Easily 8 Bit Hexa Calculator

จัดทำโดย

นายขวัญชัย	กล่อมเมฆ	09601340
นายปวีณ	สุวรรณสถาน	09601416
นายวิศว	เกตุสุมา	09601464
นายศุภวิชญ์	บุรารักษ์	09601471

เสนอ

อาจารย์ ดร.ณพงศ์ ปณิธานธรรม

รายงานนี้เป็นส่วนหนึ่งของรายวิชาระบบคอมพิวเตอร์และสมองกลฝังตัว (618354-2560)

คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม สาขาวิชาอิเล็กทรอนิกส์และระบบคอมพิวเตอร์

มหาวิทยาลัยศิลปากร วิทยาเขตพระราชวังสนามจันทร์

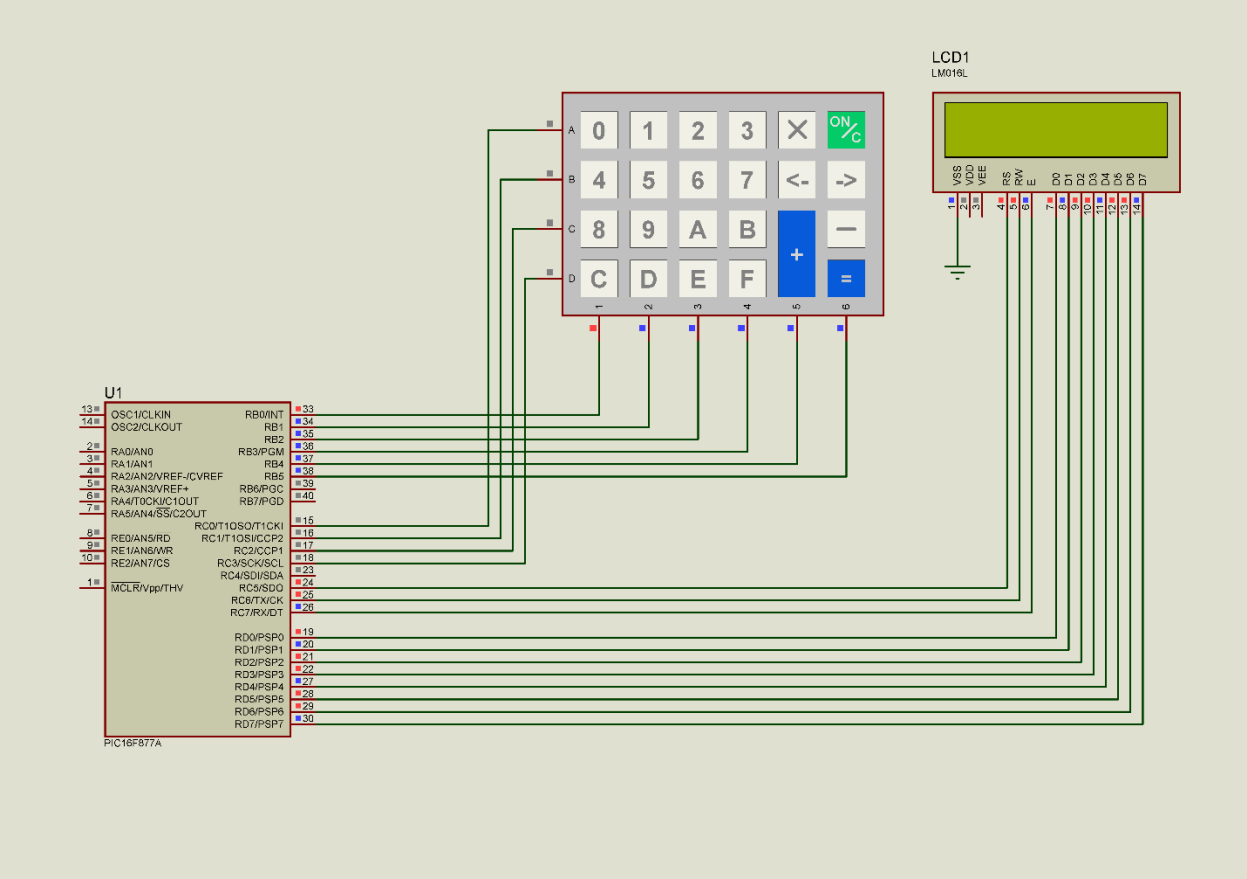
ขอบเขตโครงงาน

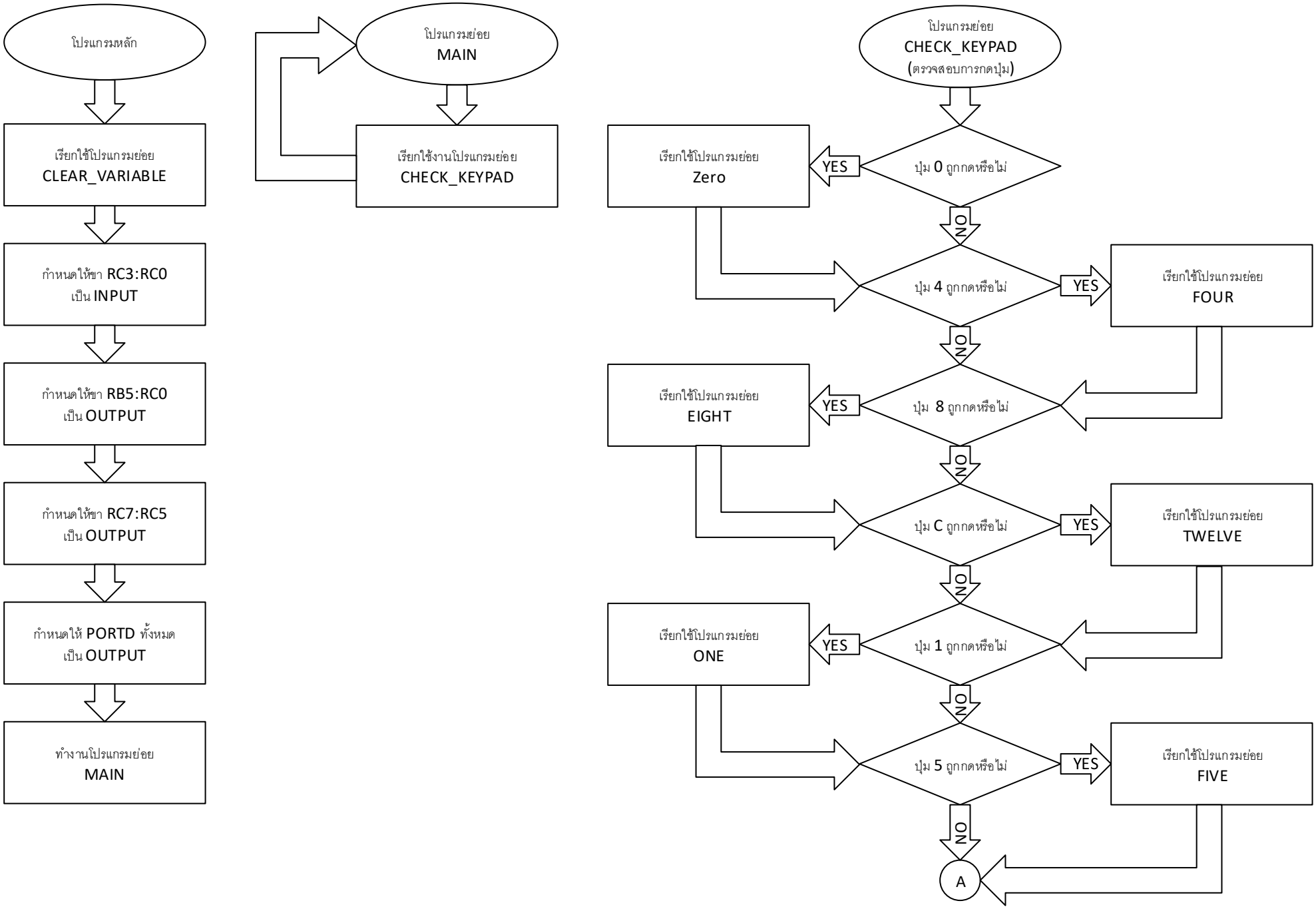
เครื่องคิดเลขฐาน 16 ขนาด 8 บิต โดยแสดงผลอยู่ในรูปของเลขฐาน 16 สูงสุดจำนวน 2 หลัก โดยมีฟังก์ชันบวก ลบ คูณ ชิฟซ้าย ชิฟขวาและออน/ซี

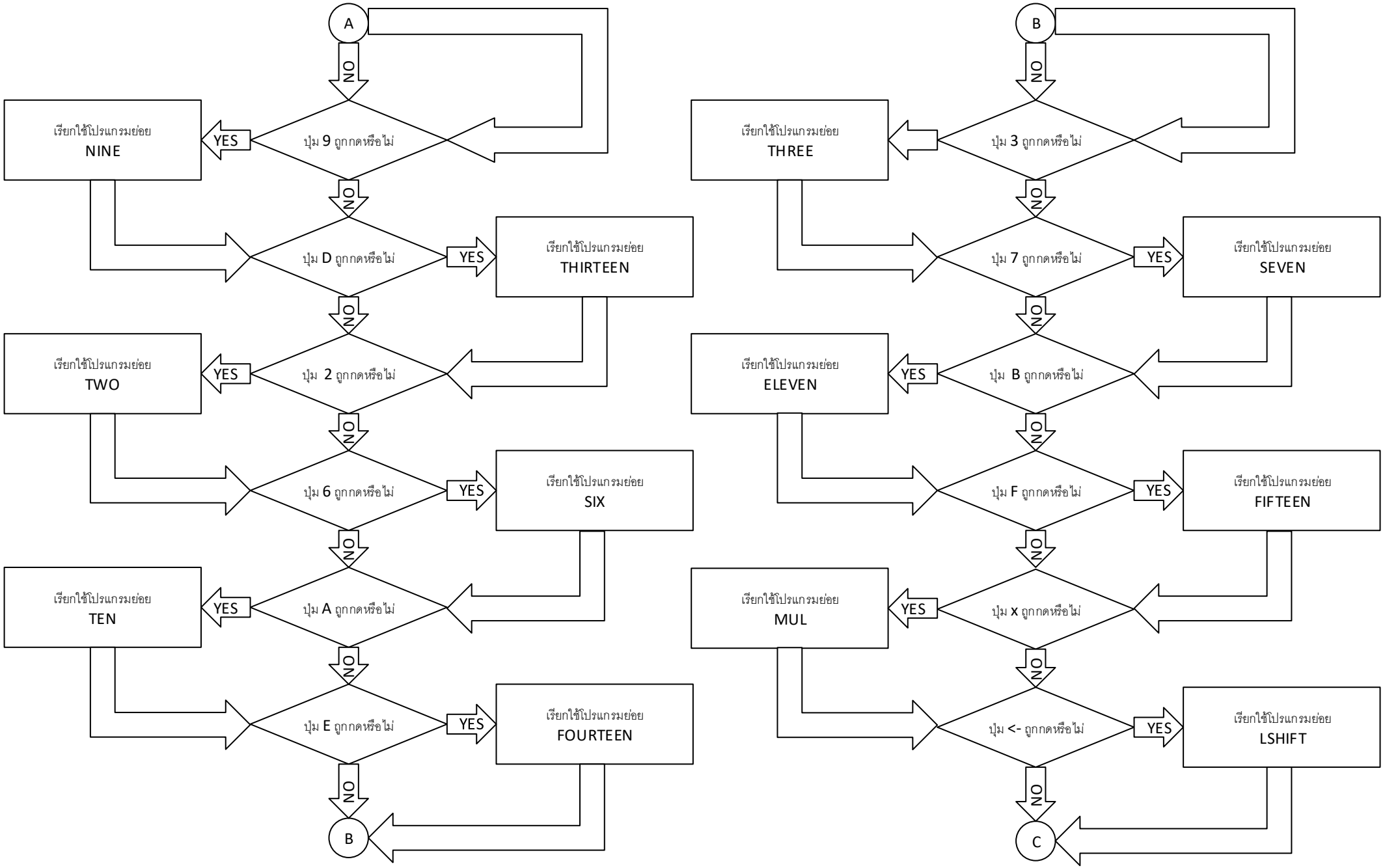
อุปกรณ์ (Hardware)

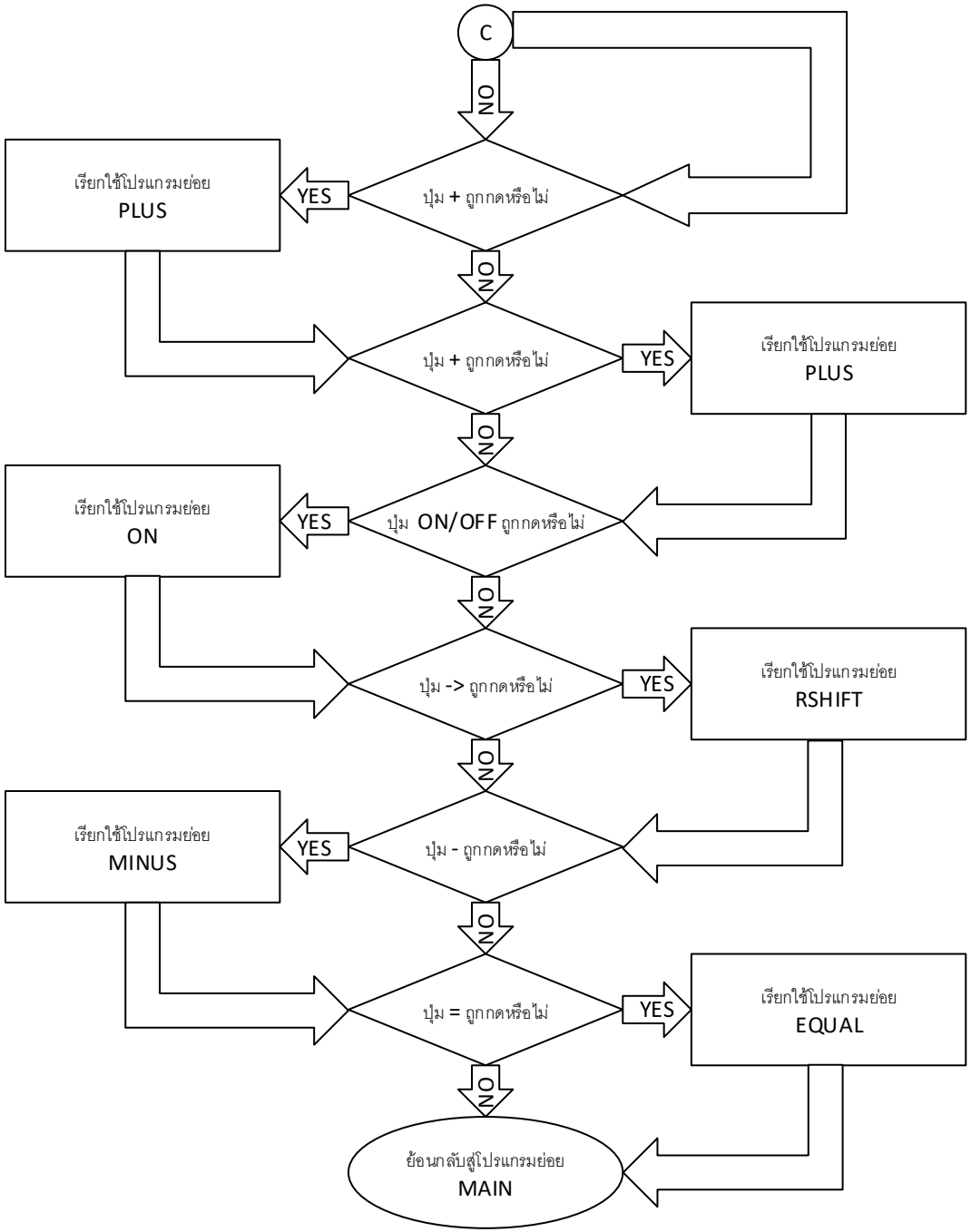
เครื่องคิดเลขฐาน 16 ขนาด 8 บิตประกอบไปด้วยอุปกรณ์ดังนี้

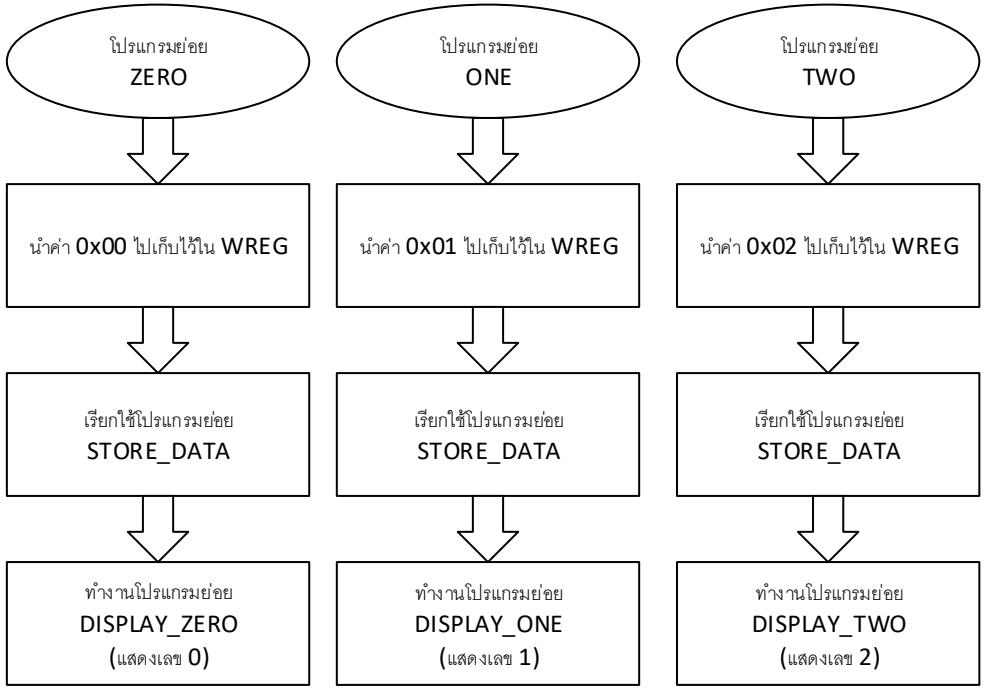
1. Microcontroller P16F877A
2. Keypad Hex Calculator
3. LCD LM016L

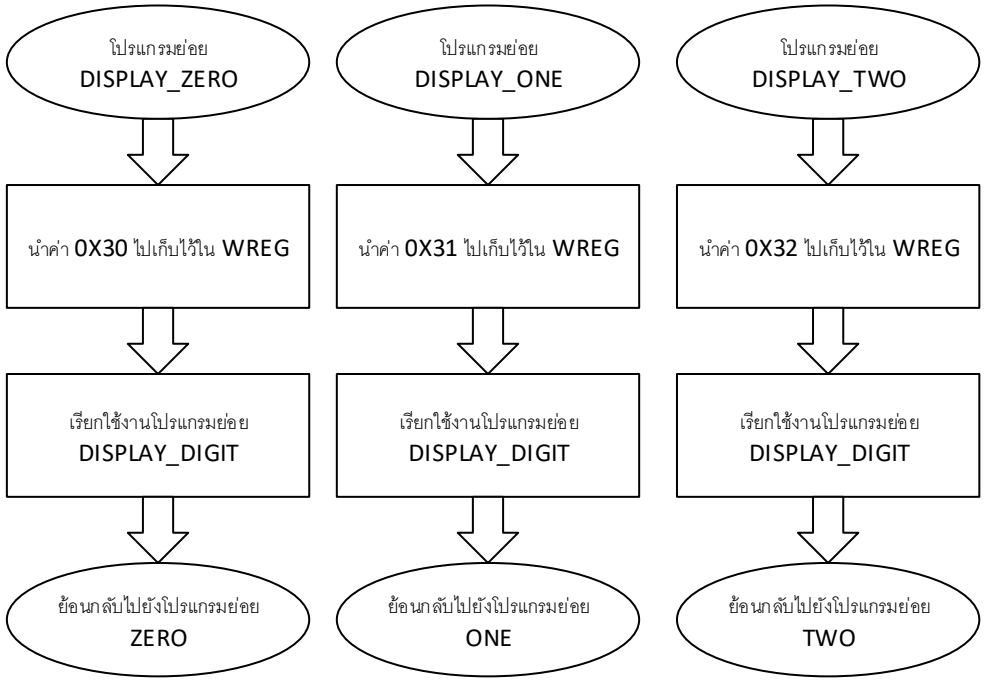


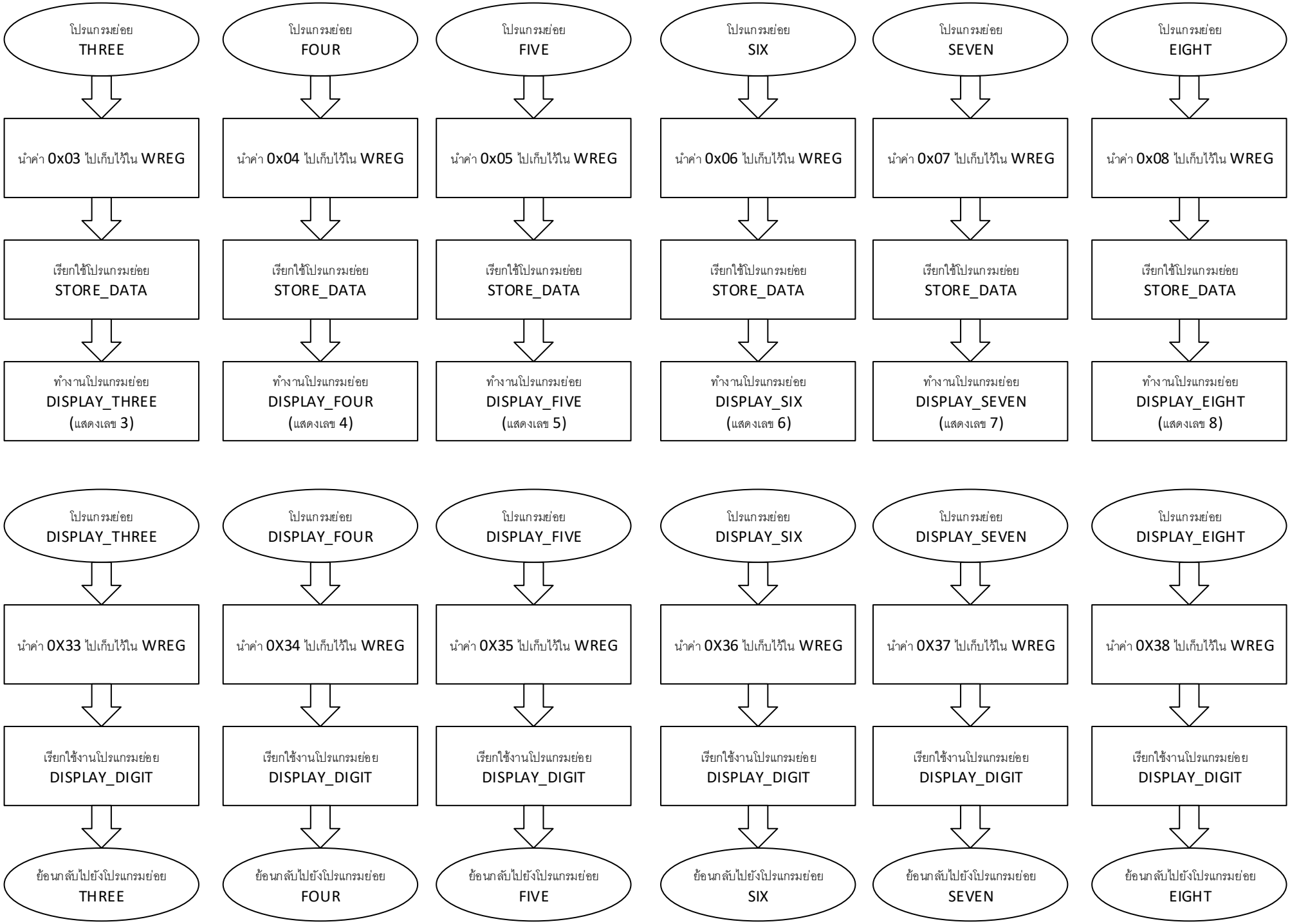

โปรแกรมหลัก
เรียกใช้โปรแกรมย่อย
CLEAR_VARIABLE
กำหนดให้ขา RC3:RC0
เป็น INPUT
กำหนดให้ขา RB5:RC0
เป็น OUTPUT
กำหนดให้ขา RC7:RC5
เป็น OUTPUT
กำหนดให้ PORTD ทั้งหมด
เป็น OUTPUT
ทำงานโปรแกรมย่อย
MAIN
โปรแกรมย่อย
MAIN
เรียกใช้งานโปรแกรมย่อย
CHECK_KEYPAD
โปรแกรมย่อย
CHECK_KEYPAD
(ตรวจสอบการกดปุ่ม)
ปุ่ม 0 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
Zero
NO
ปุ่ม 4 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
FOUR
NO
ปุ่ม 8 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
EIGHT
NO
ปุ่ม C ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
TWELVE
NO
ปุ่ม 1 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
ONE
NO
ปุ่ม 5 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
FIVE
NO
A

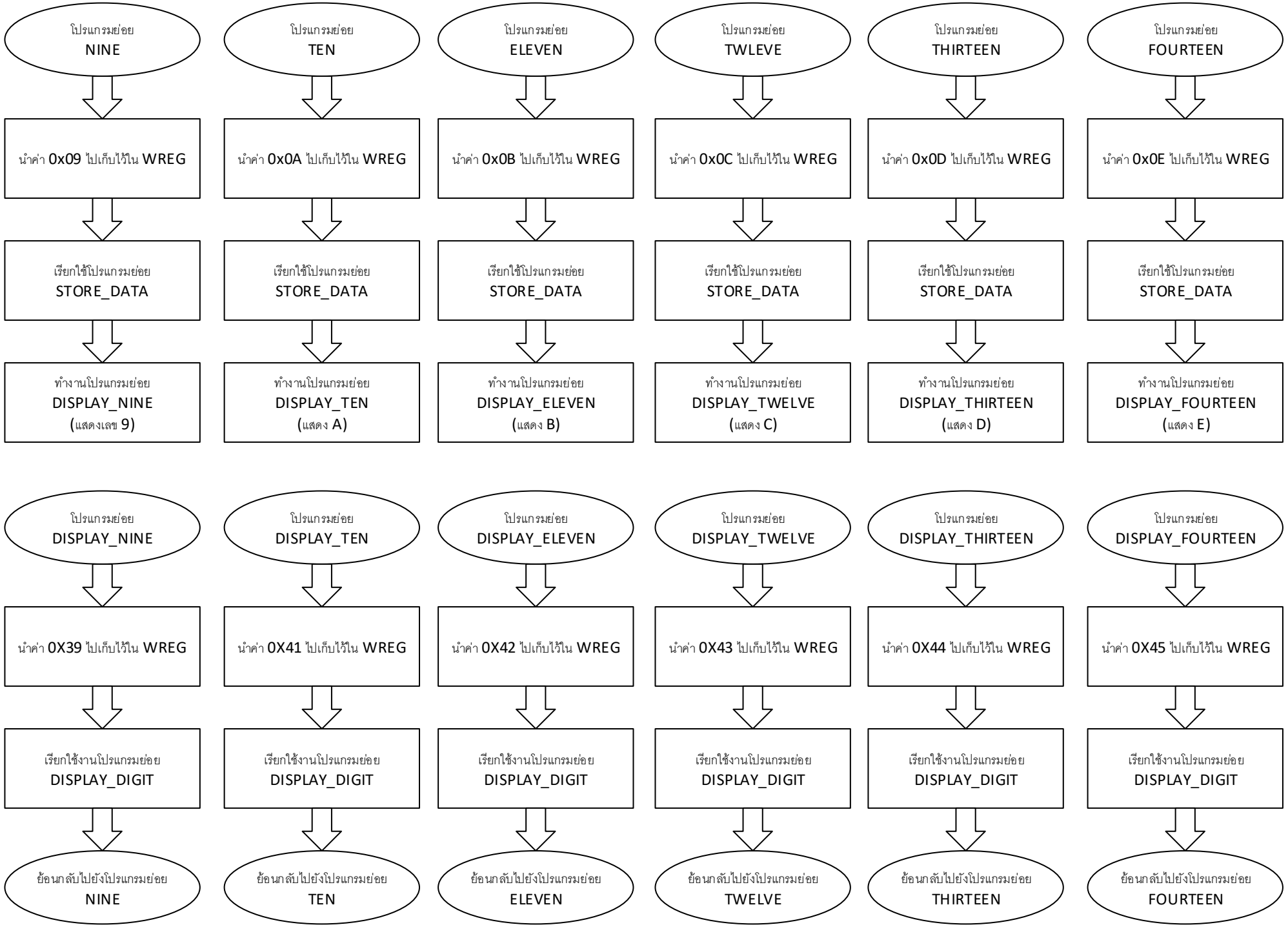

A
NO
ปุ่ม 9 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
NINE
NO
ปุ่ม D ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
THIRTEEN
NO
ปุ่ม 2 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
TWO
NO
ปุ่ม 6 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
SIX
NO
ปุ่ม A ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
TEN
NO
ปุ่ม E ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
FOURTEEN
NO
B
B
NO
ปุ่ม 3 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
THREE
NO
ปุ่ม 7 ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
SEVEN
NO
ปุ่ม B ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
ELEVEN
NO
ปุ่ม F ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
FIFTEEN
NO
ปุ่ม x ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
MUL
NO
ปุ่ม <- ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
LSHIFT
NO
C

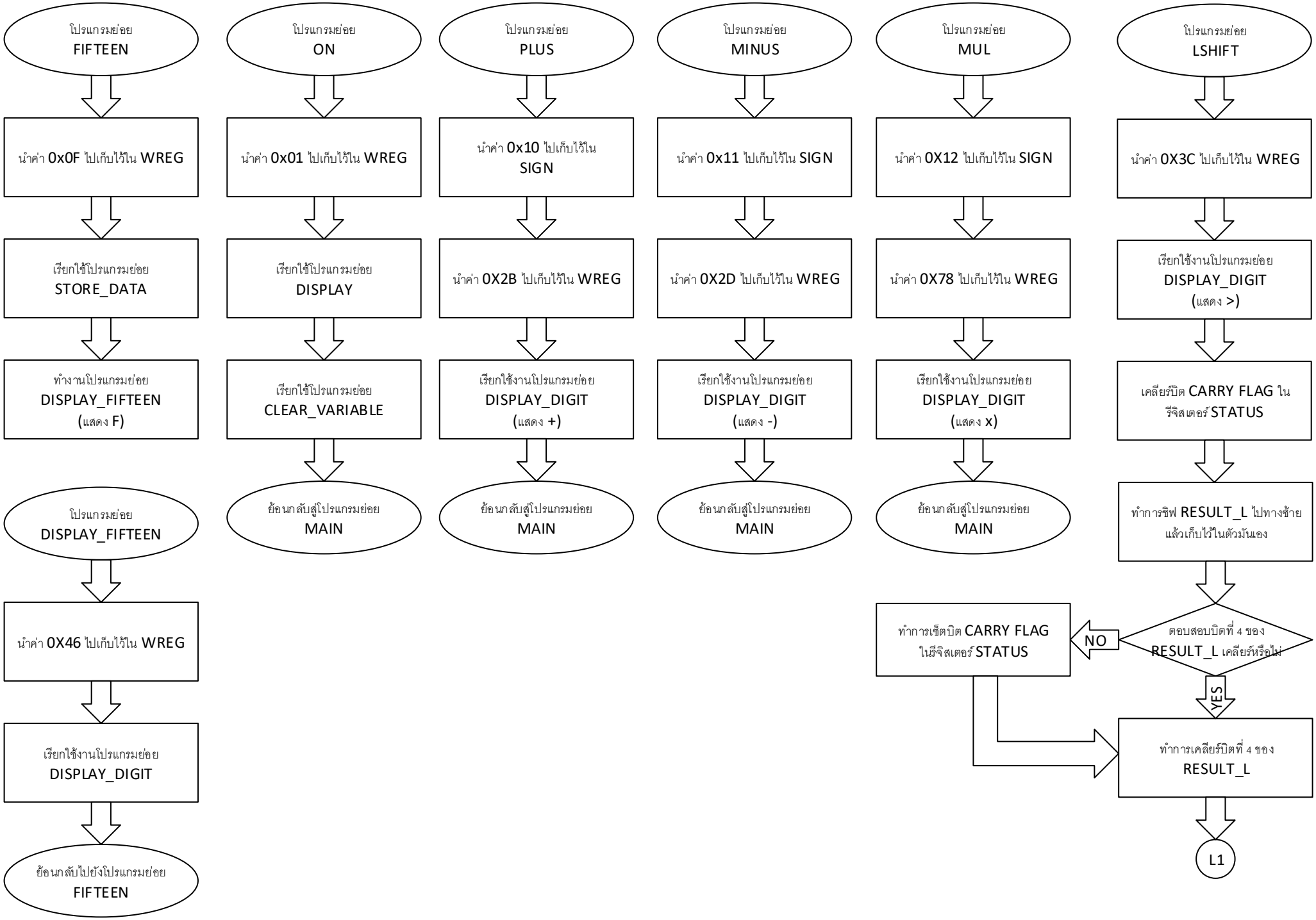

C
NO
ปุ่ม + ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
PLUS
NO
ปุ่ม + ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
PLUS
NO
ปุ่ม ON/OFF ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
ON
NO
ปุ่ม -> ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
RSHIFT
NO
ปุ่ม - ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
MINUS
NO
ปุ่ม = ถูกกดหรือไม่
YES
เรียกใช้โปรแกรมย่อย
EQUAL
NO
ย้อนกลับสู่โปรแกรมย่อย
MAIN

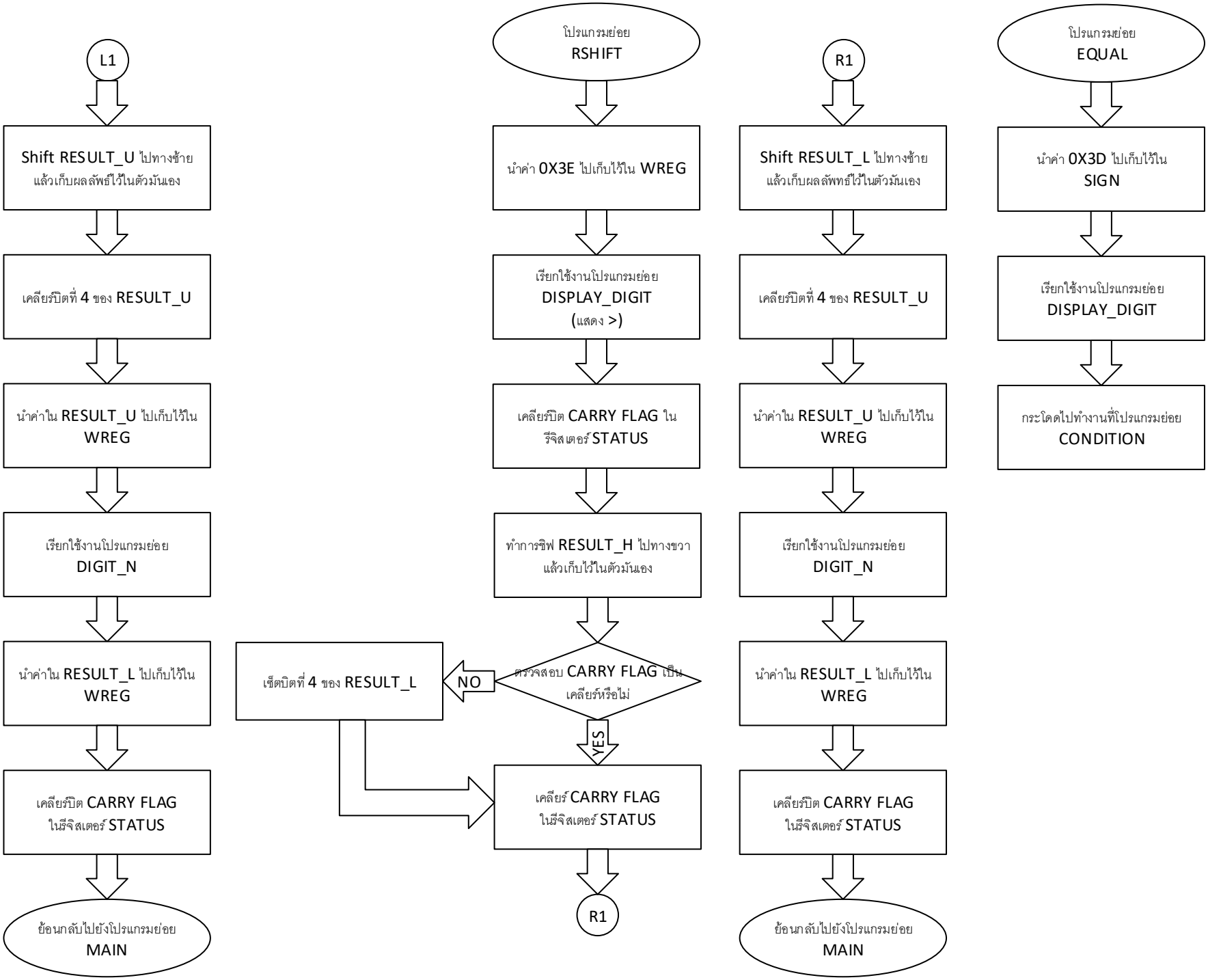

โปรแกรมย่อย
ZERO
นำค่า 0x00 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_ZERO
(แสดงเลข 0)
โปรแกรมย่อย
ONE
นำค่า 0x01 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_ONE
(แสดงเลข 1)
โปรแกรมย่อย
TWO
นำค่า 0x02 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_TWO
(แสดงเลข 2)

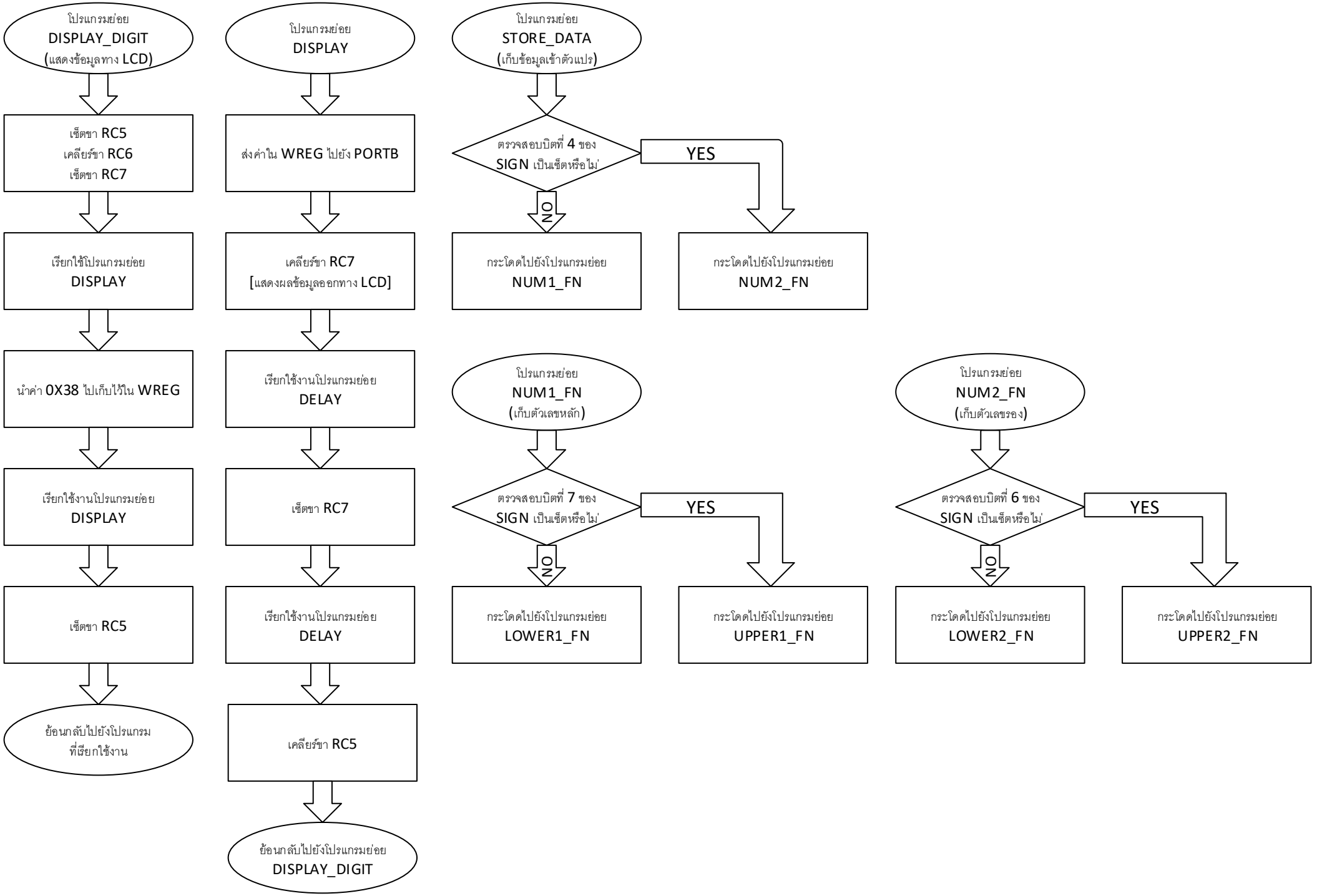

โปรแกรมย่อย
DISPLAY_ZERO
นำค่า 0X30 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
ZERO
โปรแกรมย่อย
DISPLAY_ONE
นำค่า 0X31 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
ONE
โปรแกรมย่อย
DISPLAY_TWO
นำค่า 0X32 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
TWO

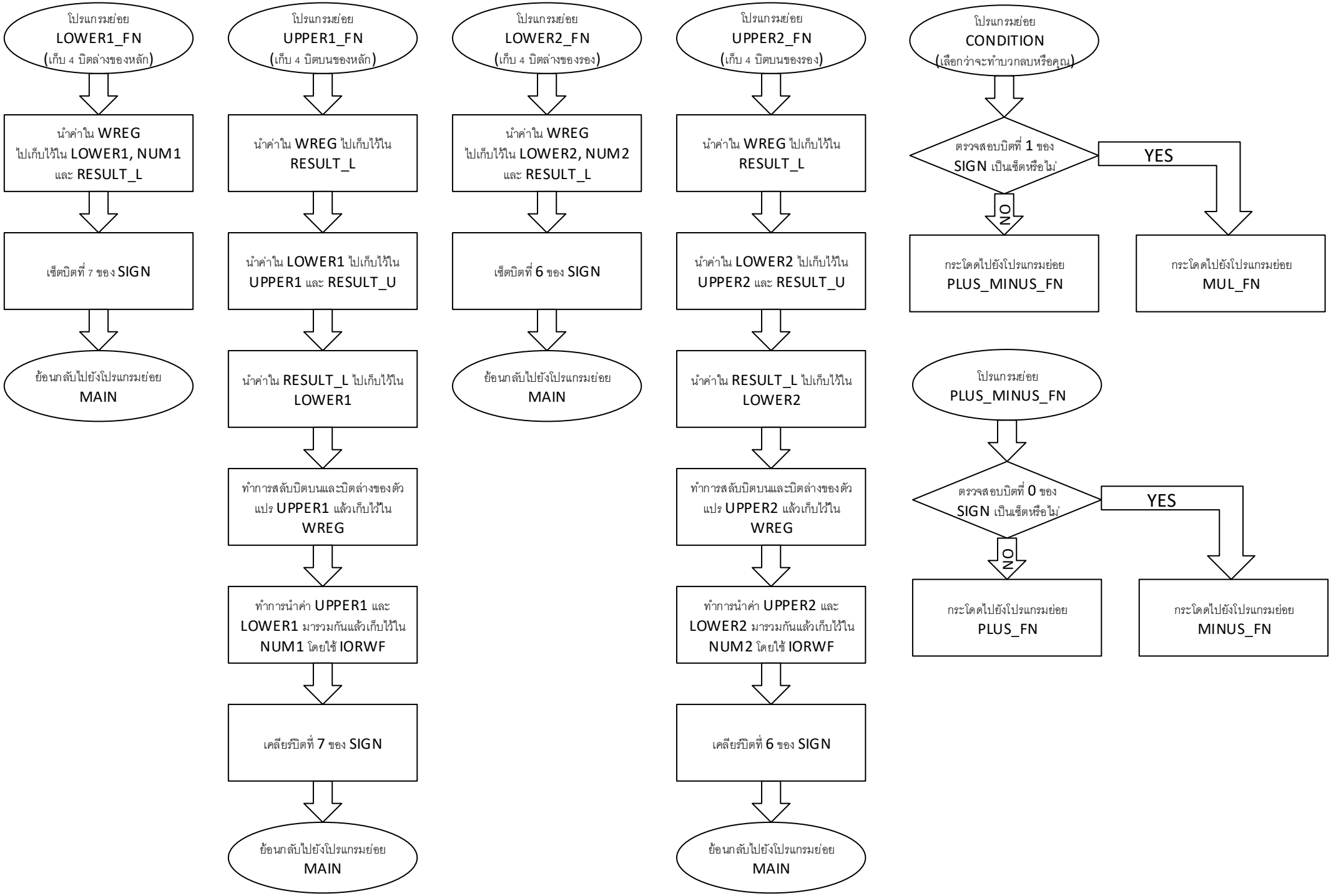

โปรแกรมย่อย THREE
นำค่า 0x03 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย STORE_DATA
ทำงานโปรแกรมย่อย DISPLAY_THREE (แสดงเลข 3)
โปรแกรมย่อย FOUR
นำค่า 0x04 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย STORE_DATA
ทำงานโปรแกรมย่อย DISPLAY_FOUR (แสดงเลข 4)
โปรแกรมย่อย FIVE
นำค่า 0x05 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย STORE_DATA
ทำงานโปรแกรมย่อย DISPLAY_FIVE (แสดงเลข 5)
โปรแกรมย่อย SIX
นำค่า 0x06 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย STORE_DATA
ทำงานโปรแกรมย่อย DISPLAY_SIX (แสดงเลข 6)
โปรแกรมย่อย SEVEN
นำค่า 0x07 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย STORE_DATA
ทำงานโปรแกรมย่อย DISPLAY_SEVEN (แสดงเลข 7)
โปรแกรมย่อย EIGHT
นำค่า 0x08 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย STORE_DATA
ทำงานโปรแกรมย่อย DISPLAY_EIGHT (แสดงเลข 8)
โปรแกรมย่อย DISPLAY_THREE
นำค่า 0X33 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย THREE
โปรแกรมย่อย DISPLAY_FOUR
นำค่า 0X34 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย FOUR
โปรแกรมย่อย DISPLAY_FIVE
นำค่า 0X35 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย FIVE
โปรแกรมย่อย DISPLAY_SIX
นำค่า 0X36 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย SIX
โปรแกรมย่อย DISPLAY_SEVEN
นำค่า 0X37 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย SEVEN
โปรแกรมย่อย DISPLAY_EIGHT
นำค่า 0X38 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย EIGHT

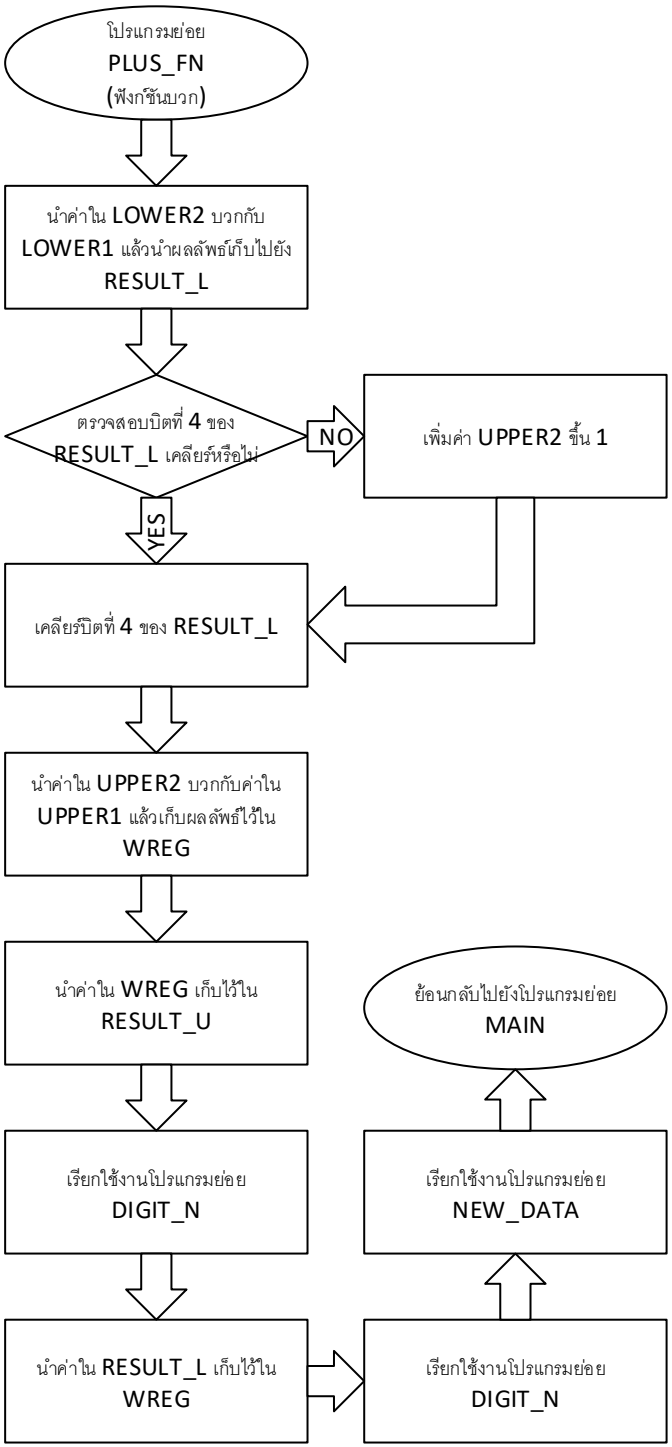

โปรแกรมย่อย
NINE
นำค่า 0x09 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_NINE
(แสดงเลข 9)
โปรแกรมย่อย
TEN
นำค่า 0x0A ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_TEN
(แสดง A)
โปรแกรมย่อย
ELEVEN
นำค่า 0x0B ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_ELEVEN
(แสดง B)
โปรแกรมย่อย
TWLEVE
นำค่า 0x0C ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_TWELVE
(แสดง C)
โปรแกรมย่อย
THIRTEEN
นำค่า 0x0D ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_THIRTEEN
(แสดง D)
โปรแกรมย่อย
FOURTEEN
นำค่า 0x0E ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย
STORE_DATA
ทำงานโปรแกรมย่อย
DISPLAY_FOURTEEN
(แสดง E)
โปรแกรมย่อย
DISPLAY_NINE
นำค่า 0X39 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
NINE
โปรแกรมย่อย
DISPLAY_TEN
นำค่า 0X41 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
TEN
โปรแกรมย่อย
DISPLAY_ELEVEN
นำค่า 0X42 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
ELEVEN
โปรแกรมย่อย
DISPLAY_TWELVE
นำค่า 0X43 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
TWELVE
โปรแกรมย่อย
DISPLAY_THIRTEEN
นำค่า 0X44 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
THIRTEEN
โปรแกรมย่อย
DISPLAY_FOURTEEN
นำค่า 0X45 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย
FOURTEEN

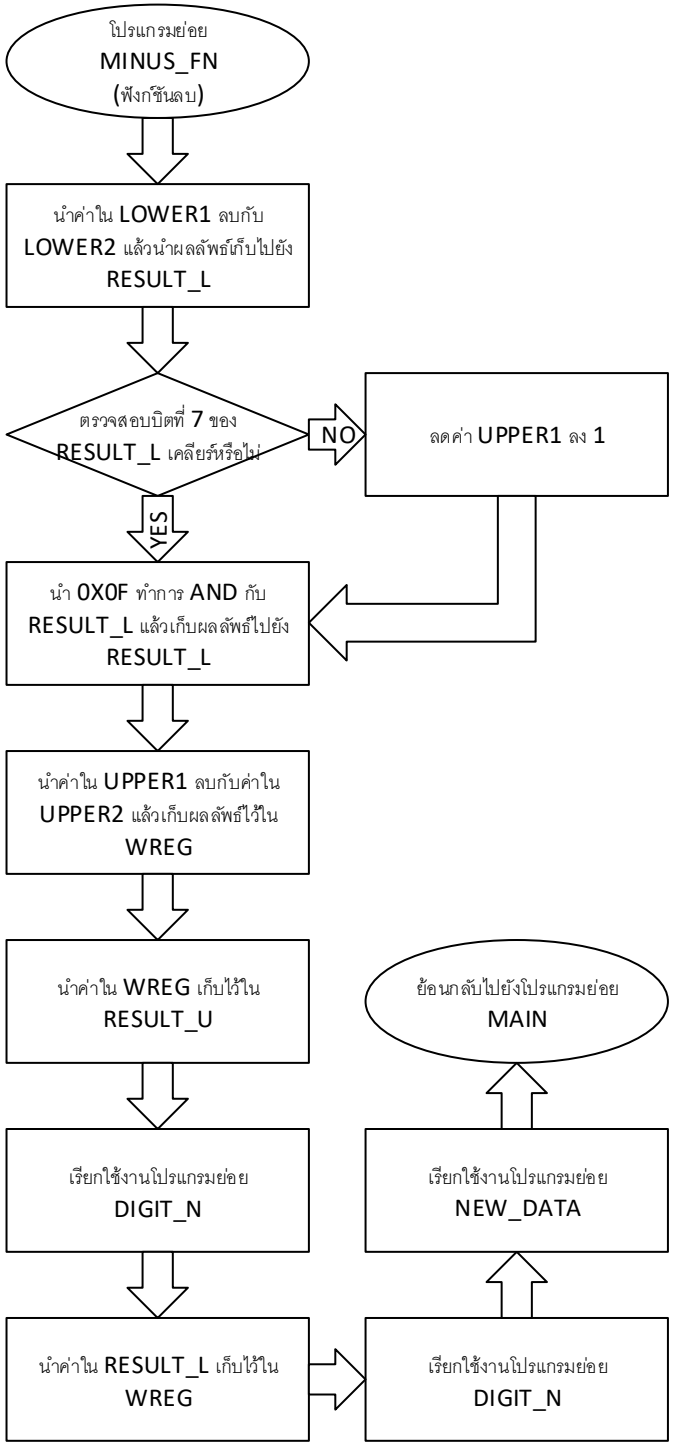

โปรแกรมย่อย FIFTEEN
นำค่า 0x0F ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย STORE_DATA
ทำงานโปรแกรมย่อย DISPLAY_FIFTEEN (แสดง F)
โปรแกรมย่อย ON
นำค่า 0x01 ไปเก็บไว้ใน WREG
เรียกใช้โปรแกรมย่อย DISPLAY
เรียกใช้โปรแกรมย่อย CLEAR_VARIABLE
ย้อนกลับสู่โปรแกรมย่อย MAIN
โปรแกรมย่อย PLUS
นำค่า 0x10 ไปเก็บไว้ใน SIGN
นำค่า 0X2B ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT (แสดง +)
ย้อนกลับสู่โปรแกรมย่อย MAIN
โปรแกรมย่อย MINUS
นำค่า 0x11 ไปเก็บไว้ใน SIGN
นำค่า 0X2D ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT (แสดง -)
ย้อนกลับสู่โปรแกรมย่อย MAIN
โปรแกรมย่อย MUL
นำค่า 0X12 ไปเก็บไว้ใน SIGN
นำค่า 0X78 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT (แสดง x)
ย้อนกลับสู่โปรแกรมย่อย MAIN
โปรแกรมย่อย LSHIFT
นำค่า 0X3C ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT (แสดง >)
เคลียร์บิต CARRY FLAG ในรีจิสเตอร์ STATUS
ทำการชิฟ RESULT_L ไปทางซ้าย แล้วเก็บไว้ในตัวมันเอง
ตอบสอบบิตที่ 4 ของ RESULT_L เคลียร์หรือไม่
NO
ทำการเซ็ตบิต CARRY FLAG ในรีจิสเตอร์ STATUS
YES
ทำการเคลียร์บิตที่ 4 ของ RESULT_L
L1
โปรแกรมย่อย DISPLAY_FIFTEEN
นำค่า 0X46 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
ย้อนกลับไปยังโปรแกรมย่อย FIFTEEN

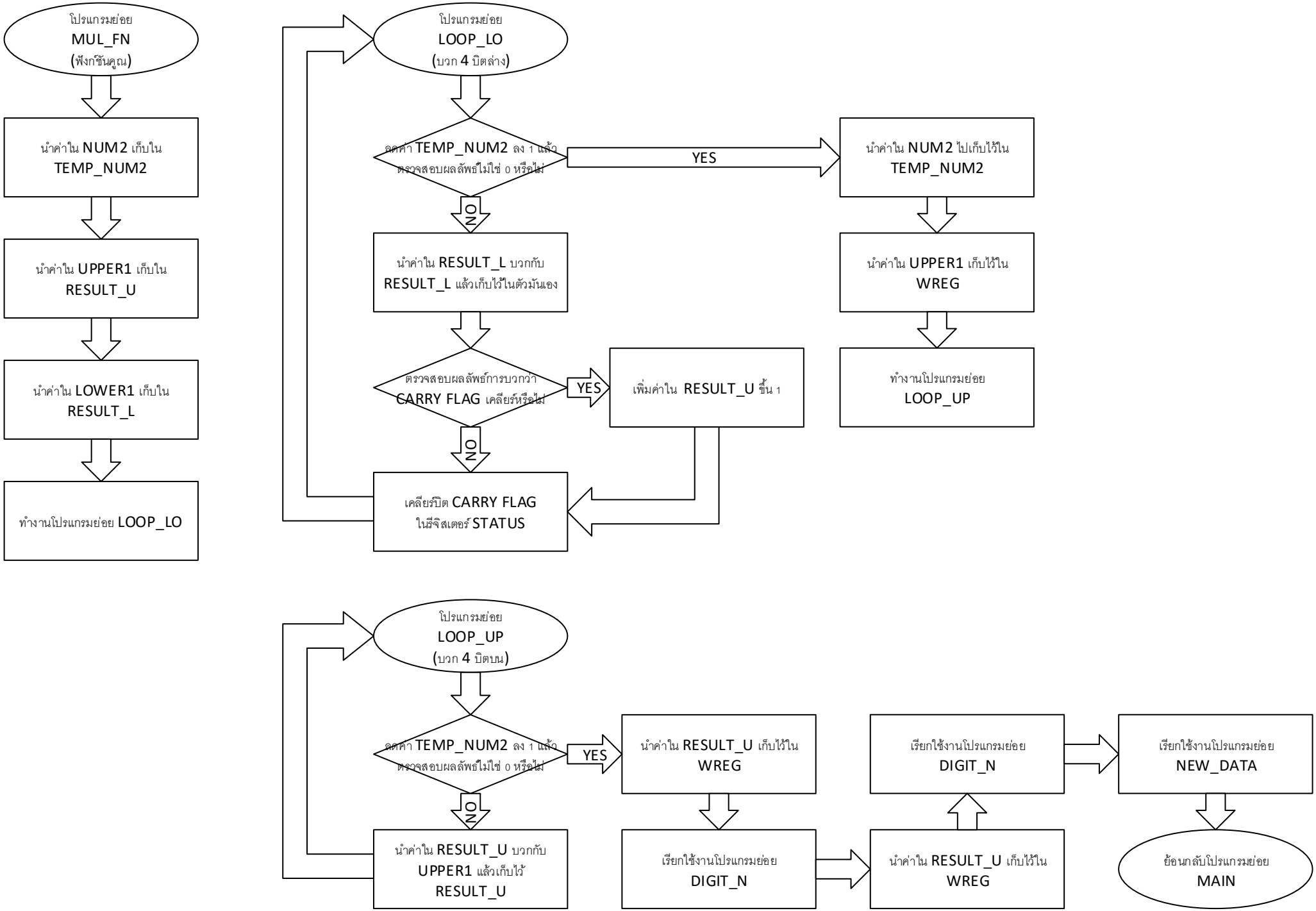

L1
Shift RESULT_U ไปทางซ้าย แล้วเก็บผลลัพธ์ไว้ในตัวมันเอง
เคลียร์บิตที่ 4 ของ RESULT_U
นำค่าใน RESULT_U ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DIGIT_N
นำค่าใน RESULT_L ไปเก็บไว้ใน WREG
เคลียร์บิต CARRY FLAG ในรีจิสเตอร์ STATUS
ย้อนกลับไปยังโปรแกรมย่อย MAIN
โปรแกรมย่อย RSHIFT
นำค่า 0X3E ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT (แสดง >)
เคลียร์บิต CARRY FLAG ในรีจิสเตอร์ STATUS
ทำการชิฟ RESULT_H ไปทางขวา แล้วเก็บไว้ในตัวมันเอง
ตรวจสอบ CARRY FLAG เป็นเคลียร์หรือไม่
NO
เซ็ตบิตที่ 4 ของ RESULT_L
YES
เคลียร์ CARRY FLAG ในรีจิสเตอร์ STATUS
R1
R1
Shift RESULT_L ไปทางซ้าย แล้วเก็บผลลัพธ์ไว้ในตัวมันเอง
เคลียร์บิตที่ 4 ของ RESULT_U
นำค่าใน RESULT_U ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DIGIT_N
นำค่าใน RESULT_L ไปเก็บไว้ใน WREG
เคลียร์บิต CARRY FLAG ในรีจิสเตอร์ STATUS
ย้อนกลับไปยังโปรแกรมย่อย MAIN
โปรแกรมย่อย EQUAL
นำค่า 0X3D ไปเก็บไว้ใน SIGN
เรียกใช้งานโปรแกรมย่อย DISPLAY_DIGIT
กระโดดไปทำงานที่โปรแกรมย่อย CONDITION

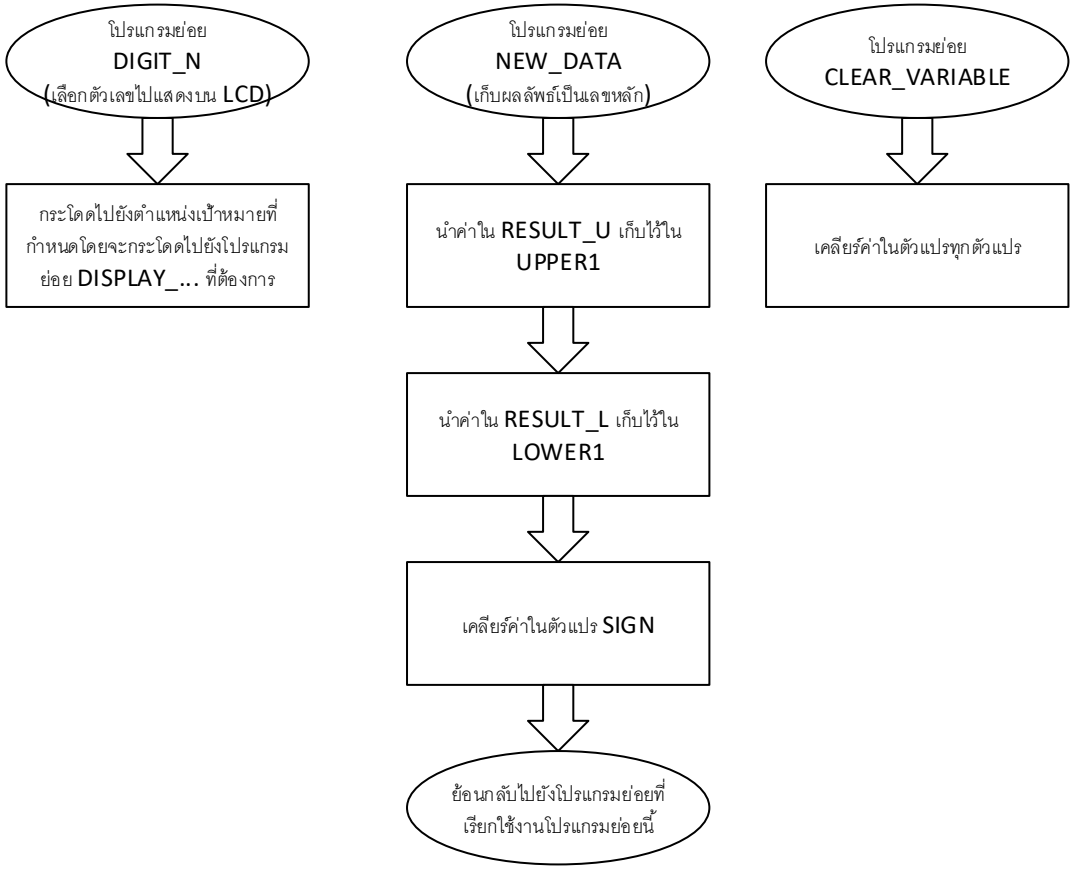

โปรแกรมย่อย
DISPLAY_DIGIT
(แสดงข้อมูลทาง LCD)
เซ็ตขา RC5
เคลียร์ขา RC6
เซ็ตขา RC7
เรียกใช้โปรแกรมย่อย
DISPLAY
นำค่า 0X38 ไปเก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย
DISPLAY
เซ็ตขา RC5
ย้อนกลับไปยังโปรแกรม
ที่เรียกใช้งาน
โปรแกรมย่อย
DISPLAY
ส่งค่าใน WREG ไปยัง PORTB
เคลียร์ขา RC7
[แสดงผลข้อมูลออกทาง LCD]
เรียกใช้งานโปรแกรมย่อย
DELAY
เซ็ตขา RC7
เรียกใช้งานโปรแกรมย่อย
DELAY
เคลียร์ขา RC5
ย้อนกลับไปยังโปรแกรมย่อย
DISPLAY_DIGIT
โปรแกรมย่อย
STORE_DATA
(เก็บข้อมูลเข้าตัวแปร)
ตรวจสอบบิตที่ 4 ของ
SIGN เป็นเซ็ตหรือไม่
YES
NO
กระโดดไปยังโปรแกรมย่อย
NUM1_FN
กระโดดไปยังโปรแกรมย่อย
NUM2_FN
โปรแกรมย่อย
NUM1_FN
(เก็บตัวเลขหลัก)
ตรวจสอบบิตที่ 7 ของ
SIGN เป็นเซ็ตหรือไม่
YES
NO
กระโดดไปยังโปรแกรมย่อย
LOWER1_FN
กระโดดไปยังโปรแกรมย่อย
UPPER1_FN
โปรแกรมย่อย
NUM2_FN
(เก็บตัวเลขรอง)
ตรวจสอบบิตที่ 6 ของ
SIGN เป็นเซ็ตหรือไม่
YES
NO
กระโดดไปยังโปรแกรมย่อย
LOWER2_FN
กระโดดไปยังโปรแกรมย่อย
UPPER2_FN


โปรแกรมย่อย
LOWER1_FN
(เก็บ 4 บิตล่างของหลัก)
นำค่าใน WREG
ไปเก็บไว้ใน LOWER1, NUM1
และ RESULT_L
เซ็ตบิตที่ 7 ของ SIGN
ย้อนกลับไปยังโปรแกรมย่อย
MAIN
โปรแกรมย่อย
UPPER1_FN
(เก็บ 4 บิตบนของหลัก)
นำค่าใน WREG ไปเก็บไว้ใน
RESULT_L
นำค่าใน LOWER1 ไปเก็บไว้ใน
UPPER1 และ RESULT_U
นำค่าใน RESULT_L ไปเก็บไว้ใน
LOWER1
ทำการสลับบิตบนและบิตล่างของตัว
แปร UPPER1 แล้วเก็บไว้ใน
WREG
ทำการนำค่า UPPER1 และ
LOWER1 มารวมกันแล้วเก็บไว้ใน
NUM1 โดยใช้ IORWF
เคลียร์บิตที่ 7 ของ SIGN
ย้อนกลับไปยังโปรแกรมย่อย
MAIN
โปรแกรมย่อย
LOWER2_FN
(เก็บ 4 บิตล่างของรอง)
นำค่าใน WREG
ไปเก็บไว้ใน LOWER2, NUM2
และ RESULT_L
เซ็ตบิตที่ 6 ของ SIGN
ย้อนกลับไปยังโปรแกรมย่อย
MAIN
โปรแกรมย่อย
UPPER2_FN
(เก็บ 4 บิตบนของรอง)
นำค่าใน WREG ไปเก็บไว้ใน
RESULT_L
นำค่าใน LOWER2 ไปเก็บไว้ใน
UPPER2 และ RESULT_U
นำค่าใน RESULT_L ไปเก็บไว้ใน
LOWER2
ทำการสลับบิตบนและบิตล่างของตัว
แปร UPPER2 แล้วเก็บไว้ใน
WREG
ทำการนำค่า UPPER2 และ
LOWER2 มารวมกันแล้วเก็บไว้ใน
NUM2 โดยใช้ IORWF
เคลียร์บิตที่ 6 ของ SIGN
ย้อนกลับไปยังโปรแกรมย่อย
MAIN
โปรแกรมย่อย
CONDITION
(เลือกว่าจะทำบวกลบหรือคูณ)
ตรวจสอบบิตที่ 1 ของ
SIGN เป็นเซ็ตหรือไม่
YES
NO
กระโดดไปยังโปรแกรมย่อย
PLUS_MINUS_FN
กระโดดไปยังโปรแกรมย่อย
MUL_FN
โปรแกรมย่อย
PLUS_MINUS_FN
ตรวจสอบบิตที่ 0 ของ
SIGN เป็นเซ็ตหรือไม่
YES
NO
กระโดดไปยังโปรแกรมย่อย
PLUS_FN
กระโดดไปยังโปรแกรมย่อย
MINUS_FN


โปรแกรมย่อย
PLUS_FN
(ฟังก์ชันบวก)
นำค่าใน LOWER2 บวกกับ LOWER1 แล้วนำผลลัพธ์เก็บไปยัง RESULT_L
ตรวจสอบบิตที่ 4 ของ RESULT_L เคลียร์หรือไม่
NO
เพิ่มค่า UPPER2 ขึ้น 1
YES
เคลียร์บิตที่ 4 ของ RESULT_L
นำค่าใน UPPER2 บวกกับค่าใน UPPER1 แล้วเก็บผลลัพธ์ไว้ใน WREG
นำค่าใน WREG เก็บไว้ใน RESULT_U
เรียกใช้งานโปรแกรมย่อย DIGIT_N
นำค่าใน RESULT_L เก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย DIGIT_N
เรียกใช้งานโปรแกรมย่อย NEW_DATA
ย้อนกลับไปยังโปรแกรมย่อย MAIN


โปรแกรมย่อย
MINUS_FN
(ฟังก์ชันลบ)
นำค่าใน LOWER1 ลบกับ LOWER2 แล้วนำผลลัพธ์เก็บไปยัง RESULT_L
ตรวจสอบบิตที่ 7 ของ RESULT_L เคลียร์หรือไม่
NO
ลดค่า UPPER1 ลง 1
YES
นำ 0X0F ทำการ AND กับ RESULT_L แล้วเก็บผลลัพธ์ไปยัง RESULT_L
นำค่าใน UPPER1 ลบกับค่าใน UPPER2 แล้วเก็บผลลัพธ์ไว้ใน WREG
นำค่าใน WREG เก็บไว้ใน RESULT_U
เรียกใช้งานโปรแกรมย่อย DIGIT_N
นำค่าใน RESULT_L เก็บไว้ใน WREG
เรียกใช้งานโปรแกรมย่อย NEW_DATA
ย้อนกลับไปยังโปรแกรมย่อย MAIN


โปรแกรมย่อย
MUL_FN
(ฟังก์ชันคูณ)
นำค่าใน NUM2 เก็บใน
TEMP_NUM2
นำค่าใน UPPER1 เก็บใน
RESULT_U
นำค่าใน LOWER1 เก็บใน
RESULT_L
ทำงานโปรแกรมย่อย LOOP_LO
โปรแกรมย่อย
LOOP_LO
(บวก 4 บิตล่าง)
ลดค่า TEMP_NUM2 ลง 1 แล้ว
ตรวจสอบผลลัพธ์ไม่ใช่ 0 หรือไม่
YES
NO
นำค่าใน NUM2 ไปเก็บไว้ใน
TEMP_NUM2
นำค่าใน RESULT_L บวกกับ
RESULT_L แล้วเก็บไว้ในตัวมันเอง
นำค่าใน UPPER1 เก็บไว้ใน
WREG
ตรวจสอบผลลัพธ์การบวกว่า
CARRY FLAG เคลียร์หรือไม่
YES
NO
เพิ่มค่าใน RESULT_U ขึ้น 1
ทำงานโปรแกรมย่อย
LOOP_UP
เคลียร์บิต CARRY FLAG
ในรีจิสเตอร์ STATUS
โปรแกรมย่อย
LOOP_UP
(บวก 4 บิตบน)
ลดค่า TEMP_NUM2 ลง 1 แล้ว
ตรวจสอบผลลัพธ์ไม่ใช่ 0 หรือไม่
YES
NO
นำค่าใน RESULT_U เก็บไว้ใน
WREG
เรียกใช้งานโปรแกรมย่อย
DIGIT_N
เรียกใช้งานโปรแกรมย่อย
NEW_DATA
นำค่าใน RESULT_U บวกกับ
UPPER1 แล้วเก็บไว้
RESULT_U
เรียกใช้งานโปรแกรมย่อย
DIGIT_N
นำค่าใน RESULT_U เก็บไว้ใน
WREG
ย้อนกลับโปรแกรมย่อย
MAIN


โปรแกรมย่อย
DIGIT_N
(เลือกตัวเลขไปแสดงบน LCD)
กระโดดไปยังตำแหน่งเป้าหมายที่กำหนดโดยจะกระโดดไปยังโปรแกรมย่อย DISPLAY_... ที่ต้องการ
โปรแกรมย่อย
NEW_DATA
(เก็บผลลัพธ์เป็นเลขหลัก)
นำค่าใน RESULT_U เก็บไว้ใน UPPER1
นำค่าใน RESULT_L เก็บไว้ใน LOWER1
เคลียร์ค่าในตัวแปร SIGN
ย้อนกลับไปยังโปรแกรมย่อยที่เรียกใช้งานโปรแกรมย่อยนี้
โปรแกรมย่อย
CLEAR_VARIABLE
เคลียร์ค่าในตัวแปรทุกตัวแปร

สรุปผลการทำงาน

จากการทำงานของโปรแกรม Easily Hex Calculator จะเริ่มจากการตรวจสอบการกดปุ่ม หากมีการกดปุ่ม ทางโปรแกรมจะทำการนำตัวเลขที่ได้ไปเก็บไว้เป็นตัวเลข 4 บิตล่างหรือ LOWER1 และเก็บไว้ใน RESULT_L เพื่อรองรับการกระทำฟังก์ชันประเภทชิฟ หลังจากนั้นโปรแกรมจะทำการรอการกดปุ่มอีกครั้ง หากมีการกดปุ่มครั้งที่สองสามารถกดเป็นตัวเลขหรือฟังก์ชันใด ๆ ก็ได้ หากเป็นการกดตัวเลขทางโปรแกรมจะนำตัวเลข 4 บิตล่างขึ้นไปเป็น 4 บิตบนหรือ UPPER1 และเก็บไว้ใน RESULT_H แล้วนำตัวเลขที่รับมาใหม่เก็บเป็น 4 บิตล่างหรือ LOWER1 และเก็บไว้ใน RESULT_L แทน หลังจากนั้นรวมตัวเลขระหว่าง 4 บิตล่างและ 4 บิตเก็บไว้ในตัวแปร NUM1 เมื่อกระทำการเสร็จสิ้น ทางโปรแกรมจะกลับมารอการกดปุ่มอีกครั้ง โดยครั้งนี้จะต้องเป็นการกดเฉพาะปุ่มฟังก์ชันเท่านั้น เพื่อให้โปรแกรมเก็บตัวเลขชุดต่อไปไว้ในตัวแปร LOWER2, UPPER2 หรือ NUM2 หากทำการกดตัวเลขโดยไม่ทำการกดปุ่มฟังก์ชันก่อน โปรแกรมจะไม่สามารถใช้งานได้ หรือ ERROR ทันที โดยข้อเสียของโปรแกรมนี้ ถ้าหากมีการบวกกันแล้วมีผลลัพธ์เกินจำนวน 8 บิตจะทำให้โปรแกรมแสดงผลลัพธ์ไม่ถูกต้อง รวมถึงการลบกันแล้วเกิด overflow แบบกลับหลัง

```
; HEX_CALCULATOR
      LIST        P=16F877A
      #INCLUDE    "P16F877A.INC"

; CREATE VARIABLE

UPPER1          EQU         0X50
LOWER1          EQU         0X51
UPPER2          EQU         0X52
LOWER2          EQU         0X53
NUM1            EQU         0X54
NUM2            EQU         0X55
TEMP_NUM2       EQU         0X56
RESULT_U        EQU         0X57
RESULT_L        EQU         0X58
SIGN            EQU         0X59
DL_M            EQU         0X60
DL_N            EQU         0X61

; SET VARIABLE

                CALL        CLEAR_VARIABLE

;=============================================

; CONFIGURING AS INPUT AND OUTPUT PORTS

      ; SELECT BANK1
                BSF         STATUS, RP0

          ; FOR KEYPAD CONFIG
                BSF         TRISC, 0
                BSF         TRISC, 1
                BSF         TRISC, 2
                BSF         TRISC, 3

                MOVLW       0XC0
                MOVWF       TRISB

          ; FOR LCD CONFIG
```

```
                BCF     TRISC, 5
                BCF     TRISC, 6
                BCF     TRISC, 7

        ; FOR LCD OUTPUT
                MOVLW   0X00
                MOVWF   TRISD

    ; SELECT BANK0
                BCF     STATUS, RP0

;=============================================

; MAIN FUNCTION
MAIN            CALL    CHECK_KEYPAD
                GOTO    MAIN

;=============================================

CHECK_KEYPAD

    ; SCAN THE 1ST COLUMN OF KEYS
                BSF     PORTB, 0

        ; HAS THE 0 KEY BEEN PRSSED ?
                BTFSC   PORTC, 0
                CALL    ZERO

        ; HAS THE 4 KEY BEEN PRESSED ?
                BTFSC   PORTC, 1
                CALL    FOUR

        ; HAS THE 8 KEY BEEN PRESSED ?
                BTFSC   PORTC, 2
                CALL    EIGHT

        ; HAS THE 12 KEY BEEN PRESSED ?
                BTFSC   PORTC, 3
                CALL    TWELVE
```

```
                BCF         PORTB, 0
;-----------------------------------------------------------------

; SCAN THE 2ND COLUMN OF KEYS
                BSF         PORTB, 1

    ; HAS THE 1 KEY BEEN PRSSED ?
                BTFSC       PORTC, 0
                CALL        ONE

    ; HAS THE 5 KEY BEEN PRESSED ?
                BTFSC       PORTC, 1
                CALL        FIVE

    ; HAS THE 9 KEY BEEN PRESSED ?
                BTFSC       PORTC, 2
                CALL        NINE

    ; HAS THE 13 KEY BEEN PRESSED ?
                BTFSC       PORTC, 3
                CALL        THIRTEEN

                BCF         PORTB, 1
;-----------------------------------------------------------------

; SCAN THE 3RD COLUMN OF KEYS
                BSF         PORTB, 2

    ; HAS THE 2 KEY BEEN PRSSED ?
                BTFSC       PORTC, 0
                CALL        TWO

    ; HAS THE 6 KEY BEEN PRESSED ?
                BTFSC       PORTC, 1
                CALL        SIX

    ; HAS THE 10 KEY BEEN PRESSED ?
                BTFSC       PORTC, 2
                CALL        TEN
```

```
        ; HAS THE 14 KEY BEEN PRESSED ?
                BTFSC   PORTC, 3
                CALL    FOURTEEN

                BCF     PORTB, 2
;-----------------------------------------------------------------

; SCAN THE 4TH COLUMN OF KEYS
                BSF     PORTB, 3

        ; HAS THE 3 KEY BEEN PRSSED ?
                BTFSC   PORTC, 0
                CALL    THREE

        ; HAS THE 7 KEY BEEN PRESSED ?
                BTFSC   PORTC, 1
                CALL    SEVEN

        ; HAS THE 11 KEY BEEN PRESSED ?
                BTFSC   PORTC, 2
                CALL    ELEVEN

        ; HAS THE 15 KEY BEEN PRESSED ?
                BTFSC   PORTC, 3
                CALL    FIFTEEN

                BCF     PORTB, 3
;-----------------------------------------------------------------

; SCAN THE 5TH COLUMN OF KEYS
                BSF     PORTB, 4

        ; HAS THE * KEY BEEN PRSSED ?
                BTFSC   PORTC, 0
                CALL    MUL

        ; HAS THE LEFT-SHIFT KEY BEEN PRESSED ?
                BTFSC   PORTC, 1
                CALL    LSHIFT
```

```
            ; HAS THE PLUS KEY BEEN PRESSED ?
                    BTFSC       PORTC, 2
                    CALL        PLUS

            ; HAS THE PLUS KEY BEEN PRESSED ?
                    BTFSC       PORTC, 3
                    CALL        PLUS

                    BCF         PORTB, 4
        ;-----------------------------------------------------------------

        ; SCAN THE 6TH COLUMN OF KEYS
                    BSF         PORTB, 5

            ; HAS THE ON/OFF KEY BEEN PRSSED ?
                    BTFSC       PORTC, 0
                    CALL        ON

            ; HAS THE RIGHT-SHIFT KEY BEEN PRESSED ?
                    BTFSC       PORTC, 1
                    CALL        RSHIFT

            ; HAS THE MINUS KEY BEEN PRESSED ?
                    BTFSC       PORTC, 2
                    CALL        MINUS

            ; HAS THE EQUAL KEY BEEN PRESSED ?
                    BTFSC       PORTC, 3
                    CALL        EQUAL

                    BCF         PORTB, 5
        ;-----------------------------------------------------------------
                    RETURN  ; RETURN TO MAIN FUNCTION

    ;=============================================

    ; KEY FUNCTION

    ZERO            MOVLW       0X00
                    CALL        STORE_DATA
```

```
DISPLAY_ZERO
                MOVLW   0X30  ; ASCII '0'
                CALL    DISPLAY_DIGIT
                RETURN

ONE             MOVLW   0X01
                CALL    STORE_DATA
DISPLAY_ONE
                MOVLW   0X31  ; ASCII '1'
                CALL    DISPLAY_DIGIT
                RETURN

TWO             MOVLW   0X02
                CALL    STORE_DATA
DISPLAY_TWO
                MOVLW   0X32  ; ASCII '2'
                CALL    DISPLAY_DIGIT
                RETURN

THREE           MOVLW   0X03
                CALL    STORE_DATA
DISPLAY_THREE
                MOVLW   0X33  ; ASCII '3'
                CALL    DISPLAY_DIGIT
                RETURN

FOUR            MOVLW   0X04
                CALL    STORE_DATA
DISPLAY_FOUR
                MOVLW   0X34  ; ASCII '4'
                CALL    DISPLAY_DIGIT
                RETURN

FIVE            MOVLW   0X05
                CALL    STORE_DATA
DISPLAY_FIVE
                MOVLW   0X35  ; ASCII '5'
                CALL    DISPLAY_DIGIT
                RETURN
```

```
SIX             MOVLW   0X06
                CALL    STORE_DATA
DISPLAY_SIX
                MOVLW   0X36  ; ASCII '6'
                CALL    DISPLAY_DIGIT
                RETURN

SEVEN           MOVLW   0X07
                CALL    STORE_DATA
DISPLAY_SEVEN
                MOVLW   0X37  ; ASCII '7'
                CALL    DISPLAY_DIGIT
                RETURN

EIGHT           MOVLW   0X08
                CALL    STORE_DATA
DISPLAY_EIGHT
                MOVLW   0X38  ; ASCII '8'
                CALL    DISPLAY_DIGIT
                RETURN

NINE            MOVLW   0X09
                CALL    STORE_DATA
DISPLAY_NINE
                MOVLW   0X39  ; ASCII '9'
                CALL    DISPLAY_DIGIT
                RETURN

TEN             MOVLW   0X0A
                CALL    STORE_DATA
DISPLAY_TEN
                MOVLW   0X41  ; ASCII 'A'
                CALL    DISPLAY_DIGIT
                RETURN

ELEVEN          MOVLW   0X0B
                CALL    STORE_DATA
DISPLAY_ELEVEN
                MOVLW   0X42  ; ASCII 'B'
                CALL    DISPLAY_DIGIT
```

```
                RETURN

TWELVE          MOVLW   0X0C
                CALL    STORE_DATA
DISPLAY_TWELVE
                MOVLW   0X43  ; ASCII 'C'
                CALL    DISPLAY_DIGIT
                RETURN

THIRTEEN        MOVLW   0X0D
                CALL    STORE_DATA
DISPLAY_THIRTEEN
                MOVLW   0X44  ; ASCII 'D'
                CALL    DISPLAY_DIGIT
                RETURN

FOURTEEN        MOVLW   0X0E
                CALL    STORE_DATA
DISPLAY_FOURTEEN
                MOVLW   0X45  ; ASCII 'E'
                CALL    DISPLAY_DIGIT
                RETURN

FIFTEEN         MOVLW   0X0F
                CALL    STORE_DATA
DISPLAY_FIFTEEN
                MOVLW   0X46  ; ASCII 'F'
                CALL    DISPLAY_DIGIT
                RETURN

;-----------------------------------------------------------------

; FUNCTION
PLUS            MOVLW   0X10  ; 'xxx1 xx00'
                MOVWF   SIGN
                MOVLW   0X2B  ; ASCII '+'
                CALL    DISPLAY_DIGIT
                RETURN

MINUS           MOVLW   0X11  ; 'xxx1 xx01'
```

```
                MOVWF   SIGN
                MOVLW   0X2D  ; ASCII '-'
                CALL    DISPLAY_DIGIT
                RETURN

MUL             MOVLW   0X12  ;  'xxx1 xx10'
                MOVWF   SIGN
                MOVLW   0X78  ; ASCII 'x'
                CALL    DISPLAY_DIGIT
                RETURN

LSHIFT          MOVLW   0X3C
                CALL    DISPLAY_DIGIT

                BCF     STATUS, C
                RLF     RESULT_L, F
                BTFSC   RESULT_L, 4
                BSF     STATUS, C
                BCF     RESULT_L, 4
                RLF     RESULT_U, F
                BCF     RESULT_U, 4

                MOVF    RESULT_U, W
                CALL    DIGIT_N
                MOVF    RESULT_L, W
                CALL    DIGIT_N
                BCF     STATUS, C
                RETURN

RSHIFT          MOVLW   0X3E
                CALL    DISPLAY_DIGIT

                BCF     STATUS, C
                RRF     RESULT_U, F
                BTFSC   STATUS, C
                BSF     RESULT_L, 4
                BCF     STATUS, C
                RRF     RESULT_L, F
                BCF     STATUS, C
```

```
                MOVF    RESULT_U, W
                CALL    DIGIT_N
                MOVF    RESULT_L, W
                CALL    DIGIT_N
                BCF     STATUS, C
                RETURN

EQUAL           MOVLW   0X3D
                CALL    DISPLAY_DIGIT
                GOTO    CONDITION
                RETURN

ON              MOVLW   0X01
                CALL    DISPLAY
                CALL    CLEAR_VARIABLE
                RETURN

;=========================================

CLEAR_VARIABLE
                CLRF    UPPER1
                CLRF    LOWER1
                CLRF    UPPER2
                CLRF    LOWER2
                CLRF    NUM1
                CLRF    NUM2
                CLRF    TEMP_NUM2
                CLRF    RESULT_U
                CLRF    RESULT_L
                CLRF    SIGN
                CLRF    DL_M
                CLRF    DL_N
                RETURN

;=========================================

DIGIT_N         ADDWF   PCL, F
                GOTO    DISPLAY_ZERO
                GOTO    DISPLAY_ONE
                GOTO    DISPLAY_TWO
```

```
                GOTO    DISPLAY_THREE
                GOTO    DISPLAY_FOUR
                GOTO    DISPLAY_FIVE
                GOTO    DISPLAY_SIX
                GOTO    DISPLAY_SEVEN
                GOTO    DISPLAY_EIGHT
                GOTO    DISPLAY_NINE
                GOTO    DISPLAY_TEN
                GOTO    DISPLAY_ELEVEN
                GOTO    DISPLAY_TWELVE
                GOTO    DISPLAY_THIRTEEN
                GOTO    DISPLAY_FOURTEEN
                GOTO    DISPLAY_FIFTEEN

;=============================================

CONDITION BTFSS     SIGN, 1
                GOTO    PLUS_MINUS_FN
                GOTO    MUL_FN

PLUS_MINUS_FN
                BTFSS   SIGN, 0
                GOTO    PLUS_FN
                GOTO    MINUS_FN

PLUS_FN         MOVF    LOWER2, W
                ADDWF   LOWER1, W
                MOVWF   RESULT_L
                BTFSC   RESULT_L, 4
                INCF    UPPER2, F
                BCF     RESULT_L, 4
                MOVF    UPPER2, W
                ADDWF   UPPER1, W

                MOVWF   RESULT_U
                CALL    DIGIT_N
                MOVF    RESULT_L, W
                CALL    DIGIT_N

                CALL    NEW_DATA
```

```
                RETURN

MINUS_FN        MOVF    LOWER2, W
                SUBWF   LOWER1, W
                MOVWF   RESULT_L
                BTFSC   RESULT_L, 7
                DECF    UPPER1, F
                MOVLW   0X0F
                ANDWF   RESULT_L, F  ; CLEAR BIT7 - BIT4
                MOVF    UPPER2, W
                SUBWF   UPPER1, W

                MOVWF   RESULT_U
                CALL    DIGIT_N
                MOVF    RESULT_L, W
                CALL    DIGIT_N

                CALL    NEW_DATA
                RETURN

MUL_FN          MOVF    NUM2, W
                MOVWF   TEMP_NUM2
                MOVF    UPPER1, W
                MOVWF   RESULT_U
                MOVF    LOWER1, W
                MOVWF   RESULT_L

LOOP_LO         DECF    TEMP_NUM2, F
                BTFSC   STATUS, Z  ; 0 SKIPED
                GOTO    $+6
                ADDWF   RESULT_L, F
                BTFSC   STATUS, C
                INCF    RESULT_U, F
                BCF     STATUS, C
                GOTO    LOOP_LO

                MOVF    NUM2, W
                MOVWF   TEMP_NUM2
                MOVF    UPPER1, W
                ; W = UPPER1
```

```
LOOP_UP     DECF    TEMP_NUM2, F
            BTFSC   STATUS, Z
            GOTO    $+3
            ADDWF   RESULT_U, F
            GOTO    LOOP_UP

            MOVF    RESULT_U, W
            CALL    DIGIT_N
            MOVF    RESULT_L, W
            CALL    DIGIT_N

            CALL    NEW_DATA
            RETURN

;===============================================

STORE_DATA  BTFSS   SIGN, 4
            GOTO    NUM1_FN
            GOTO    NUM2_FN

NUM1_FN     BTFSS   SIGN, 7
            GOTO    LOWER1_FN
            GOTO    UPPER1_FN

NUM2_FN     BTFSS   SIGN, 6
            GOTO    LOWER2_FN
            GOTO    UPPER2_FN

LOWER1_FN   MOVWF   LOWER1
            MOVWF   NUM1
            MOVWF   RESULT_L
            BSF     SIGN, 7
            RETURN

UPPER1_FN   MOVWF   RESULT_L
            MOVF    LOWER1, W
            MOVWF   UPPER1
            MOVWF   RESULT_U
            MOVF    RESULT_L, W
            MOVWF   LOWER1
```

```
                SWAPF     UPPER1, W
                IORWF     LOWER1, W
                MOVWF     NUM1
                BCF       SIGN, 7
                RETURN

LOWER2_FN       MOVWF     LOWER2
                MOVWF     NUM2
                MOVWF     RESULT_L
                BSF       SIGN, 6
                RETURN

UPPER2_FN       MOVWF     RESULT_L
                MOVF      LOWER2, W
                MOVWF     UPPER2
                MOVWF     RESULT_U
                MOVF      RESULT_L, W
                MOVWF     LOWER2

                SWAPF     UPPER2, W
                IORWF     LOWER2, W
                MOVWF     NUM2
                BCF       SIGN, 6
                RETURN

NEW_DATA MOVF       RESULT_U, W
                MOVWF     UPPER1
                MOVF      RESULT_L, W
                MOVWF     LOWER1
                CLRF      SIGN
                RETURN
;===============================================

; DISPLAY THE DIGIT IN THE 7SEGMENT
      ; IF
            ; RS=0  INSTRUCTION COMMAND CODE REGISTER IS
SELECTED, ALLOWING USER TO SEND COMMAND
            ; RS=1  DATA REGISTER IS SELECTED ALLOWING TO SEND
DATA THAT HAS TO BE DISPLAYED.
```

```
        ; R\W=0  READING
        ; R\W=1  WRITING
        ; E- ENABLE

        ; THE ENABLE PIN IS USED BY THE LCD TO LATCH
INFORMATION AT ITS DATA PINS. WHEN DATA IS SUPPLIED TO DATA
PINS,
        ; A HIGH TO LOW PULSE MUST BE APPLIED TO THIS PIN IN
ORDER FOR THE LCD TO LATCH THE DATA PRESENT IN THE DATA
PINS.
        ; E SHOULD  TOGGLE

        ; DATA MODE:  RS=1, R\W=0, E=1\0

DISPLAY_DIGIT
                BSF     PORTC, 5  ; CONTROL SIGNAL TO RS
                BCF     PORTC, 6  ; CONTROL SIGNAL TO R/W
                BSF     PORTC, 7  ; CONTROL SIGNAL TO ENABLE

                CALL    DISPLAY

                MOVLW   0X38  ; INITIALISES THE DISPLAY
                CALL    DISPLAY

                MOVLW   0X0E  ; DON'T BLINK THE CURSOR
                CALL    DISPLAY

                BSF     PORTC, 5
                RETURN

DISPLAY         MOVWF   PORTD
            ; SEND DATA TO DATA PINS
                BCF     PORTC, 7
                CALL    DELAY

                BSF     PORTC, 7
                CALL    DELAY
            ; SELECT COMMAND REGISTER
                BCF     PORTC, 5
```

```
                RETURN

DELAY           MOVLW   0X0D  ; VERY SMALL DELAY
                MOVWF   DL_N
                MOVLW   0XFB
                MOVWF   DL_M
LOOP            DECFSZ  DL_M
                GOTO    LOOP
                DECFSZ  DL_N
                GOTO    LOOP
                RETURN
                END
```