+-----+
| MCU | การใช้งาน port เพื่อขับ seven segment เบื้องต้น(1)
| U03 | โดย อ.กำธร เรือนฝายกาศ
+-----+

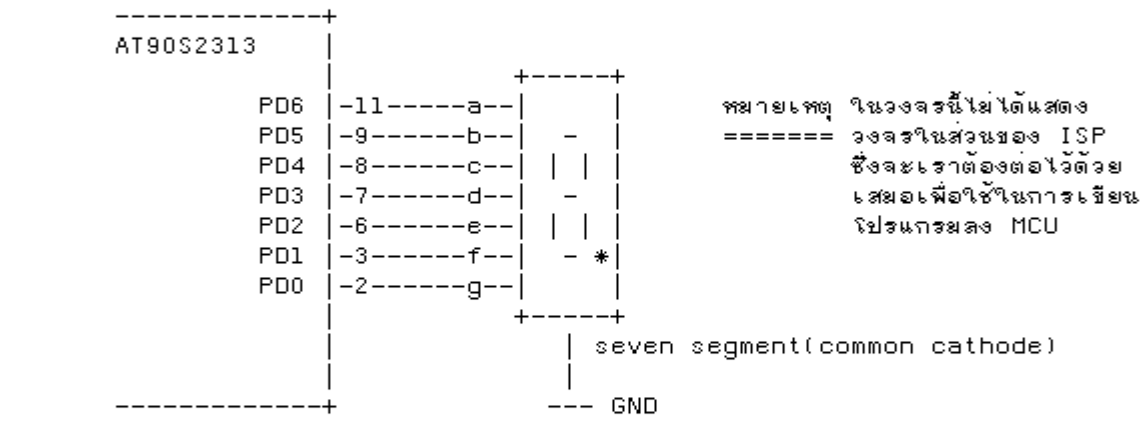
(1) กล่าวโดยทั่วไป

AT90S2313 มีพอร์ตสำหรับใช้งานติดต่อกับอุปกรณ์ภายนอก จำนวน 2 พอร์ต คือ PORTB และ PORTD โดย PORTD จะมีขาสำหรับต่อใช้งานจำนวน 7 ขา และ PORTB มีจำนวนขาสำหรับต่อใช้งานจำนวน 8 ขา แต่ละขา ของพอร์ตโปรแกรมให้เป็นอินพุต หรือเอ้าต์พุตก็ได้โดยอิสระจากกัน โดยการควบคุมของ DDRB และ DDRD ซึ่งย่อมาจากคำว่า DDRD = Data Directional Register port D และ DDRB = Data Directional Register port B

(2) การทดลองที่ 1

เป็นการใช้งาน PORTD และ PORTB เบื้องต้น เพื่อแสดงตัวเลขออกทาง seven segment แบบคอมมอนคาโธด

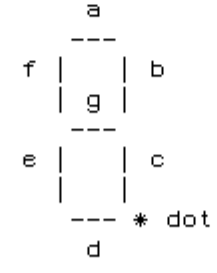
(3) วงจร



(4) รายการอุปกรณ์

seven segment แบบ commnon cathode	1	ตัว
AT90S2313	1	ตัว

(5) การจัดวางแต่ละ segment ของ seven segment



(6) เขียนโปรแกรมดังนี้

```
;***************************************************************************
; AT90S2313
; 1 seven segment
; copyright (c) 2001 by Gumtorn Ruanfaigad.Allright reserved.
; XTAL  = 4.0MHz
; MCU   = AT90S2313,Atmel corperation,www.atmel.com
;***************************************************************************

.include        "2313def.inc"           ;000
.DEVICE         AT90S2313               ;001
.CSEG                                   ;002
.ORG            0x0000                  ;003
                                        ;004 ======================
                                        ;005 interrupt vector table
                                        ;006 ======================
        rjmp    F_main                  ;004 #0 RESET
        reti                            ;005 #1 INT0
        reti                            ;006 #2 INT1
        reti                            ;007 #3 TIMER1 CAPT11
        reti                            ;008 #4 TIMER1 COMP1
        reti                            ;009 #5 TIMER1 OVF1
        reti                            ;010 #6 TIMER0 OVF0
        reti                            ;011 #7 UART,RX
        reti                            ;012 #8 UART,UDRE
        reti                            ;013 #9 UART,TX
        reti                            ;014 #10 ANA_COMP

;=============================================================================
;Function       OutSegment1
;{rin}          r16 = seven segment code
;{rout}         none
;{port}         PORTD[x6543210] (bits)
;                    [xabcdefg] (segment)
;
;         AT90S2313 |
;                   |           +-----+
;               PD6 |-11-----a--|     |
;               PD5 |-9------b--|  -  |
;               PD4 |-8------c--| | | |
;               PD3 |-7------d--|  -  |
;               PD2 |-6------e--| | | |
;               PD1 |-3------f--|  - *|
;               PD0 |-2------g--|     |
;                   |           +-----+
;                   |              | seven segment(common cathode)
;                   |              |
;        -----------+               --- GND
;=========================================
F_OutSeg1:                              ;105 function start here
        ldi     r17,    0b11111111      ;106 all port pin of PORTD = output
        out     DDRD,   r17             ;107 -
        out     PORTD,  r16             ;108 -
        ret                             ;109 function call return
;.............................................................................
```

```
;***************************************************************************
;void main(void)
;***************************************
F_Main:                                 ;202 main function start here
        ldi     r16,    low(RAMEND)     ;203 set stack pointer
        out     SPL,    r16             ;204 -
L_MainLoop:                             ;205 main loop
        ldi     r16,    0b00110000      ;206 display number "1"
        rcall           F_OutSeg1       ;207 function call
        rjmp            L_MainLoop      ;208 infinite loop
;.....................................................................
```

(8) บันทึกไฟล์โดย

1. เลือกเมนู File>>Save
2. เก็บไฟล์ไว้ในไดเรคทอรี่ c:\avrtools\appnotes โดยใช้ชื่อ outseg1.asm

(9) ออกไป DOS เพื่อทำการ assemble โดย

1. เลือกเมนู File>>DOS shell หรือกดคีย์ F7
2. ตอนนี้เราจะออกไปอยู่ที่ดอส ให้เรียกโปรแกรม avrasm มาทำงาน
   <> พิมพ์คำสั่งว่า

```
avrasm outseg1.asm outseg1.lst outseg1.hex <-:
```

   <> สัญญลักษณ์ <-: หมายถึงให้กดปุ่ม Enter บนคีย์บอร์ด จากนั้นโปรแกรมก็ทำการ assembling จะปรากฏข้อความ

```
C:\MYAVR avrasm outseg1.asm outseg1.lst outseg1.hex
AVRASM: AVR macro assembler version 1.21  (May  5 1998 01:21:00)
Copyright (C) 1995-1997 ATMEL Corporation

Creating   'outseg1.eep'
Creating   'outseg1.hex'
Creating   'outseg1.obj'
Creating   'outseg1.lst'

Assembling 'outseg1.asm'
Including  '2313def.inc'

Program memory usage:
Code              :   18 words
Constants (dw/db):    0 words
Unused            :    0 words
Total             :   18 words

Assembly complete with no errors.
Deleting   'outseg1.eep'
```

   <> กลับมาเข้าโปรแกรม edtv โดยการพิมพ์คำสั่ง

```
exit <-:
```

อธิบายการทำงานของ avrasm

```
Creating   'outseg1.eep' สร้าง epp file
Creating   'outseg1.hex' สร้าง hex file
Creating   'outseg1.obj' สร้าง object file
Creating   'outseg1.lst' สร้าง list file
```

```
Assembling 'outseg1.asm' ทำการแปลโปรแกรมที่เราเขียนขึ้นมา
Including  '2313def.inc' ผนวกไฟล์ 2313def.inc ตามที่เราสั่งไว้
                         ในไฟล์ outseg1.asm

Program memory usage:          รายงานการใช้หน่วยความจำของ MCU
Code              :   20 words   ใช้สำหรับโปรแกรม 20 เวิร์ด
Constants (dw/db):    0 words   ใช้สำหรับค่าคงที่ 0 เวิร์ด
Unused            :    0 words   -
Total             :   20 words   รวม 20 เวิร์ด

Assembly complete with no errors ไม่พบข้อผิดพลาด
Deleting   'outseg1.eep'          ลบไฟล์ outseg1.eep ทิ้ง
                                  เพราะมีไฟล์ outseg1.hex แล้ว
```

<> ใช้คำสั่ง dir outseg1.* <-:

```
C:\MYAVR>dir outseg1.*

Volume in drive C is GMT
Volume Serial Number is 114E-0FD6
Directory of C:\MYAVR

OUTSEG1  ASM         2,197  08-06-01 10:10a
OUTSEG1  LST         9,780  08-06-01 11:02a
OUTSEG1  OBJ           213  08-06-01 11:02a
OUTSEG1  HEX           124  08-06-01 11:02a
4 file(s)         12,314 bytes
0 dir(s)     277,839,872 bytes free
```

จะพบว่าโปรแกรม avrasm ได้สร้างไฟล์ขึ้นมาใหม่อีก 3 ไฟล์ คือ

```
outseg1.lst     List file
outseg1.obj     Object file ใช้สำหรับการ simulate โปรแกรม
outseg1.hex     Hex file ใช้สำหรับเขียน(write)ลงบนตัว MCU
```

<> เปิดดูไฟล์ outseg1.hex ด้วย editv จะได้ดังนี้

```
:100000000EC018951895189518951895189518951895189567
:10001000189518951895189518951FEF11BB02BB089506E0BF
:04002000FADFFDCF37
:00000001FF
```

ที่เห็นนี้คือโค้ดที่จะเขียนลงบนตัวไอซี MCU

(10) ออกไปดอส เพื่อเขียนโปรแกรมลงบนไอซี MCU ด้วยโปรแกรม SP12 ใช้คำสั่ง

```
sp12 -wpcf outseg1.hex <-:
```

จะได้ผลลัพธ์ดังนี้

```
C:\MYAVR>sp12 -wpfc outseg1.hex
SP12 version 1.0.2 performing init...
rcPath: _sp12rc
Running in SP12 cable/dongle compatible mode.
The character `c' is not a valid option.
Enabling AVR serial reading/programming...
The device code bytes 0,1,2: 0x1e, 0x91, 0x1 were read
from parallel port 0x378 and indicate the following:
You have connected an AT90S2313
The device was made by Atmel
```

```
Performing chip erase...
Writing content of outseg1.hex into program area.
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
outseg1.hex written and verified.
Write retries: 0, byteWrite: 16 percent of default
```

ตอนนี้ seven segment ยังไม่สว่าง เพราะขา reset ของไอซีที่ต่อกับ
พอร์ตขนานยังเป็นลอจิก 0 ทำให้ MCU ไม่ทำงาน ดังนั้นเราต้องสั่งให้
พอร์ตขนานที่ต่ออยู่กับขา reset ของ MCU เป็นลอจิก 1 ด้วยคำสั่ง

```
spl2 -P2  <-:
```

คราวนี้จะเห็นว่า seven segment สว่าง แสดงตัวเลข 1 ข้อสำคัญเวลาเขียน
คำสั่งตัวอักษร ตัว P ต้องเป็นตัวพิมพ์ใหญ่ ไม่ใช่ตัว p เล็ก

(11) อธิบายการทำงานของโปรแกรม

```
.....................................................................

        .include        "2313def.inc"           ;000

        <> นำไฟล์ชื่อ 2313def.inc มาผนวก(include) เข้ากับโปรแกรมที่เราสร้าง
        ขึ้นซึ่งเราจะต้อง include ไฟล์นี้เสมอ ถ้าหากใช้ไอซี MCU เบอร์อื่น ก็จะต้องเลือก
        ไฟล์ include ของไอซีเบอร์นั้นๆ
.....................................................................

        .DEVICE         AT90S2313               ;001

        <> คำสั่งนี้เป็นการบอกให้ assembler ทราบว่าเราต้องการให้ assembler
        ทำการ assemble ไฟล์สำหรับไอซีเบอร์ AT90S2313 ถ้าใช้เบอร์อื่นเราก็ต้อง
        เปลี่ยนเบอร์ให้ถูกต้อง
.....................................................................

        .CSEG                                   ;002

        <> เป็นคำสั่งบอกให้ assembler ทราบว่าตั้งแต่จุดนี้เป็นต้นไป จะเป็นส่วนของ
        โปรแกรม(code segment) โปรแกรมจะถูกนำไปไว้ในส่วนของ flash memory
        ซึ่งสำหรับไอซีเบอร์ AT90S2313 จะมีเท่ากับ 1 กิโลเวิร์ด(1024 เวิร์ด) คำสั่ง
        ของไอซีเบอร์นี้ หนึ่งคำสั่งจะใช้ที่เกิน 1,2 หรือ 3 เวิร์ด นั่นหมายถึง เราจะเขียน
        โปรแกรมได้หนึ่งพันคำสั่งโดยประมาณ
.....................................................................

        .ORG            0x0000                  ;003

        <> เป็นคำสั่งเพื่อให้ code segment ที่เรากำหนดในบรรทัด 002 มีจุดเริ่มต้นที่
        ตำแหน่งเวิร์ด 0
.....................................................................

                                        ;004 ======================
                                        ;005 interrupt vector table
                                        ;006 ======================
        rjmp    F_main                  ;004 #0 RESET
        reti                            ;005 #1 INT0
        reti                            ;006 #2 INT1
        reti                            ;007 #3 TIMER1 CAPT11
        reti                            ;008 #4 TIMER1 COMP1
        reti                            ;009 #5 TIMER1 OVF1
        reti                            ;010 #6 TIMER0 OVF0
        reti                            ;011 #7 UART,RX
```

```
        reti                            ;012 #8 UART,UDRE
        reti                            ;013 #9 UART,TX
        reti                            ;014 #10 ANA_COMP
```

<> โปรแกรมในส่วนนี้จะเป็นการกำหนด ตารางอินเตอร์รัพท์ ของอินเตอร์รัพท์แต่ละหมายเลข เริ่มจากอินเตอร์รัพท์หมายเลข 0 (RESET) เป็นอินเตอร์รัพท์ที่จะเกิดขึ้นเองทุกครั้งที่ไอซี MCU เกิดการรีเซตขึ้น ซึ่งตอนนี้ถูกตั้งให้ชี้ให้กระโดด(jump) ไปที่ฟังค์ชั่น F_main ซึ่งเป็นส่วนของฟังค์ชั่นหลักที่จะทำงานทุกครั้งที่ MCU เริ่มทำงาน สำหรับอินเตอร์รัพท์เบอร์ 1 ถึง เบอร์ 10 นั้นตอนนี้ยังไม่ใช้จึงเขียนคำสั่ง reti ไว้ ทำให้เมื่อเกิดอินเตอร์รัพท์ก็ MCU ก็จะกลับไป ไม่เกิดการกระโดดไปทำงานเหมือนกับอินเตอร์รัพท์ 0 ที่เราตั้งไว้ (reti คือ interrupt return)

.....

```
;=====================================================================
;Function       OutSegment1
;(rin)          r16 = sevensegment code
;(rout)         none
;(port)         PORTD[x6543210] port pin
;                    [xabcdefg] seven segment mapping
;=======================================
```

<> ในส่วนนี้จะเป็นฟังค์ชั่นสำหรับแสดงตัวเลขบน seven segment จำนวนหนึ่งตัว ชื่อฟังค์ชั่น OutSegment1 มี rin คือ input register ที่ใช้งานคือ r16 ซึ่งจะใช้เป็นที่ส่งตัวเลขที่ต้องการแสดงผล เข้ามายังฟังค์ชั่นนี้ เช่นถ้าต้องการแสดงเลข 1 ก็เขียนคำสั่งดังนี้

```
        ldi     r16,    0b00110000 ; บิต PD6,PD5 เป็น 1 เพื่อให้
        rcall           F_OutSeg1  ; ∫ segment b,c สว่าง
```

ไอซี MCU ก็จะเข้ามาเริ่มทำงานที่ F_OutSeg1 โดยนำข้อมูลคือเลข 3 เข้ามาด้วย สำหรับ rout คือ output register นั้นไม่มีการใช้งาน เพราะฟังค์ชั่นนี้ไม่ส่งข้อมูลกลับไปยังผู้เรียก(caller) และในส่วนของ port ในที่นี้จะใช้พอร์ตจำนวน 1 พอร์ต คือ พอร์ต D ซึ่งมีจำนวน 7 บิต เราจะใช้ทั้ง 7 บิต เพื่อขับแต่ละ segment ของ seven segment โดยขา common cathode ของ seven segment จะถูกต่อกับกราวด์ เพื่อให้กระแสไฟฟ้าไหลครบวงจร

.....

```
F_OutSeg1:                              ;105 function start here
```

<> เป็น Lebel บอกทางเข้า(Entry point) ของฟังค์ชั่น ซึ่งเราอาจจะใช้คำอื่นใดก็ได้ถ้าไม่ผิดกฎการตั้งชื่อ Label ของ Assembler(ดูได้จากภาคผนวก) สำหรับวิธีการที่ผมใช้ และคิดว่าน่าจะเป็นอีกวิธีหนึ่งที่ดีก็คือ

1) ขึ้นต้นด้วยตัว
   - `F` ใช้สำหรับ function(routine) เรียกใช้ด้วยคำสั่ง rcall
   - `FS` ใช้สำหรับ subfunction ของ function เรียกใช้ด้วยคำสั่ง rcall
   - `L` ใช้สำหรับ label เพื่อการ jump เรียกใช้ด้วยคำสั่ง rjmp
   - `I` ใช้สำหรับ Interrupt routine
2) ตามด้วยอักขระ '_' และชื่อของฟังค์ชั่นซึ่งควรเขียนให้สื่อความหมายที่ชัดเจน และวินหลังกลับมาอ่านแล้วยังเข้าใจว่าชื่อนี้หมายถึงอะไร

การตั้งชื่อเลเบล แบบนี้จะทำให้การตรวจสอบโปรแกรมทำได้ง่าย เพราะเลเบลใดที่ขึ้นต้นด้วย F เช่น F_OutSeg1 นี้จะหมายถึงฟังค์ชั่น ซึ่งจะต้องเรียกใช้ด้วยคำสั่ง rcall เท่านั้น ถ้าเราเห็นคำสั่งอย่างนี้

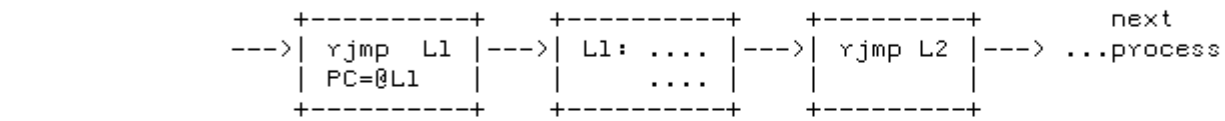
```
        rjmp    F_OutSeg1
```

แสดงว่าเขียนผิด ถึงแม้นว่าคำสั่ง rjmp นี้จะทำให้ MCU กระโดดไปทำงานตามที่เราสั่งก็ตาม แต่ตอนที่เสร็จงานในฟังค์ชั่นแล้ว ท้ายสุดของฟังค์ชั่นจะมีคำสั่ง ret ซึ่งจะสั่งให้ MCU ทำการดึงค่า PC (PC คือ Program Counter)ออกมาจาก stack เพื่อให้ MCU กลับไปทำคำสั่งในแอดเดรสตามค่าของ PC แต่อย่าลืมว่า ก่อนที่ MCU จะเข้ามาฟังค์ชั่นนี้เราเรียกฟังค์ชั่นด้วยคำสั่ง rjmp ซึ่ง MCU จะกระโดดเข้ามาเลยโดยที่ยังไม่ได้ PUSH ค่า PC ลง stack ก่อน ดังนั้นตอนที่ทำคำสั่ง ret ค่า PC ที่ POP ออกมาจาก stack จึงเป็นขยะ แล้วจะทำให้โปรแกรมของเรามีพฤติกรรมที่ไม่อาจคาดเดาได้ ดังนั้นการเรียกใช้งานฟังค์ชั่นจึงต้องเรียกด้วยคำสั่ง rcall เสมอ เช่น

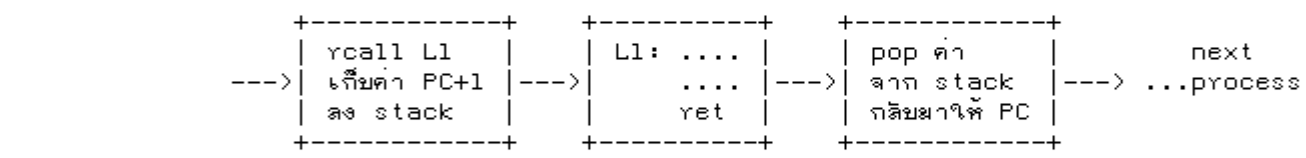
```
rcall   F_OutSeg1
```

อย่างนี้เป็นการเขียนคำสั่งที่ถูกต้อง เพราะเมื่อ MCU ทำคำสั่ง rcall ตัว MCU จะทำการ PUSH ค่า PC ลง stack ก่อนที่จะกระโดดไปทำฟังค์ชั่น F_OutSeg1 ลองดูการเปรียบเทียบระหว่าง rcall กับ rjmp จะเห็นว่าการสั่ง rcall แล้วเราควรจะใช้คู่กับคำสั่ง ret เสมอ การสั่ง rcall แล้วไม่ทำ ret จะทำให้มีข้อมูลคือค่า PC+1 ตกค้างอยู่ใน stack ซึ่งอาจจะทำให้โปรแกรมของเรามีพฤติกรรมที่ไม่อาจจะคาดเดาได้

rjmp
====



rcall
=====

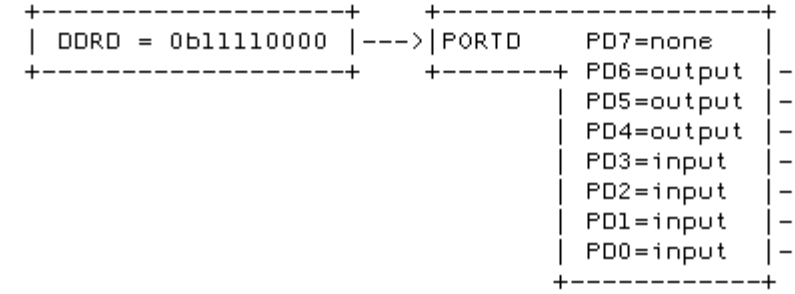


.....

```
ldi     r17,    0b11111111      ;106 all port pin of PORTD = output
out     DDRD,   r17             ;107 -
```

<> เป็นคำสั่งโหลดค่า 0xFF หรือ 0b11111111 เข้าไปเก็บไว้ในรีจิสเตอร์ แล้วโหลดเข้าไปเก็บไว้ในรีจิสเตอร์ DDRD(Data Direction Register for port D) ซึ่งเป็นรีจิสเตอร์กำหนดทิศทางข้อมูลดิจิตอลของพอร์ต D รีจิสเตอร์ DDRD เป็นรีจิสเตอร์ขนาด 8 บิต การแมป DDRD กับ PORTD จะเป็นแบบบิตต่อบิต ยกเว้นบิต 7 เพราะ PORTD มีเพียง 7 ขาเท่านั้น ถ้าบิตใดของรีจิสเตอร์ DDRD เป็น '1' ก็จะทำให้ขานั้นของ PORTD ทำงานเป็น output pin แต่ถ้ากำหนดให้เป็น '0' จะทำให้บิตนั้นของ PORTD ทำงานเป็น input pin

ตัวอย่าง ถ้ากำหนดให้ DDRD เท่ากับ 0b01110000
======



ทำให้ PD0,PD1,PD2 และ PD3 ทำงานเป็นอินพุต และ PD4,PD5 และ PD6 ทำงานเป็น
เอ้าต์พุต การจะกำหนดให้ขาใดของ PORTD เป็นอินพุตหรือเอ้าต์พุต ก็ขึ้นอยู่กับความ
ต้องการของเรา การกำหนดแบบนี้มีใน PORTB ด้วย โดยมี DDRB เป็นรีจิสเตอร์กำหนด
ทิศทางของ PORTB

```
.....................................................................

        out     PORTD,  r16             ;108 -

        <> เป็นคำสั่งให้ส่งค่าที่เข้ามาทางรีจิสเตอร์ r16 ออกทาง PORTD ไปขับ segent
        ทั้ง 7 ของ seven segment ถ้าบิตใดเป็นลอจิก 1 ก็จะทำให้ segment นั้นสว่าง และ
        จะกลับกันถ้าเป็นลอจิก 0
.....................................................................

        ret                             ;109 function call return

        <> เป็นคำสั่งให้ MCU กลับไปทำคำสั่งต่อจากคำสั่ง rcall ที่เรียกเข้ามายัง
        ฟังค์ชั่นนี้ ซึ่งในที่นี้ MCU จะทำคำสั่งบรรทัดที่ 204 คือ rjmp L_MainLoop เป็น
        คำสั่งถัดไป
.....................................................................

;=====================================================================
;void main(void)
;=======================================
F_Main:                                 ;202 main function entry point
        ldi     r16,    low(RAMEND)     ;203 set stack pointer
        out     SPL,    r16             ;204 -
L_MainLoop:                             ;205 main loop
        ldi     r16,    0b00110000      ;206 display number "1"
        rcall           F_OutSeg1       ;207 function call

        <> ส่วนนี้เป็นฟังค์ชั่นหลักมีเลเบลอยู่ที่จุดทางเข้าของฟังค์ชั่นคือ F_Main ซึ่ง
        MCU จะเริ่มต้นทำงานที่ฟังค์ชั่นนี้เป็นครั้งแรก ทุกครั้ง ด้วยอินเตอร์รัพท์ 0
        ที่เราได้ตั้งไว้นั่นเอง บรรทัดที่ 203,204 เป็นการตั้งค่า stack pointer
        ให้ชี้ไปยังแอดเดรสสูงสุดของ sram (static ram) จากนั้นจะทำคำสั่ง
        โหลดค่าที่ต้องการแสดงทาง seven segmentเข้าไปเก็บไว้ในรีจิสเตอร์ r16
        ซึ่งตอนนี้ตั้งให้แสดงเลข 1 จากนั้นทำคำสั่งrcall F_OutSeg1 เพื่อเรียก
        ไปยังฟังค์ชั่น OutSeg1 ให้ทำงาน
.....................................................................

        rjmp            L_MainLoop      ;208 infinite loop

        <> เมื่อทำฟังค์ชั่น OutSeg1 เสร็จแล้ว MCU ก็จะกลับมาทำคำสั่งนี้ต่อ คำสั่ง
        บรรทัดนี้สั่งให้การะโดดไปยังเลเบล L_MainLoop วนอยู่ตลอดไป
```