ปิด(Disable) การทำงานของวงจร Debounce ของ Pin นั้นๆได้ด้วย ตามต้องการ

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะเป็นการสั่งปิด(Disable) การ Debounce ของ Pin นั้นๆ
- บิตใดถูกกำหนดให้มีค่าเป็น “1”จะเป็นการสั่งเปิด(Enable) การ Debounce ของ Pin นั้นๆ

รีจิสเตอร์ PxOD(Px Open-drain Selection Register)

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Pin	Px.7	Px.6	Px.5	Px.4	Px.3	Px.2	Px.1	Px.0

รีจิสเตอร์ PxOD(P0OD,P1OD,P2OD,P3OD,P4OD,P5OD,P6OD,P7OD,P8OD) ทำหน้าที่กำหนดรูปแบบการทำงานของ Output ให้เป็นแบบ Open Drain โดยทุกๆ Pin ของทุกๆ Port ที่ถูกกำหนดให้ทำหน้าที่เป็น Output ตามปรกติแล้วจะเป็นแบบ Push-Pull (มีการ Pull-Up ภายในไว้แล้ว) แต่ในกรณีที่ผู้ใช้ต้องการกำหนดให้ Output Port เป็นแบบ Open Drain ก็สามารถกำหนดได้จากรีจิสเตอร์นี้ ซึ่งในกรณีนี้ผู้ใช้ต้องต่อ External Resistor ให้กับแต่ละ Pin ที่กำหนดให้เป็น Open Drain เองด้วย โดยผู้ใช้สามารถกำหนดให้ทุกๆ Pin Port Output เป็น Open Drain ได้โดยอิสระทุก Pin

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะเป็นการเลือก Output แบบ Internal Pull-Up
- บิตใดถูกกำหนดให้มีค่าเป็น “1”จะเป็นการเลือก Open-Drain Output

รีจิสเตอร์ PCI0 (P0 Pin Change Interrupt Enable Register)

Bit	PCI07	PCI06	PCI05	PCI04	PCI03	PCI02	PCI01	PCI00
Pin	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

รีจิสเตอร์ PCI0 ทำหน้าที่กำหนดรูปแบบการทำงานของ Port P0 ว่าจะให้ Pin ใด ทำหน้าที่เป็น Input ตรวจจับการเปลี่ยนแปลง Pin Change Interrupt บ้าง

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะเป็นการไม่เลือก Pin นั้นเป็น Input Pin Change Interrupt
- บิตใดถูกกำหนดให้มีค่าเป็น “1” จะเป็นการเลือกให้ Pin นั้นเป็น Input Pin Change Interrupt

รีจิสเตอร์ PCI7 (P7 Pin Change Interrupt Enable Register)

Bit	PCI77	PCI76	PCI75	PCI74	PCI73	PCI72	PCI71	PCI70
Pin	P7.7	P7.6	P7.5	P7.4	P7.3	P7.2	P7.1	P7.0

รีจิสเตอร์ PCI7 ทำหน้าที่กำหนดรูปแบบการทำงานของ Port P7 ว่าจะให้ Pin ใด ทำหน้าที่เป็น Input ตรวจจับการเปลี่ยนแปลง Pin Change Interrupt บ้าง

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะเป็นการไม่เลือก Pin นั้นเป็น Input Pin Change Interrupt
- บิตใดถูกกำหนดให้มีค่าเป็น “1” จะเป็นการเลือกให้ Pin นั้นเป็น Input Pin Change Interrupt

รีจิสเตอร์ PSR0 (Port Selection Register 0)

Bit	PSR07	PSR06	PSR05	PSR04	PSR03	PSR02	PSR01	PSR00
Pin	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
Analog	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

รีจิสเตอร์ PSR0 ทำหน้าที่กำหนดรูปแบบการทำงานของ Port P2 ว่าจะให้ทำหน้าที่เป็น Analog (ADC) หรือแบบ Digital I/O(Port P2)

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะเป็นการเลือก Pin นั้นเป็น Digital Pin
- บิตใดถูกกำหนดให้มีค่าเป็น “1” จะเป็นการเลือก Pin นั้นเป็น Analog Pin

รีจิสเตอร์ PSR1 (Port Selection Register 1)

Bit	PSR17	PSR16	PSR15	PSR14	PSR13	PSR12	PSR11	PSR10
Pin	-	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
Analog	-	AN14	AN13	AN12	AN11	AN10	AN9	AN8

รีจิสเตอร์ PSR1 ทำหน้าที่กำหนดรูปแบบการทำงานของ Port P3 ว่าจะให้ทำหน้าที่เป็น Analog (ADC) หรือแบบ Digital I/O(Port P3)

- บิตใดถูกกำหนดให้มีค่าเป็น “0” จะเป็นการเลือก Pin นั้นเป็น Digital Pin
- บิตใดถูกกำหนดให้มีค่าเป็น “1” จะเป็นการเลือก Pin นั้นเป็น Analog Pin

Port P0

โดยสัญญาณจาก Port P0 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ทั้งแบบ GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P0[7..0])
- ใช้งานเป็นขาตรวจจับการเปลี่ยนแปลง Input (Input Pin Change Interrupt : PCI0[7..0])
- ใช้งานเป็น UART0 Function(P0[3..0])
- ใช้เชื่อมต่อกับ Sub Crystal Oscillator ค่า 32.768KHz
- ใช้งานเป็น I2C Bus Function(P0[7..6])

Port	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
GPIO	I/OP0.7	I/OP0.6	I/OP0.5	I/O P0.4	I/OP0.3	I/O P0.2	I/O P0.1	I/O P0.0
PCIO	PCI0.7	PCI0.6	PCI0.5	PCI0.4	PCI0.3	PCI0.2	PCI0.1	PCI0.0
UART0	-	-	-	-	RXD0	TXD0	ACK0	USS0
XTAL	-	-	XOUT	XIN	-	-	-	-
I2C	SDA	SCL	-	-	-	-	-	-

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่าง ๆของ Port P0

Port P1

โดยสัญญาณจาก Port P1 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ทั้งแบบ GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P1[7..0])
- ใช้งานเป็นขาตรวจจับการ Interrupt จากภายนอก (External Interrupt :INT[7..0])

Port	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
GPIO	I/O P1.7	I/O P1.6	I/O P1.5	I/O P1.4	I/O P1.3	I/O P1.2	I/O P1.1	I/O P1.0
INT	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่าง ๆของ Port P1

Port P2

โดยสัญญาณจาก Port P2 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ ทั้งแบบ GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P2[7..0])
- ใช้งานเป็นขา Input Analog ADC ขนาดความละเอียด 12บิต (AN[7..0])
- ใช้งานเป็น UART3 Function(P2[7..4])

Port	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
GPIO	I/O P2.7	I/O P2.6	I/O P2.5	I/O P2.4	I/O P2.3	I/O P2.2	I/O P2.1	I/O P2.0
ADC	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
UART3	RXD3	TXD3	ACK3	USS3	-	-	-	-

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่าง ๆของ Port P2

Port P3

โดยสัญญาณจาก Port P3 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ ทั้งแบบ GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P3[7..0])
- ใช้งานเป็นขา Input Analog ADC ขนาดความละเอียด 12บิต (AN[14..8])
- ใช้งานเป็น UART1 Function(P3[3..0])
- ใช้งานเป็น SPI0 Function(P3[7..4])

Port	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
GPIO	I/O P3.7	I/O P3.6	I/O P3.5	I/O P3.4	I/O P3.3	I/O P3.2	I/O P3.1	I/O P3.0
ADC	-	AN14	AN13	AN12	AN11	AN10	AN9	AN8
UART1	-	-	-	-	RXD1	TXD1	ACK1	USS1
SPI0	MISO0	MOSI0	SCK0	SSS0	-	-	-	-

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่าง ๆของ Port P3

Port P4

โดยสัญญาณจาก Port P4 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ ทั้งแบบ GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P4[7..0])
- ใช้งานเป็น UART2 Function(P4[3..0])
- ใช้งานเป็น SPI1 Function(P4[7..4])

Port	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
GPIO	I/O P4.7	I/O P4.6	I/O P4.5	I/O P4.4	I/O P4.3	I/O P4.2	I/O P4.1	I/O P4.0
UART2	-	-	-	-	RXD2	TXD2	ACK2	USS2
SPI1	MISO1	MOSI1	SCK1	SSS1	-	-	-	-

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่างๆของ Port P4

Port P5

โดยสัญญาณจาก Port P5 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ ทั้งแบบ GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P5[7..0])
- ใช้งานเป็น Buzzer Drive(P5[0])
- ใช้งานเป็น PWM Function(P5[7..3])
- ใช้งานเป็น Timer Input/Output Function(P5[7..1])

Port	**P5.7**	**P5.6**	**P5.5**	**P5.4**	**P5.3**	**P5.2**	**P5.1**	**P5.0**
GPIO	I/O P5.7	I/O P5.6	I/O P5.5	I/O P5.4	I/O P5.3	I/O P5.2	I/O P5.1	I/O P5.0
BUZZER	-	-	-	-	-	-	-	BUZZER
PWM	PWM5	PWM4	PWM3	PWM2	PWM1	-	-	-
Timer	T5	T4	T3	T2	T1	T0	EC0	-

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่างๆของ Port P5

Port P6

โดยสัญญาณจาก Port P6 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ ทั้ง GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P6[7..0])
- ใช้งานเป็น Main Crystal Oscillator(P6[3..2])
- ใช้งานเป็น Timer Input Function(P6[5,4,1,0])

Port	P6.7	P6.6	P6.5	P6.4	P6.3	P6.2	P6.1	P6.0
GPIO	I/O P6.7	I/O P6.6	I/O P6.5	I/O P6.4	I/O P6.3	I/O P6.2	I/O P6.1	I/O P6.0
XTAL	-	-	-	-	XOUT	XIN	-	-
Timer	-	-	EC5	EC4	-	-	EC3	EC2

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่าง ๆของ Port P6

Port P7

โดยสัญญาณจาก Port P7 ของ Z51F6412 สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ ทั้งแบบ GPIO Input/Output ปรกติ และใช้งานร่วมกับฟังก์ชั่นพิเศษต่างๆ ดังนี้

- ใช้งานเป็น GPIO Input/Output Port(P7[7..0])
- ใช้งานเป็นขาตรวจจับการเปลี่ยนแปลง Input (Input Pin Change Interrupt : PCI7[7..0])

Port	P7.7	P7.6	P7.5	P7.4	P7.3	P7.2	P7.1	P7.0
GPIO	I/O P7.7	I/O P7.6	I/O P7.5	I/O P7.4	I/O P7.3	I/O P7.2	I/O P7.1	I/O P7.0
PCI7	PCI7.7	PCI7.6	PCI7.5	PCI7.4	PCI7.3	PCI7.2	PCI7.1	PCI7.0

ตาราง แสดงหน้าที่การใช้งาน Pin Port ต่าง ๆของ Port P7

ตัวอย่างการใช้งาน Input Port

```
#include <z51f6412.h>                      // Z51F6412 Register

/***********************************/
/* ET-BASE Z51F6412 Hardware SW Pin */
/***********************************/
#define SW_PIN           (1 << 0)          // P8[0] = SW Pin
#define SW_PORT_DIR      P8IO              // Port P8 Direction
#define SW_PORT_DATA     P8                // Port P8 Data
#define SW_PORT_PULLUP   P8PU              // Port P8 Pull-Up
#define SW_PORT_DEBOUNCE P8DB              // Port P8 Debounce

static bit this_sw;
static bit last_sw;
sbit   SW_READ = P8^0;                     // Pin Read SW = P8.0

void main(void)
{
  ...
  SW_PORT_DIR       &= ~(SW_PIN);          // Config SW Pin = Input
  SW_PORT_DEBOUNCE  |=  (SW_PIN);          // Enable Debounce For SW Pin
  SW_PORT_PULLUP    |=  (SW_PIN);          // Enable Pull-Up For SW Pin
  SW_PORT_DATA      |=  (SW_PIN);          // Defualt Logic "1"

  last_sw = 1;                             // Default Status = Release

  while(1)
  {
    this_sw = SW_READ;                     // Read Bit SW
    if(this_sw != last_sw)                 // If SW Status Change
    {

      if((last_sw==1)&&(this_sw==0))       //Verify SW Press & Service
      {
       ...
      }

      if((last_sw==0)&&(this_sw==1))       //Verify SW Release & Service
      {
       ...
      }

      last_sw = this_sw;                   //Update SW Reference Status
    }
  }
}
```

แสดง ตัวอย่าง Code สำหรับใช้งาน Input Pin

ตัวอย่างการใช้งาน Output Port

```
#include <z51f6412.h>                          // Z8051 : Z51F6412

/*************************************/
/* ET-BASE Z51F6412 Hardware LED Pin */
/*************************************/
#define LED_PIN               (1 << 1)         // P8[1] = LED
#define LED_PORT_DIR          P8IO             // Port P8 Direction
#define LED_PORT_DATA         P8               // Port P8 Data
.
.
.
void main(void)
{
  .
  .
  .
  LED_PORT_DIR |= (LED_PIN);                   // Configure LED =
Output
  …
  LED_PORT_DATA |=  (LED_PIN)                  // LED Pin = 1(ON LED)
  …
  LED_PORT_DATA &= ~(LED_PIN)                  // LED Pin = 0(OFF LED)
  …
  LED_PORT_DATA ^=  (LED_PIN)                  // LED Pin = Toggle
  .
  .
  .
}
```

แสดง ตัวอย่าง Code สำหรับ ใช้งาน Output Pin