# รายการอ้างอิง


1. Clarke, D.W. PID algorithms and their computer implementation. Trans Inst MC. Oct-Dec, 1984. Vol 6, No 6, pp. 305-316.
2. สมบูรณ์ จงชัยกิจ. เอกสารประกอบการบรรยายเรื่องการควบคุมแบบอัตโนมัติและการควบคุมแบบ PID. สมาคมเทคโนโลยี (ไทย-ญี่ปุ่น).
3. Model SLPC, SLMC & SPLR Programmable Instrument Function & Application. Technical Information. YEW, 1985
4. Telerm Compact Controller E. Technical Manual Fuji Electric Co., Ltd.
5. Shimaden, Temperature and Humidity Control Specialists Model SR24. Tokyo : SHIMADEN CO.,LTD.
6. Bennett, Stuart. Real-Time Computer Control: An introduction. London : Prentice Hall International (UK) Ltd., 1989.
7. 8-Bit Embedded Controller Handbook. Santa Clara, USA : Intel Corporation, 1989.
8. Embedded Controll Applications Handbook. Santa Clara, USA : Intel Corporation, 1989.
9. Petersen, Christopher M. RAM test program prevents crashes. EDN magazine. October 25,1990. pp. 204-205.
10. Yammiyavar, Pradeep. Control Panel Design. Bangalore, India : CEDT Publication, 1988.
11. Yammiyavar, Pradeep. Ergonomics for Electronics equipment design. Bangalore, India : CEDT Publication, 1987.
12. อำนวย แสงวิโรจนพัฒน์. ตัวควบคุมเชิงเลขชนิดโปรแกรมได้สำหรับกระบวนการทางอุตสาหกรรมแบบต่อเนื่อง. กรุงเทพมหานคร : วิทยานิพนธ์ จุฬาลงกรณ์วิทยาลัย, 2532.

13. สุริยงค์ เลิศกุลวาณิชย์, ดร.สมบูรณ์ จงชัยกิจ. การพัฒนาเครื่องควบคุม PID เชิงเลขขนาดกะทัดรัด. กรุงเทพมหานคร : การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 15 สถาบันเทคโนโลยีพระจอมเกล้าธนบุรี, 2535.

14. Shimaden. SR21 series service manual. Tokyo : SHIMADEN Co., LTd.

15. Ogata, Katsuhiko. Discrete-time control systems. Englewood Cliffs, New Jersey : Prentice-Hall International, Inc., 1987.

16. Lukas, Michael P. "Distributed Control Systems Their Evaluation and Design. New York : Van Nostrand Reinhold Company, 1986.

17. Dot Matrix LCD module user manual. Bangkok : Sila Research Co., LTD., 1993.

18. Linear Data book. Santa Clara, California, USA : National Semiconductor Corporation.

19. Linear Applications Handbook. Santa Clara, California, USA : National Semiconductor Corporation.

20. Sheingold, Daniel H. Transducer interfacing handbook A guide to Analog Signal Conditioning. Massachusetts 02062, USA : Analog Devices Inc., 1981.

21. Weber, Samuel. Circuits for eletronics engineers. New York : McGraw-Hill,Inc., 1977.

22. Kenneth L.Short. Microprocessors and Programmed Logic. India : Prentic-Hall of India Private Limited, 1988.

23. PAL Device Data Book. Santra Clara, USA : Advanced Micre Devices, Inc., 1982.

24. REFERENCE MANUAL. California USA : FRANKLIN SOFRWARE, INC., 1987-1990.

25. ICE-51FX/PC. In-Circuit Emulator. USA : INTEL CORPORATION, 1990

# ภาคผนวก

# ภาคผนวก ก

# คู่มือการใช้งานตัวควบคุม PID เชิงเลขขนาดกะทัดรัด

ตัวควบคุม PID เชิงเลข NECTEC-EDL เป็นตัวควบคุมที่ใช้สำหรับโปรเซสที่ต้องการการควบคุมแบบต่อเนื่องภายในโรงงานอุตสาหกรรม โดยลักษณะของการควบคุมใช้ได้ทั้งแบบลูปเดี่ยว (Single Loop) และแบบต่อเรียงกัน (Cascade) สำหรับอินพุตที่ใช้ได้กับ TC (สำหรับโมเดล TC) และแบบกระแสมาตรฐาน (สำหรับโมเดล mA) ส่วนเอาต์พุตเป็นกระแสมาตรฐาน นอกจากนี้แล้วมีหน้าสัมผัสสำหรับการเตือนเมื่อการควบคุมออกนอกช่วงการทำงาน

ในการกำหนดค่าพารามิเตอร์หรือเลือกฟังก์ชันการทำงานของตัวควบคุม สามารถทำได้โดยใช้ปุ่มเพียง 5 ปุ่ม บนแผงหน้าปัด และการกำหนดวิธีการทำงานหรือสถานะการทำงานของเครื่องด้วยการเลือกดิพสวิตช์

## 1. คุณลักษณะของเครื่อง

### 1.1 ลักษณะของแผงหน้าปัด

**ภาคแสดงผล**

ตัวแสดงผล : LCD ขนาด 20 อักขระ 2 แถว ขนาดอักขระ 5x8 จุด

รูปแบบ : 2 รูปแบบ แสดงค่าข้อมูลแท้จริงของตัวแปร และแสดงกราฟแท่งของตัวแปรในโปรเซส

**ภาครับข้อมูล**

ลักษณะ : ปุ่มจำนวน 5 ปุ่ม เป็นปุ่มสวิตช์แบบผิวบาง (Membrane switch)

คุณลักษณะของปุ่ม : บางปุ่มประกอบด้วยฟังก์ชันพื้นฐาน (สีฟ้า) และฟังก์ชันพิเศษ (สีขาว)

ลักษณะการกด : ฟังก์ชันพื้นฐาน กดแล้วปล่อยทันที ส่วนฟังก์ชันพิเศษกดนานกว่า 1 วินาที

**Note** สำหรับปุ่ม Increment และ Decrement มีฟังก์ชันพิเศษ เป็นการทำงานอย่างรวดเร็ว (Fast mode)

1.2 สถานะการทำงานของเครื่อง

สวิตช์ที่ 6, 7 และ 8 ของดิพสวิตช์ใช้กำหนดลักษณะการทำงานของเครื่อง

สวิตช์ที่ 5 ของดิพสวิตช์ตัวเดียวกันใช้กำหนดสถานะของเอาต์พุต เมื่อเกิดความผิดพลาด

1.3 อินพุต

กำหนดด้วยสวิตช์ที่ 1 - 4 ของดิพสวิตช์ตัวเดียวกัน

ชนิดของอินพุต : ขึ้นอยู่กับชนิดของโมเดล

Model TC : Type B, E, J, K, R, S, T

Model mA : 4-20 mA DC และแยกโดดสัญญาณ (Isolated)

1.4 เอาต์พุต

สัญญาณ : 4-20 mA DC และแยกโดยสัญญาณ ทั้งโมเดล TC และ mA

ปริมาณโหลด : 500 Ohm max.

1.5 การเตือน (Alarm)

เอาต์พุต : Optocoupler 250 mA max. จำนวน 2 จุดสำหรับ Normal alarm และ Severe alarm

1.6 แหล่งจ่ายไฟขาเข้า : 24 V DC

1.7 การใช้กำลังงานไฟ : 5 Watts max.

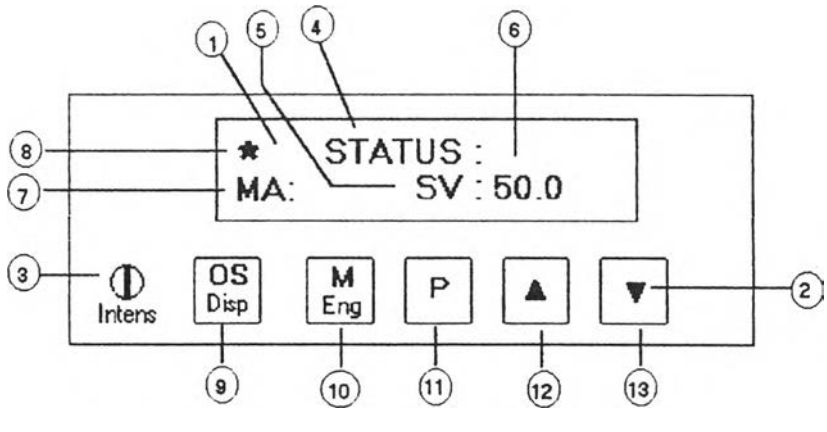
## 2. ช่วงการทำงานของอินพุตตามชนิดของ TC (โมเดล TC เท่านั้น)

| ชนิดของ TC | ช่วงอุณหภูมิการทำงาน (องศา C) | หมายเหตุ |
|---|---|---|
| B | 0- 1800 | ช่วงใช้งานเหมาะสม 220 - 1800 |
| E | 0- 1000 | |
| J | 0- 760 | |
| K | 0- 1370 | |
| R | 0- 1760 | |
| S | 0- 1760 | |
| T | 0- 400 | |

## 3. โครงสร้างของแผงหน้าปัดของตัวควบคุม

3.1 ลักษณะของแผงหน้าปัด (Front Panel)

ลักษณะของแผงหน้าปัดแบ่งตามลักษณะการแสดงผลได้ 2 แบบ




รูปที่ 1 แสดงแผงหน้าปัดขณะแสดงผลแบบแสดงค่าข้อมูล

(1) ภาคแสดงผลแบบ LCD

(2) ภาครับข้อมูล (ปุ่มรับข้อมูล)

(3) ปุ่มปรับความเข้มของภาคแสดงผล

(4) ส่วนแสดงชื่อกลุ่มของตัวพารามิเตอร์

(5) ส่วนแสดงชื่อของตัวพารามิเตอร์

(6) ส่วนแสดงค่าของพารามิเตอร์

(7) ส่วนแสดงลักษณะเอาต์พุตของการควบคุมแบบ MANUAL

(8) ส่วนแสดงผลการเตือนแบบ Normal Alarm (*) และ Severe alarm (**)

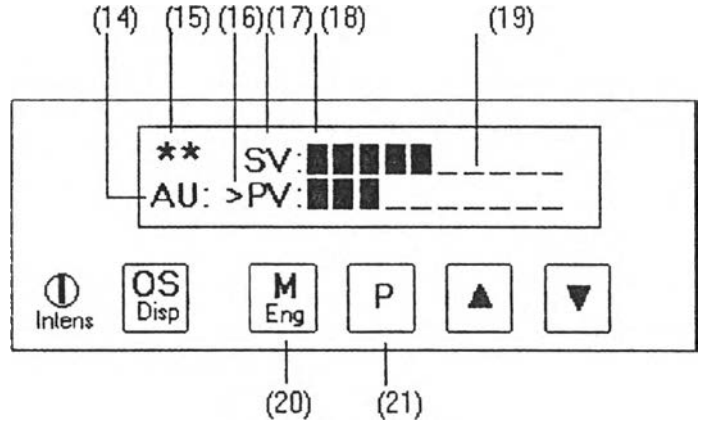
(9) ปุ่มเลือกลักษณะสัญญาณเอาต์พุต (ฟังก์ชัน OS) / เปลี่ยนลักษณะการแสดงผล (ฟังก์ชัน Disp)

(10) ปุ่มเลือกกลุ่มตัวแปร (ฟังก์ชัน M) / เลือกกลุ่ม ENG (ฟังก์ชัน Eng)

(11) ปุ่มเลือกตัวแปร

(12) ปุ่มเพิ่มค่าตัวแปร (Increment)

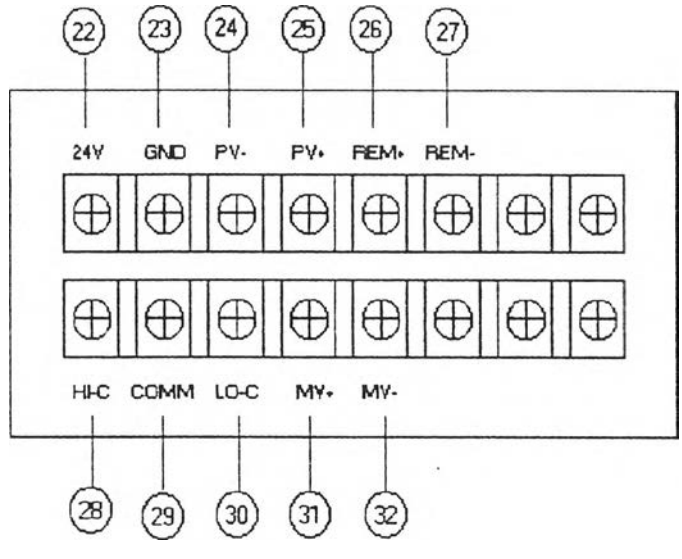
(13) ปุ่มลดค่าตัวแปร (Decrement)




รูปที่ 2 แสดงแผงหน้าปัดขณะแสดงผลแบบแสดงกราฟของตัวแปรโปรเซส

- ลักษณะเอาต์พุตของการควบคุม โดยแสดงตัวอย่างเป็นแบบ Auto (14)
- การแสดงผลการเตือน โดยแสดงตัวอย่างเป็นระบบแบบ Severe alarm (15)
- ตัวบอกแถว ( Prompt ) ที่ใช้งานปัจจุบัน (16)
- ชื่อของตัวแปรในโปรเซส (17)
- เซลล์ของกราฟแท่งที่แสดงค่า 10 % ของค่าเต็มสเกล (18)
- เซลล์ของกราฟแท่งที่แสดงส่วนเติมเต็มของกราฟ (19)
- ปุ่มสำหรับเปลี่ยนแถวที่สนใจ (ฟังก์ชัน M) (20)
- ปุ่มเลือกตัวแปรโปรเซสในแถวที่สนใจ (ฟังก์ชัน P) (21)

## 4. ลักษณะของเครื่องด้านหลัง (Rear Panel)




รูปที่ 3 แสดงลักษณะของเครื่องด้านหลัง

(22) แหล่งจ่ายไฟ ขั้ว 24 V
(23) แหล่งจ่ายไฟ ขั้ว Ground
(24) สัญญาณ PV ขั้ว -
(25) สัญญาณ PV ขั้ว +

(26) สัญญาณ SV ขั้ว +

(27) สัญญาณ SV ขั้ว -

(28) หน้าสัมผัสการเตือน Severe Alarm (High boundary contact)

(29) หน้าสัมผัสการเตือนร่วม (Common)

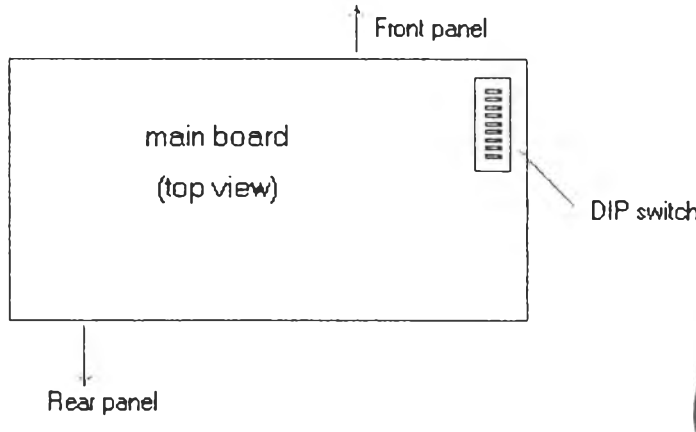
(30) หน้าสัมผัสการเตือน Normal Alarm (Low boundary contact)

(31) สัญญาณกระแส MV ขั้ว +

(32) สัญญาณกระแส MV ขั้ว -

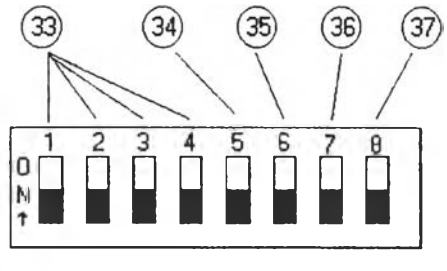
**5. ดิพสวิตช์เพื่อกำหนดสถานะของตัวควบคุม**

ใช้กำหนดลักษณะของอินพุต (ทั้งโมเดล TC และ โมเดล mA) ลักษณะของเอาต์พุต เมื่อตัวควบคุมทำงานผิดพลาด และกำหนดสถานะการทำงานของตัวควบคุมตำแหน่งของดิพสวิตช์จะอยู่บนบอร์ดหลัก รูปที่ 4 และรูปที่ 5 แสดงตำแหน่งและหน้าที่ของดิพสวิตช์ดังกล่าว






รูปที่ 4 แสดงตำแหน่งดิพสวิตช์บนบอร์ดหลัก




รูปที่ 5 แสดงหน้าที่ของดิพสวิตช์ของตัวควบคุม

(33) ชุดสวิตช์เลือกชนิดของอินพุต

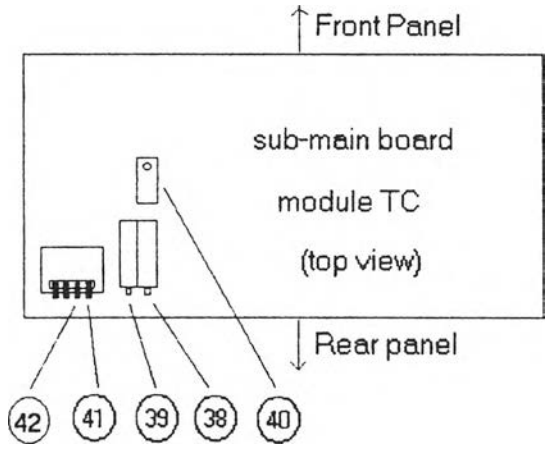
(34) สวิตช์สำหรับกำหนดค่าเอาต์พุตขณะเครื่องทำงานผิดพลาด

(35) สวิตช์เลือก PID1/PID2

(36) สวิตช์เลือกที่มาของสัญญาณ SV ( Remote/Local )

(37) สวิตช์เลือก Direct/Reverse

## 6. Signal conditioner (สำหรับโมเดล TC เท่านั้น)




รูปที่ 6 แสดงวงจร Signal conditioner

(38) ตำแหน่งปรับตั้งค่าศูนย์ ( Zero adj.)

(39) ตำแหน่งปรับตั้งช่วงการทำงาน ( Span adj.)

(40) ตำแหน่งปรับตั้งค่าชดเชย Cold junction (Cold junction compensation adj.)

(41) สวิตช์ลัดวงจรสำหรับการปรับค่าศูนย์ ( Zero adj. short sw.)

(42) สวิตช์ลัดวงจรสำหรับการปรับค่าชดเชย Cold junction ( Cold junction adj. short sw.)

**7. การกำหนดลักษณะของสัญญาณ SV**

กำหนดโดยสวิตช์ที่ 7 ของดิพสวิตช์ (ดูรูปที่ 5) เมื่อ

ON ------ REMOTE SV

OFF ------- LOCAL SV

**8. การกำหนดลักษณะของ DV**

กำหนดโดยสวิตช์ที่ 8 ของดิพสวิตช์ตัวเดียวกัน (ดูรูปที่ 5) เมื่อ

ON ------ Direct DV ( DV= SV-PV)

OFF ------ Reverse DV (DV = PV-SV)

**9. การกำหนดเลือกการควบคุมแบบ PID**

กำหนดโดยสวิตช์ที่ 6 ของดิพสวิตช์ (ดูรูปที่ 5) เมื่อ

ON ---- ฟังก์ชัน PID2 สำหรับการควบคุมที่ค่าเป้าหมาย (SV) เปลี่ยนแปลงตามเวลา

OFF ---- ฟังก์ชัน PID1 สำหรับการควบคุมที่ค่าเป้าหมาย (SV) มีค่าคงที่

## 10. การกำหนดดิพสวิตช์เลือกชนิดของอินพุต

### 10.1 สำหรับโมเดล TC

เพื่อกำหนดชนิดของเทอร์โมคัปเปิล 7 ชนิดแสดงได้ดังตาราง

| ตำแหน่งของสวิตช์ที่ | | | | TC type | หมายเหตุ |
|---|---|---|---|---|---|
| 4 | 3 | 2 | 1 | | |
| ON | ON | ON | ON | B | โมเดล TC เท่านั้น |
| ON | ON | ON | OFF | E | "----------" |
| ON | ON | OFF | ON | J | "----------" |
| ON | ON | OFF | OFF | K | "----------" |
| ON | OFF | ON | ON | R | "----------" |
| ON | OFF | ON | OFF | S | "----------" |
| ON | OFF | OFF | ON | T | "----------" |

### 10.2 สำหรับโมเดล mA

สวิตช์ตำแหน่งที่ 4 จะต้องอยู่ในตำแหน่ง OFF เสมอ ดังนี้

| | | | | | |
|---|---|---|---|---|---|
| OFF | X | X | X | Standard current | โมเดล mA เท่านั้น |

## 11. การกำหนดดิพสวิตช์เลือกสถานะของเอาต์พุตเมื่อเกิดความผิดพลาด

เพื่อกำหนดลักษณะของเอาต์พุตเมื่อตัวควบคุมทำงานผิดพลาด โดยกำหนดด้วยสวิตช์ที่ 5 ของดิพสวิตช์ โดยที่

ON -- เอาต์พุต ( MV ) มีค่าเป็น 0 % เมื่อตัวควบคุมทำงานผิดพลาด

ON -- เอาต์พุต ( MV ) มีค่าเป็น 100 % เมื่อตัวควบคุมทำงานผิดพลาด

12. การใช้งานแผงหน้าปัด

การใช้งานโดยทั่วไป จะเป็นการป้อนหรือเปลี่ยนแปลงค่าพารามิเตอร์เปลี่ยนลักษณะการแสดงผล กำหนดลักษณะการควบคุม และการปรับแต่งสัญญาณสำหรับส่วนรับข้อมูลของแผงหน้าปัดเป็นปุ่มจำนวน 5 ปุ่ม ที่ประกอบด้วยฟังก์ชันหลัก (แสดงด้วยสีน้ำเงิน) และฟังกชันพิเศษ (แสดงด้วยสีขาว) รายละเอียดของแต่ละปุ่มเป็นดังนี้

OS Disp กำหนดลักษณะการควบคุม / เลือกลักษณะของการแสดงผล

ฟังก์ชันหลัก -- OS (Output Seletion) สำหรับเลือกที่มาของสัญญาณ MV และลักษณะการควบคุมว่าเป็นแบบ Auto (AU) หรือ Manual (MA)

ฟังก์ชันพิเศษ -- Disp (Display) เลือกลักษณะของการแสดงผลกราฟแท่ง (Bar graph display) หรือแสดงค่าข้อมูลเพื่อการแก้ไข (Data and modified display) อย่างใดอย่างหนึ่ง

M Eng เลือกโหมดหรือกลุ่มตัวแปรที่สนใจ เปลี่ยนแถวของกราฟแท่ง / เลือกโหมด Eng

ฟังก์ชันหลัก -- M (Mode) ขึ้นอยู่กับลักษณะการแสดงผล

สำหรับการแสดงค่าข้อมูล -- เลือกกลุ่มของตัวแปรหรือพารามิเตอร์ ได้แก่ STATUS, ALARM และ TUNING

สำหรับการแสดงผลกราฟแท่ง -- เลือกกราฟที่สนใจ

ฟังก์ชันพิเศษ -- Eng (Engineering Mode) ใช้กับการแสดงค่าข้อมูลเท่านั้น

เพื่อตั้งช่วงอุณหภูมิการทำงานของเทอร์โมคัปเปิล (โมเดล TC เท่านั้น) ได้แก่ตัวแปร T_MIN, T_MAX (ส่วนโมเดล mA ตัวแปรทั้งสองจะเป็น 0) และใช้สำหรับปรับแต่งสัญญาณเอาต์พุตที่ 0% และที่ 100% ได้แก่ตัวแปร OP_CAL <0%>, OP_CAL <100%> เพื่อให้สมนัยกับกระแสมาตรฐาน 4-20 mA

P Parameter การใช้งานขึ้นอยู่กับลักษณะการแสดงผล

สำหรับการแสดงผลแบบแสดงค่าข้อมูล เป็นการเลือกตัวแปรหรือพารามิเตอร์ภายใต้โหมดการทำงานหนึ่งๆ

สำหรับการแสดงผลเป็นกราฟแท่ง เป็นการเลือกตัวแปรใน

โปรเซสมาแสดงผล

▲ ใช้งานภายใต้การแสดงผลแบบแสดงค่าข้อมูลเท่านั้น

ฟังก์ชันหลัก – เพิ่มค่าตัวแปรหรือพารามิเตอร์ ทีละ 1 หน่วย

ฟังก์ชันพิเศษ -- เพิ่มค่าตัวแปรหรือพารามิเตอร์อย่างรวดเร็ว

▼ ใช้งานภายใต้การแสดงผลแบบแสดงค่าข้อมูลเท่านั้น

ฟังก์ชันหลัก – ลดค่าตัวแปรหรือพารามิเตอร์ทีละ 1 หน่วย

ฟังก์ชันพิเศษ -- ลดค่าตัวแปรหรือพารามิเตอร์อย่างรวดเร็ว

note สำหรับปุ่ม ▲ และ ▼ ค่าที่เปลี่ยนแปลงจะบันทึกเข้าเครื่องหลังจากกดปุ่ม M Eng อีกครั้งหนึ่ง

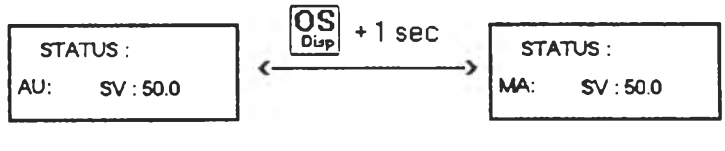
สำหรับวิธีการใช้งานฟังก์ชันของปุ่ม

ฟังก์ชันหลัก – กดปุ่มแล้วปล่อยทันที

ฟังก์ชันพิเศษ -- กดปุ่มนานกว่า 1 วินาที

## 13. การเลือกลักษณะของสัญญาณ MV

การเลือกลักษณะของสัญญาณ MV เพื่อกำหนดเอาต์พุตที่ได้จากการควบคุมแบบอัตโนมัติ (Auto) หรือกำหนดโดยผู้ใช้ (Manual)




" Auto " " Manual "

Controller defines output User defines output

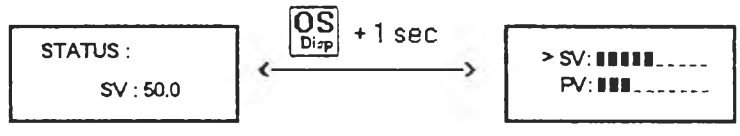
## 14. การเปลี่ยนลักษณะการแสดงผล

การแสดงผลของตัวควบคุมสามารถแสดงผลได้ 2 แบบ คือ

14.1 แสดงค่าข้อมูลเพื่อการแก้ไข (Data and Modified Display)

14.2 แสดงค่ากราฟแท่งของตัวแปรในโปรเซส ( Bar Display)

ในการเปลี่ยนลักษณะการแสดงผลทำได้โดยกด [OS Disp] นานกว่า 1 วินาที เพื่อเลือกลักษณะการแสดงผล

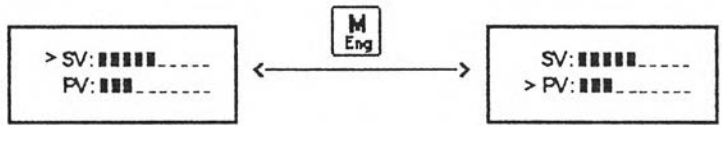



## 15. การใช้ปุ่มเมื่อการแสดงผลเป็นการแสดงกราฟแท่ง

ใช้สำหรับแสดงผลตัวแปร SV, PV, DV และ MV โดยแสดงได้ครั้งละ 2 กราฟ ในขณะเดียวกัน สำหรับ SV, PV และ MV มีความละเอียดในการแสดงผล 10% ต่อ 1 ช่องตัวอักษร DV มีความละเอียด 20% ต่อ 1 ช่อง สำหรับการใช้งานปุ่มเป็นดังนี้

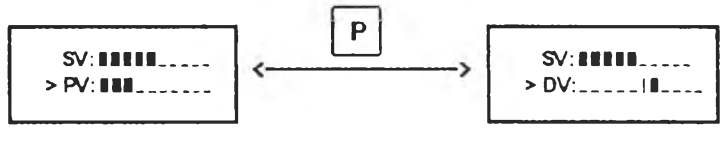
กด [M Eng] เพื่อเลือกแถวที่ต้องการ

ตัวอย่าง




กด [P] เลือกตัวแปรในการแสดงผล

ตัวอย่าง



## 16. การปรับตั้งค่าตัวแปร

กระทำได้โดยลักษณะการแสดงผลจะต้องเป็นการแสดงผลแบบแสดงค่าข้อมูลเพื่อการแก้ไขเท่านั้น ตัวแปรหรือพารามิเตอร์ดังกล่าวได้แสดงในตาราง ก ซึ่งจะแสดงค่าของตัวพารามิเตอร์ และคุณสมบัติภายในโหมดต่างๆ

## 17. การปรับตั้งค่าตัวแปรในโปรเซส

ตัวแปรในโปรเซสที่สามารถตั้งค่าได้คือ SV และ MV

17.1 การตั้งค่า SV

ตั้งค่า SV ได้ภายใต้ฟังก์ชัน OS และ DIP Switch เลือกไว้ที่ "Local "

กด [M Eng] เลือก STATUS Mode

กด [P] เลือกตัวแปร SV

กด [▲] หรือ [▼] ในกรณีที่ต้องการเพิ่มหรือลดค่า หรือ

กดนานกว่า 1 วินาที ในกรณีที่ต้องการเพิ่มหรือลดค่าอย่างรวดเร็ว

กด [M Eng] อีกครั้งหนึ่งเพื่อบันทึกค่าเข้าไป

| Mode | Parameter | ลักษณะข้อมูล | รายละเอียด |
|---|---|---|---|
| STATUS | SV | 0-100% | 1 |
| (Variable | PV | " | 2 |
| in process) | DV | " | 2 |
| | MV | " | 3 |
| | AL | 0 - 100%, | Normal Alarm Lower set |
| | AH | 0 - 100%, | Normal Alarm Higher set |
| ALARM | | | |
| (PV alarm set) | LL | 0 - 100%, | Severe alarm Lower set |
| | HH | 0 - 100% | Severe alarm Higher set |
| | PB | 1 - 200% | Proportional Band |
| TUNING | | | |
| (PID tuning) | TI | 0 - 3600 วินาที | Integral Time |
| | TD | 0 - 1200 วินาที | Differential Time |
| | T_MIN | ขึ้นอยู่กับชนิดของเซนเซอร์ | อุณหภูมิต่ำสุด |
| ENG | T_MAX | ขึ้นอยู่กับชนิดของเซนเซอร์ | อุณหภูมิสูงสุด |
| | OP<0%> | ไม่แสดงค่า | ค่าศูนย์ของเอาต์พุต |
| | OP<100%> | ไม่แสดงค่า | ค่าช่วงของเอาต์พุต |

ตาราง ก แสดงค่าตัวพารามิเตอร์และคุณสมบัติในโหมดต่างๆ

**note**

1 ดิพสวิตช์ต้องเลือกไว้ที่ Local

2. สำหรับแสดงค่าตัวแปรเท่านั้นปรับตั้งค่าไม่ได้

3. ปรับตั้งค่าได้ในกรณีที่ [OS Disp] เลือกไว้ที่ Manual

17.2 การตั้งค่า MV

ตั้งค่า MV ได้ในกรณีที่ Output Selection เลือกเป็น Manual

กด [M Eng] เลือก STATUS mode

กด [P] เลือกตัวแปร MV

กด [▲] หรือ [▼] ในกรณีที่ต้องการเพิ่มหรือลดค่า หรือ

กดนานกว่า 1 วินาที ในกรณีที่ต้องการเพิ่มหรือลดค่าอย่างรวดเร็ว

กด [M Eng] อีกครั้งหนึ่งเพื่อบันทึกค่าเข้าไป

18. **การตั้งค่าการเตือน (Alarm)**

การตั้งค่าการเตือนเป็นการตั้งค่าช่วงการทำงานของ PV ที่ยอมรับได้ ภายใต้ช่วงการทำงานที่ผู้ใช้ยอมรับ เมื่อ PV ออกนอกช่วงที่กำหนด จะเกิดการเตือนขึ้นใน 2 ลักษณะ คือ Normal alarm และ Severe alarm ดังแสดงในรูปที่ 7

ขั้นตอนการปรับตั้งค่า Alarm

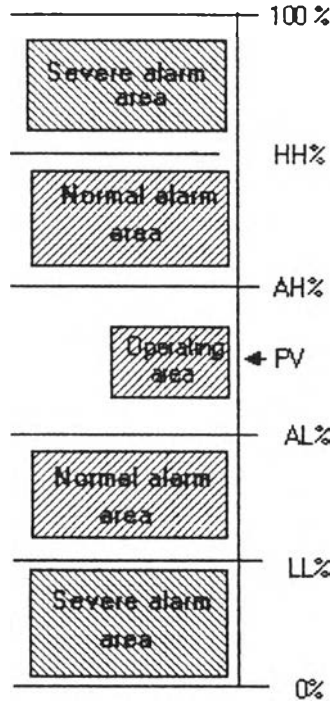
กด [M Eng] เลือกจนกว่าจะแสดงผลกลุ่มตัวแปรเป็น ALARM

กด [P] เลือก AL, AH, HL หรือ HH ตามต้องการ

กด [▲] หรือ [▼] ในกรณีที่ต้องการเพิ่มหรือลดค่า หรือ

กดนานกว่า 1 วินาที ในกรณีที่ต้องการเพิ่มหรือลดค่าอย่างรวดเร็ว

กด [M Eng] อีกครั้งหนึ่งเพื่อบันทึกค่าเข้าไปในเครื่อง




รูป 7 แสดงลักษณะการทำงานของ PV ภายใต้ช่วงที่กำหนด

เอาต์พุตของการเตือนจะแสดงออกทางส่วนแสดงผลและทางฮาร์ดแวร์ สำหรับส่วนแสดงผล Normal alarm จะแสดงด้วยสัญลักษณ์ "*" และ Severe alarm แสดงด้วยสัญลักษณ์ "**" ที่กระพริบด้วยความถี่ 1 เฮิรตซ์ สำหรับฮาร์ดแวร์เทอร์มินอล Lo_co หรือ Hi_co (ดูรูปที่ 3) จะอยู่ในสภาวะ "ON"

## 19. การปรับตั้งค่าพารามิเตอร์ของการควบคุมแบบ PID

เพื่อปรับค่าพารามิเตอร์ของการควบคุมแบบ PID

กด [M Eng] เลือกจนกว่าจะแสดงผลกลุ่มตัวแปรเป็น TUNING

กด [P] เลือกตัวแปรที่สนใจ PB, $T_I$ หรือ $T_D$

กด [▲] หรือ [▼] ในกรณีที่ต้องการเพิ่มหรือลดค่า หรือ กดนานกว่า 1 วินาที ในกรณีที่ต้องการเพิ่มหรือลดค่าอย่างรวดเร็ว

กด [M Eng] เพื่อบันทึกค่าเข้าไป

**20. การปรับตั้งค่าอุณหภูมิการทำงาน (Range Setting) ของเซนเซอร์ (สำหรับโมเดล TC)**

โดยปกติโปรแกรมจะตั้งค่าช่วงการทำงานของเซนเซอร์เป็นค่าสูงสุด ต่ำสุดของเซนเซอร์แต่ละชนิดไว้อยู่แล้ว ขึ้นอยู่กับการเลือกชนิดเซนเซอร์จากดิพสวิตช์แต่ผู้ใช้สามารถกำหนดช่วงการใช้งานจริง (Working range) ให้แคบลงอีกได้

กด [M Eng] ค้างนานกว่า 1 วินาที

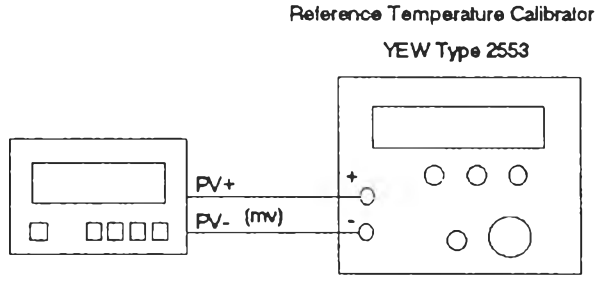
กด [P] เลือกตัวแปรที่สนใจ T_MIN หรือ T_MAX

กด [▲] หรือ [▼] ในกรณีที่ต้องการเพิ่มหรือลดค่าอุณภูมิตามที่ต้องการ หรือกดนานกว่า 1 วินาที ในกรณีที่ต้องการเพิ่มหรือลดค่าอย่างรวดเร็ว

กด [M Eng] อีกครั้งหนึ่งเพื่อบันทึกค่าเข้าไป

ในการปรับตั้งค่าอุณหภูมิของเซนเซอร์ ผู้ใช้จะต้องปรับค่าช่วงการทำงานของ Signal conditioner ด้วยวิธีการปรับตั้งค่าเป็นดังนี้

1. ต่อวงจรดังรูป




2. กดดิพสวิตช์ สำหรับ Cold junction และ Zero adj. ให้ "ON"
3. ป้อนแรงดันอ้างอิงที่สมนัยกับอุณหภูมิ T_MAX
4. ปรับ Span adj. ให้อ่านค่า PV ได้ 75 %
5. กดดิพสวิตช์สำหรับ Zero adj. ให้ "OFF"

6. ปรับ Zero adj. ให้อ่านค่าได้ 100 %

7. ปรับ Generator ให้ได้แรงดันเสมือนปลายด้านหนึ่งของ TC วัดที่อุณหภูมิสภาวะแวดล้อม

8. เปิดวงจร Cold junction จากนั้นปรับค่าจนอ่านค่า PV ได้ 100 %

## 21. การปรับแต่งเอาต์พุต

เป็นการปรับแต่งค่า MV ที่ 0-100% ให้สมนัยกับ 4-20 มิลลิแอมป์ วิธีการปรับเทียบ มีดังนี้

กด [OS Disp] เพื่อเลือก MV ให้เป็น "Manual"

กด [M Eng] และเลือก "Status"

กด [P] และเลือก MV

กด [▼] เพิ่มค่า MV จนเป็น 100%

กด [M Eng] เพื่อบันทึกค่าข้อมูลเข้าเครื่อง

กด [M Eng] ค้างนานกว่า 1 วินาที เพื่อเลือก Engineering mode

กด [P] และเลือก OP_CAL <100%>

กด [▲] หรือ [▼] แล้วปรับค่าให้ได้เอาต์พุตเป็น 75 % หรือ 16 มิลลิแอมป์

ทำตามขั้นตอนที่ - โดยปรับ MV ให้มีค่าเป็น 0%

ทำตามขั้นตอนที่ - โดยเลือกตัวแปร OP_CAL <0%>

กดเพิ่มค่าให้กับเอาต์พุตจนอ่านค่าได้ 20 มิลลิแอมป์

## 22. การรายงานความผิดพลาดของเครื่อง

การทำงานที่ผิดพลาดของเครื่องจะเกิดจากวินิจฉัยฮาร์ดแวร์ภายในของตัวควบคุม ตาราง ข จะแสดงรายละเอียดของการทำงานผิดพลาด

| ชื่อ | อธิบายคุณลักษณะขณะผิดพลาด |
|---|---|
| "Watchdog count out" | ระยะเวลาของการทำงานของตัวควบคุมนานมากกว่าระยะเวลาอยู่ 15 วินาที |
| "RAM check error" | RAM ไม่สามารถอ่านเขียนได้ในบางตำแหน่ง |
| "A/D fail" | ส่วนแปลงผันข้อมูลแอนะลอกเป็นค่าเชิงเลขผิดพลาด |
| "I/P loose" | อินพุตมีค่าออกนอกช่วง 0 - 100% |

ตาราง ข แสดงรายละเอียดการทำงานที่ผิดพลาดของตัวควบคุม

# ภาคผนวก ข

## โปรแกรมตัวควบคุม PID เชิงเลขขนาดกะทัดรัด

```
/*
   BACKGND.C51 -- Background task
   External -- var : timecount
               func: -
   Descript --
*/

#pragma CODE SYMBOLS DEBUG  /* compiler options */

/* --> header file <-- */
#include <STDIO.H>
#include <ABSACC.H>
#include "83C51FA.H"
#include "GLOBAL.H"
#include "PID.H"
#include "LCD.H"
#include "BACKGND.H"

/* --> variable defining <-- */
  /* --> byte defining */
  data unsigned char  wd_flag; /* watchdog count out flag */
  data unsigned char wd_count; /* watchdog count out counter */
  /* --> bit type */
  data bit evtimer_on,   /* event time turn on flag */
           bar_flag,   /* display bar flag */
             update,   /* data need display flag */
           fast_flag,   /* keyboard fast flag */
             ad_test,   /* A/D test flag */
            done1sec,   /* specital key doing one time flag */
            err_flag,   /* error report require display */
           swap_flag;   /* display swapping flag */

  /* --> structure type */
  pstruct vtab[] = {
    {"STATUS:", 4, "    SV:","    PV:","    DV:","    MV:", "%4d.%1d",
      {RDWR,READ,READ,READ}, {0,0,0,0}, {1000,1000,1000,1000}},
    {" ALARM:", 4, "    AL:","    AH:","    LL:","    HH:", "%4d.%1d",
      {RDWR,RDWR,RDWR,RDWR}, {0,0,0,0}, {1000,1000,1000,1000},
      {0,0,99.8,998,0,0,99.8,998}},
    {"TUNING:", 3, "    PB:","    TI:","    TD:","    BI:", "%-6d",
      {RDWR,RDWR,RDWR,RDWR}, {1,1,0,0}, {200,3600,1200,100},
      {100,100,3600,3600,0,0,0,0}},
    {"   ENG:", 4, " T_MIN:"," T_MAX:","OP_CAL:","OP_CAL:", "%-6d",
      {RDWR,RDWR,RDWR,RDWR},{0,0,0,0}, {3000,3000,3277,3277}}
   };

  /* --> integer type */
    int  tmp[MAXDPAR];   /* temporary use for PID calculation */

main () {
  data unsigned char  i, j;     /* general counter */
  /* --> Single rounded main program <-- */
```

```
  /* --> power on delay */
  for(i=0; i<255; i++)
    for(j=0; j<255; j++);
  /* --> watchdog checking */
  if(wd_flag==WD_SET) {
    if(++wd_count==COUNT_OUT_DAT) {
      err_type = WATCHDOG;
      err_set();
      send_output();
      update = OK;              /* require display */
      display_routine();
      while(1);
    }
  }
  else
    wd_flag  = wd_count = RESET;
  /* --> initialized display */
  init_lcd();
  /* --> Start-up time diagonostic <-- */
  /* --> 2k RAM checked */
  STARTADD = 0x0;
  ENDADD   = 0x7ff;
  if(xramchk()) {
    err_type = NV_RAM;
    err_set();
    update = OK;
    display_routine();
    while(1);
  }
  /* --> Non-Volatile event checking */
  if(nv_flag!=NONVOLATILED) {
    init_max_engmode();           /* initialized RAM */
    nv_flag = NONVOLATILED;       /* SET flag for next test */
  }
  else
    restore_vtab();               /* restore last data */
  /* --> Input type checking */
  if(SENSOR_TYPE > 6 ) {
      T_MIN_FLAG = READ;
      T_MAX_FLAG = READ;
      T_MIN_I = T_MAX_I = 0;
      T_MIN_F = T_MAX_F = 0;
  }
  else {  /* input is sensor */
    /* --> Check type of used sensor */
    if(sens_buf != SENSOR_TYPE) {  /* Initialize parameters for the new
one */
      init_max_engmode();
      sens_buf = SENSOR_TYPE;
    }
```

```
    mode_change = cal_sens_parameter(); /* pre-calculate sensor
condition */
  }
  /* --> Read SV source type (remote/local sw (P1B7)) */
  LOCAL_FLAG = (LOCAL_SW)? RDWR:READ;

                  /* PID variable initilized */
  /* --> restore display buffer */
  for(i=0;i<MAXDPAR; i++)
    tmp[i] = vtab[STATUS].par[i].ipar;
  /* --> clear variables */
  sv_1 = SV_F;
  INTE = INTE_1 = 0;     /* Integral term */
  dv_1 = DV_F;
  DER  = DER_1  = 0;     /* Difference term */
  pv_1 = PV_F;
  /* --> Pre-calculate PID gain */
  store_int2float(TUNING,vtab[TUNING].nparm,1.0);
  mode_change = cal_coeff();
  /* --> Set display to data display */
  ddisplay = DMODE_CHANGE;
  update = OK;
  /* --> Timer setting */
  init_timebase();              /* timebase (250 usec)    */
  init_event_timer();           /* event timer (1 sec)    */
  init_system_timer();          /* system timer (sampling timer) (0.1
sec) */
  /* --> Set general purpose variable */
  i = 0;                        /* counter reset */

  /* --> Rounded type main program <-- */
  do {
    /* --> User-input interface routine */
    keyboard_routine();
    /* --> User-output interface routine */
    display_routine();
    /* --> Convert user interface data to controller data */
    switch (mode_change) {
      case ALARM_CHANGE :
        store_int2float(ALARM,vtab[ALARM].nparm,0.1);
        mode_change =-1;
        break;
      case TUNING_CHANGE :
        store_int2float(TUNING,vtab[TUNING].nparm,1.0);
        mode_change = cal_coeff();  /* recalculate PID coefficient */
        break;
      case EMODE_CHANGE:
        mode_change = cal_sens_parameter();
        break;
    }
    /* --> Run-time diagonostic <-- */
    /* --> Check A/D */
```

```
    if(ad_test) {
      ad_test = NOK; /* protect interrupt function */
      ad_min_val = READ_AD(MIN_CHAN);
      ad_max_val = READ_AD(MAX_CHAN);
      if(!ad_test) {
        if((ad_min_val>=0x31b)&&(ad_min_val<=0x33b)) { /* check min */
          if(ad_max_val<0xff0) { /* check max */
            ad_max_val = 0;      /* clear data */
            ++i;                 /* increment if error */
            if(i==20) {          /* displays message if arrive 20 counts
*/
              i=0;
              err_type = AD_FAIL;
              err_set();
              update = OK;
            }
          }
          else {                 /* no max error */
            i=0;                 /* clear counter */
            err_flag = RESET;
          }
        }
        else {                                          /* min error */
          ad_min_val = 0;        /* clear data */
          ++i;                   /* increment if error */
          if(i==10) {            /* display message if arrive 10 counts
*/
            err_type = AD_FAIL;
            err_set();
            update = OK;
          }
        }
      }
    }
  } while (1);
}
```

```
/*
  CALCOEF.C51 -- Calculate PID coefficient
*/

#pragma  CODE SYMBOLS DEBUG                /* compiler options */

/* --> header files <-- */
#include "GLOBAL.H"           /* global define */
#include "PID.H"              /* controller define */
#include "CALCOEF.H"          /* its header */

/* --> calculate difference coefficient */
unsigned char busy; /* flag */

unsigned char cal_coeff() {
  float gamma;       /* gamma = Td/Ts */
  unsigned char i;  /* counter */

  gamma = Td_F*(1.0/TS);
  /* --> save to their mirror storages at first */
  for(i=0; i<4; i++)
    pid_kd[i] = pid_k[i];
  /* --> signal to the caller that busy */
  busy  = BUSY;
  /* --> calculate PID gain */
  KP    = 100/PB_F;
  KI    = 0.5*TS*KP/Ti_F;
  KD1   = Kd1_NUMERATOR/Kd_DENOMINATOR;
  KD2   = Kd2_NUMERATOR/Kd_DENOMINATOR;
  busy  = NBUSY;
  return -1;
}
```

```
/*
  DISPLAY.C51 -- Display routine
*/

#pragma  CODE SYMBOLS DEBUG /* compiler options */

/* --> header file <-- */
#include <STDIO.H>          /* standard i/o */
#include "83C51FA.H"        /* CPU register definitions */
#include "GLOBAL.H"         /* global define */
#include "PID.H"            /* controller define */
#include "LCD.H"            /* LCD display define */
#include "DISPLAY.H"        /* its header file */

/* --> function declaration */
  void alarm_echo();     /* concurrent alarm display on screen */
  void display_error();  /* error report */

/* --> variable definition */
  /* --> alarm swapping in display control variable */
    data bit            ala_swap;
    unsigned char ala_period_cnt;
  /* --> bar  variable */
    unsigned char bhead[MAXBMODE];    /* bar header control variable */
  /* --> error message buffer */
    char *err_msg[4]= { "    NV RAM fail     ",
                        "  Watchdog count out",
                        "     I/P Loose      ",
                        "      A/D fail      "
                      };
    unsigned char err_type; /* error type number */
    unsigned char err_disp; /* display error if set */


display_routine() {

  alarm_echo();                    /* display user alarm */
  if(update) {
    if(err_flag) {
      display_error(err_type);
      err_disp = SET;
      ddisplay = DPAR_CHANGE;
    }
    else {
      if(err_disp) {
        err_disp = RESET;
        CLEAR_DISPLAY;
        ddisplay = DMODE_CHANGE;
      }
    }
    if(!swap_flag) {  /* function "OS" */
      DISPLAY_CONTROL(0xc);          /* turn off cursor */
```

```
      MOV_CURSOR(MAN_LOC);            /* display auto/manual */
      if(MAN_FLAG)
        printf("MA:");
      else
        printf("AU:");
        if(bar_flag) {      /* bar display */
          if(bmode) {       /* prompt on row 2 */
            bhead[0] = display_bar(bpar1, ROW1, bhead[0], INERT);
            bhead[1] = display_bar(bpar2, ROW2, bhead[1], ACTIVE);
          }
          else {
            bhead[0] = display_bar(bpar1, ROW1, bhead[0], ACTIVE);
            bhead[1] = display_bar(bpar2, ROW2, bhead[1], INERT);
          }
        }
        else                 /* data and modified display */
          display_data();
      update = NOK;
    }
    else {   /* function "Disp" */
      if(bar_flag)  /* bar display */
        RESTORE_BAR;
      else {         /* data display */
        SAVE_BAR;
        CLEAR_DISPLAY;
        ddisplay = DMODE_CHANGE;
      }
      swap_flag = RESET;
    }
  }
}


void alarm_echo() {

  MOV_CURSOR(ALA_LOC); /* set to alarm warning position */
  switch(P3¦0xe7) {    /* determine alarm event */
    case SEVERE_NORMAL:
    case SEVERE_ALARM:
      if(ala_swap)
        printf("  ");
      else
        printf("**");
      break;
    case NORMAL_ALARM:
      printf("* ");
      break;
    default :
      printf("  ");
  }
}
```

```
void display_error(unsigned char err_nbr) {
  MOV_CURSOR(HOME);
  printf(err_msg[err_nbr]);
}
```

```
/*
  DISP_BAR.C51 -- DISPLAY BAR ROUTINE
*/

#pragma  CODE SYMBOLS DEBUG  /* compiler options */

/* --> header files <-- */
#include <STDIO.H>    /* standard function call */
#include "LCD.H"      /* user output display */
#include "GLOBAL.H"   /* general declaration */
#include "PID.H"      /* PID declaration */
#include "DISP_BAR.H" /* its header */

/* --> variables defining <-- */
static unsigned char bar_buffer[MAXBARCELL] =
 {"    SV:__________       SV:__________   "};


unsigned char display_bar(bpar, row, head_flag, status)
  unsigned char      bpar;  /* parameter counter */
  unsigned char       row;  /* th row to show */
  unsigned char head_flag;  /* head echoed flag */
  unsigned char    status;  /* row status */
{
  int   ipar,    /* parameter counter */
           i;    /* general counter   */

  if(head_flag)                /* bar name already echoed */
    MOV_CURSOR(row+OFFSET+7);
  else {                       /* bar name not echoed yet */
    MOV_CURSOR(row+OFFSET+4);
    printf("%3s", vtab[STATUS].pchr[bpar]+4);
  }
  if(bpar != DV) {
    ipar = (vtab[STATUS].par[bpar].ipar+5)/100;
    if(ipar<0)
      ipar = 0;
    /* modified to 10% resolution */
    for(i=0; i<=9; i++) {
      if(ipar) {
        putchar(BLACK);         /* sending black cursor */
        --ipar;
      }
      else
        putchar('_');         /* sending underscore   */
    }
    putchar(0x20);
  }
  else {
    ipar = vtab[STATUS].par[DV].ipar/200+6;
    /* modified to 20% resolution */
    for(i=1; i<=11; i++) {
```

```
      if(i==MID_REL)
        putchar('¦');
      else
        if(i<MID_REL)
          if(i>=ipar)
            putchar(BLACK);
          else
            putchar('_');
        else
          if(i<=ipar)
            putchar(BLACK);
          else
            putchar('_');
    }
  }
  MOV_CURSOR(row+OFFSET+3);
  if(status)
    putchar('>');
  else
    putchar(' ');
  return ECHOED;
}


move_bar(unsigned char direction) {
  data unsigned char i, j;

  i = j = 0;
  for(i=0; i<NBR_ROW; i++)
    for(j=0; j<NBR_COL; j++)
      if(i) {
        MOV_CURSOR(j+0x40);
        if(direction)
          bar_buffer[i*NBR_COL+j] = rd_dat();
        else
          wr_dat(bar_buffer[i*NBR_COL+j]);
      }
      else {
        MOV_CURSOR(j);
        if(direction)
          bar_buffer[i*NBR_COL+j] = rd_dat();
        else
          wr_dat(bar_buffer[i*NBR_COL+j]);
      }
}
```

```
/*
  DISP_DAT.C51 -- Display Data
*/

#pragma  CODE SYMBOLS DEBUG  /* compiler options */

/* --> header files <-- */
#include <STDIO.H>    /* - standard output declaration header file */
#include "83C51FA.H"  /* - CPU type declaration */
#include "GLOBAL.H"   /* - general declaration */
#include "PID.H"      /* - PID declaration */
#include "LCD.H"      /* - user display */
#include "DISP_DAT.H" /* - local declaration */


display_data() {
  int quotient, remain1;

  switch (ddisplay) {
    case DMODE_CHANGE:           /* mode message job */
      MOV_CURSOR(STR_DMODE);     /* echo mode */
      printf("%6s", vtab[dmode].mchr);
    case DPAR_CHANGE:            /* group message job */
      MOV_CURSOR(STR_DPAR);
      printf("%6s", vtab[dmode].pchr[dpar]);
    case DDAT_CHANGE:            /* data message job */
      MOV_CURSOR(STR_DPAR+7);
      switch (dmode) {           /* which mode to display */
        case TUNING:             /* tuning mode */
          printf(vtab[dmode].fchr, tmp[dpar]);
          break;
        case  ALARM:             /* PID alarm */
        case STATUS:             /* its value */
          quotient = tmp[dpar]/10;
          remain1  = tmp[dpar]%10;
          if((tmp[dpar]>=-9)&&(tmp[dpar]<0))
            printf("  -%1d.%1d", quotient,remain1);
          else
            printf(vtab[dmode].fchr, quotient, remain1);
          break;
        case ENG:            /* ENGineering MODe */
          switch (dpar) {
            case T_MIN :
            case T_MAX :
              printf(vtab[dmode].fchr, tmp[dpar]);
              break;
            case OP_0 : /* output offset */
              printf("<  0%%>");
              break;
            case OP_100 : /* output span */
              printf("<100%%>");
              break;
```

```
            }
          break;
        }
        break;
    }
    ddisplay = RESET; /* reset display flag when already done */
}
```

```
/*
  GETKEY.C51 -- get user key
  note : This file is used by printf function. So user should link
         this program to used library at first. At this time use
         C51L.LIB standard library.
*/

#pragma CODE SYMBOLS DEBUG /* compiler options */

/* --> header files <-- */
#include <reg51.h>          /* CPU register define */
#include <absacc.h>         /* absolute address access */

#define USER_KEY  XBYTE[0x8000]

extern bit user_keyboard;
extern get_user_key();

char _getkey () {
  return USER_KEY;
}
```

```
/*
  KEYBOARD.C51 -- Keyboard service routine
  External -- var : timebase
              func: -
  Descript --
*/

#pragma  CODE DEBUG SYMBOLS  /* compiler options */

/* --> header files <-- */
#include <STDIO.H>    /* standard i/o defined */
#include "83C51FA.H"  /* CPU defined */
#include "GLOBAL.H"   /* global defined */
#include "PID.H"      /* PID system defined */
#include "LCD.H"      /* display defined */
#include "KEYBOARD.H" /* keyboard defined */

/* control panel counter declaration */
                              /* data display */
data unsigned char         dmode,  /* mode */
                            dpar,  /* parameter */
                                   /* bar display  */
                           bmode,  /* mode */
                           bpar1,  /* parameter for row 1 */
                           bpar2;  /* parameter for row 2 */
                                   /* engineering data display */
data unsigned char        keybuf;  /* keyboard buffer */
unsigned char           ddisplay;  /* display change */
unsigned char        mode_change;  /* mode change determine by calcoef
*/
                                   /* mode move from tmp to vtab flag */
unsigned char cast_flag[MAXDMODE] = {CAST,CAST,CAST,CAST};

/* function declaration */
keyboard_routine();           /* keyboard_routine */
void service_one_key();       /* key service */
void save_mode_tmp(void);     /* save tmp[] to vtab[] */
void restore_mode_tmp(void);  /* restore vtab[] to tmp[] */
unsigned char alarm_update(); /* update float data alarm */
void store_int2float();       /* store integer to floating point storage
*/

keyboard_routine() {
  void service_one_key();

  if(!update) {
    if(_getkey() != NOPRESS) {   /* if key is pressed */
      keybuf = _getkey();
      /* turn on timer if not yet  */
      if(!evtimer_on) {
        /* set next match to compare with timebase */
        CCAP1L = CL+LOW_T_SFAST;
```

```
        CCAP1H = CH+ (unsigned char) CY+ HIG_T_SFAST;
        CCF1  = RESET;            /* clear event timer interrupt flag */
        CCAPM1 |= 0x9;            /* enable event timer interrupt */
        evtimer_on=ON;            /* timer flag on */
      }
      if(fast_flag && (keybuf!=ISPARA)) { /* fast flag on & neglect PAR
key */
        service_one_key(keybuf);          /* service alternate function
of key */
        update = OK;
      }
    }
    else { /* determine release or nopress key */
      /* getkey still in keybuf */
      if(keybuf!=NOPRESS) {             /* release key */
        service_one_key(keybuf);
        keybuf = NOPRESS;
        update = OK;
      }
      fast_flag   = done1sec = OFF;
      evtimer_on  = OFF;              /* reset event timer  */
      CCAPM1     &=0xf6;              /* reset PCA module 0 */
    }
  }
}

void service_one_key(unsigned char key) {
  unsigned char i;          /* general counter */
  void save_mode_tmp();     /* save tmp[] when mode pressed */
  void restore_mode_tmp(); /* restore tmp[] when mode pressed */

  switch(key) {

    case ISAUTOMAN:
      if(!fast_flag) {      /* function "OS" */
        if((dmode!=EMODE)&&(!err_flag)) /* If not engineering mode */
          MAN_FLAG = ~MAN_FLAG; /* swapp auto/manual flag */
        else
          MAN_FLAG = RDWR; /* Mode ENG, lock to manual */
      }
      else {                /* key function "Disp" */
        keybuf = DUMMY;
        if(!done1sec) { /* protect for doing one time only */
          if(!err_flag) {
            bar_flag = ~bar_flag;
            if(!bar_flag)
              tmp[dpar] = vtab[dmode].par[dpar].ipar;
          }
          else
            bar_flag = RESET;
          swap_flag = SET; /* changing display */
          done1sec = ON;
```

```
    }
  }
  break;
case ISMODE:
  if(!fast_flag) {              /* function "M" */
    if(!bar_flag) {  /* data display */
      mode_change = dmode; /* calculate old mode */
      cast_flag[dmode] = CASTING; /* need cast to type float */
      save_mode_tmp(); /* save last mode */
      if(!err_flag) {
        dmode = (++dmode<NDMODE)? dmode:0;
        dpar = 0;
      }
      else {
        dmode = STATUS;
        dpar = MV;
      }
      restore_mode_tmp(); /* restore current mode */
      ddisplay = DMODE_CHANGE;
    }
    else              /* bar display */
      bmode = (++bmode<MAXBMODE)? bmode:0;
  }
  else {                        /* function "Eng" */
    keybuf   = DUMMY;
    if(!done1sec) {
      done1sec = ON;
      if(!bar_flag) {   /* data mode */
        mode_change = dmode;
        save_mode_tmp();
        if(!err_flag) {
          dmode = EMODE;
          dpar = 0;
        }
        else {
          dmode = STATUS;
          dpar  = MV;
        }
        restore_mode_tmp(); /* restore current mode */
        MAN_FLAG = RDWR;
        ddisplay = DMODE_CHANGE;
      }
    }
  }
  break;
case ISPARA:                      /* function "P"*/
  if(bar_flag) {                      /* bar display */
    if(!bmode)
      bpar1 = (++bpar1 < MAXBPAR)? bpar1:0;
        else
      bpar2 = (++bpar2 < MAXBPAR)? bpar2:0;
    bhead[bmode]      = ECHO;
```

```
    }
    else {                                  /* data mode */
      dpar = (++dpar<vtab[dmode].nparm)? dpar:0;
      ddisplay = DPAR_CHANGE;               /* status on */
    }
    break;
  case ISUP :                               /* function "Up" or "Down" */
  case ISDOWN :
    /* check range first for specified mode */
    switch (dmode) {
      case EMODE :
        switch (dpar) {
          case T_MAX :
            MIN_TMAX_I = tmp[T_MIN];
            break;
          case T_MIN :
            MAX_TMIN_I = tmp[T_MAX];
            break;
          case OP_0:
          case OP_100:
            vtab[EMODE].par[dpar].ipar = tmp[dpar];
            vtab[EMODE].par[dpar].fpar = (float) tmp[dpar]*.01;
            break;
        }
        break;
      case ALARM :
        switch (dpar) {
          case AL :
            MAX_AL_I = tmp[AH];
            MIN_AL_I = tmp[LL];
            break;
          case AH :
            MAX_AH_I = tmp[HH];
            MIN_AH_I = tmp[AL];
            break;
          case HH :
            MIN_HH_I = tmp[AH];
            break;
          case LL :
            MAX_LL_I = tmp[AL];
            break;
        }
        break;
    }
    /* Use these cases if data display AND its flag = RDWR */
    if(vtab[dmode].flag[dpar] & (!bar_flag)) {
      if(key==ISUP)
        tmp[dpar] = (++tmp[dpar]>vtab[dmode].max[dpar])?
                    vtab[dmode].max[dpar]:tmp[dpar];
      else
        tmp[dpar] = (--tmp[dpar]<vtab[dmode].min[dpar])?
                    vtab[dmode].min[dpar]:tmp[dpar];
```

```
        ddisplay = DDAT_CHANGE;
      }
      break;
  }
}

/* move tmp[] to table variables */
void save_mode_tmp() {
  unsigned char i;

  for(i=0; i<vtab[dmode].nparm; i++) /* save old mode */
    if(vtab[dmode].flag[i] == RDWR)
      vtab[dmode].par[i].ipar = tmp[i];
}

/* load tmp[] with table variables */
void restore_mode_tmp(void) {
  unsigned char i;

  for(i=0; i<vtab[dmode].nparm; i++) /* load new mode */
    tmp[i] = vtab[dmode].par[i].ipar;
}

void store_int2float(mode, num_para, multiplier)
  unsigned char mode;
  unsigned char num_para;
  float multiplier;
{
  unsigned char i;

  for(i=0; i<num_para; i++)
    vtab[mode].par[i].fpar = ((float)
vtab[mode].par[i].ipar)*multiplier;
}

void err_set() { /* error set global variables routine */

  err_flag = SET;
  MAN_FLAG = RDWR;
  dmode    = STATUS;
  bar_flag = RESET;
  /* --> check output set DIP switch */
  MV_I = (OP_SET)? 0:1000;
  tmp[3] = MV_I;
}
```

```
/*
  LCD.C51 -- LCD display

  Include : LCD.H   -- LCD declarations header file
  Function: init_lcd(), wait(), wr_inst(), rd_inst(), wr_dat(), rd_dat()
  note    : before the operation to read or write the data port,user
should
            set the instruction port (by using "wr_inst" function)
whether
            DD RAM address or CG RAM address.

*/

#pragma CODE SYMBOLS DEBUG /* compiler options */

/* --> header files <-- */
#include <ABSACC.H>     /* absolute address access */
#include "LCD.H"        /* user display defined header */



void wr_inst();
void wr_dat();
unsigned char rd_inst();
unsigned char rd_dat();

/* ------------ initialized LCD routine ------------- */
init_lcd() {
  data unsigned char i,j;

  for(j=0; j<=3; ++j) {
    LCD_INST_WR = 56;           /* 8bit 2row 5x10dot */
    for(i=1; i<=100; i++);
  }
  DISPLAY_CONTROL(12); /* display on cursor off blink off */
  CLEAR_DISPLAY;       /* clear display  */
  SET_ENTRY(6);        /* increment shift */
  SET_CGRAM(0);
  for(i=0; i<7; ++i)
    WRITE_CG(255);
  RETURN_HOME;         /* return home */
}

/* ------------- busy waiting routine  ------------- */
void wait() {
  while(LCD_INST_RD&0x80);
}
/* ------------- write instruction port ------------ */
void wr_inst(unsigned char inst_data) {
  wait();
  LCD_INST_WR = inst_data;
}
```

```
/* ------------- read  instruction port ------------ */
/*
unsigned char rd_inst() {
  wait();
  return LCD_INST_RD;
}
*/
/* ---------------- write data port --------------- */
void wr_dat(char c) {
  wait();
  LCD_DAT_WR = c;
}

/* ---------------- read data port ----------------- */
unsigned char rd_dat() {
  wait();
  return LCD_DAT_RD;
}
```

```
/*
   LINEAR.C51 -- software linearlized function
*/

/*
  DIP switch define type of sensor location

    Note : for ThermoCouple the value are

              TYPE_B   0x0
              TYPE_E   0x1
              TYPE_J   0x2
              TYPE_K   0x3
              TYPE_R   0x4
              TYPE_S   0x5
              TYPE_T   0x6
    Note : for Pt 100 RTD the value are (spare)
              DISABLE  0x8 ; disable sensor and use Standard current

    This means defined sequence of look-up table
    should be followed this rule.
*/
#pragma CODE SYMBOLS DEBUG
#include "83c51fa.h"
#include "pid.h"

/* look-up table define value */
  #define VALUE_X(n)    sensor_table[n].value_x
  #define VALUE_Y(n)    sensor_table[n].value_y
  #define SLOPE(n)      sensor_table[n].slope
  #define OFFSET(n)     sensor_table[n].offset

/* variable declaration */
  /* local type */
  /* sensor buffer */
    unsigned char       sens_buf;
  /* look-up table data structure */
  typedef struct {
    float value_x;        /* independent variable such as characteristic
voltage */
    float value_y;        /* dependent variable such as temperature */
    float   slope;        /* linear equation slope  */
    float  offset;        /* linear equation offset */
  }table;                 /* structure of sensor lookup table */
#include "linear.h"       /* sensor look-up table */

     /* Offset of the first data of each sensor type, the first is 0. */
                             /* Type  B,  E,  J,  K,  R,  S,  T */
  code unsigned char lin_offset[] = { 0,  9, 14, 18, 23, 29, 35 };
                            /* Number of data of each sensor type. */
                           /* Type   B,  E,  J,  K,  R,  S,  T */
  code unsigned char    lin_set[] = { 9,  5,  4,  5,  6,  6,  5 };
```

```
  float gain;              /* opamp gain  */
  float x_delta;           /* max-min of independent variable */
  float x_max;             /* max of x that give output of OpAmp 5 V */
  float z;                 /* standard voltage offset */
  float y_range_100;

/* function declaration */
  float find_x_value(float y_value);
  float convert(float delta_ratio, float min, float y);
  float linear_and_convert(float x);
  unsigned char cal_sens_parameter();

/*
  function : convert -- convert the sensor value to % of sensor value
  argument :
    y      : the value which will be converted
    delta_ratio : 100/(max-min)
    min    : minimum value
  call     : none
  called   : linear_and_convert(), forground_routine() (module SYSTIME)
  return : % of full scale of argument
*/

float convert(float delta_ratio, float min, float y) {
  return  delta_ratio*(y-min);  /* convert to % of full scale */
}

/* ---------------------------------------------------------------------
  function : linear_and_convert - linearlize, deamplification and find
             the corresponding sensor value from standard input voltage.
  argument : x ;digital value from input
  variable :
    global : gain
             VALUE_X()
             OFFSET()
    local  : n
  return   : % of sensor value
  call     : convert() (module LINEAR)
  called   : forground_routine() (module SYSTIME)
  purpose  : Find the corresponding sensor value
*/

float linear_and_convert(float x) {
  unsigned char n;
  unsigned char offset;

  offset = lin_offset[SENSOR_TYPE];
  x  = (x-z)/gain;     /* deamplification the input */
  for(n=0; n<lin_set[SENSOR_TYPE]; n++)  /* look-up table */
    if(x > VALUE_X(offset+n))
      continue;
    else
```

```
      break;
  return
convert(y_range_100,T_MIN_F,SLOPE(offset+n)*x+OFFSET(offset+n));
}

/*
  function : cal_sens_parameter - calculate sensor parameter
  argument : none
  variable :
    global : T_MIN_F - minimum value of Y (such as Temperature) defind
by user
             T_MAX_F - maximum value of y
             VALUE_Y() - maximum of Y of sensor table which is able to
use its linealized
                         equation.
             OFFSET() - offset of sensor table equation
             SLOPE() - slope of sensor table equation
             x_max - maximum of independent variable which is
corresponded to T_MAX_F
             x_delta - delta variable which is x_max-x_min
             gain - opamp gain
             z - standard voltage offset
             y_range_100 - coefficient for calculate dependent variable
    local  : n - counter
             offset - for sensor linearlized data array offset
  call     : none
  called   : main (module BACKGND)
  description : sensor parameters are x_min, op_gain_i, y_range_100.
These parameter
    are calculated only once when turn on the controller for calculate
the range of the
    sensor.
*/

unsigned char cal_sens_parameter() {
  unsigned char n;
  unsigned char offset;

  for(n=2; n<4; n++)
    vtab[ENG].par[n].fpar = (float) vtab[ENG].par[n].ipar*.01;
  if(SENSOR_TYPE <= 6) {
    for(n=0; n<2; n++)
      vtab[ENG].par[n].fpar = (float) vtab[ENG].par[n].ipar;
    x_max    = x_delta = find_x_value(T_MAX_F); /* find max value first
*/
    x_delta -= find_x_value(T_MIN_F);        /* next find min value and
make delta value */
    gain     = 3276.8/x_delta;               /* OpAmp gain */
    z        = 4096-gain*x_max;
    y_range_100 = 100/(T_MAX_F-T_MIN_F);     /* mitiplier of Y */
  }
  return -1;
```

```
}

float find_x_value(float y_value) {
  unsigned char n;  /* general counter */
  unsigned char x_offset;  /* offset of sensor table array */

  x_offset = lin_offset[SENSOR_TYPE];
  n = 0;
  while(n<lin_set[SENSOR_TYPE]) {
    if(y_value > VALUE_Y(x_offset+n)) {
      ++n;
      continue;
    }
    else
      break;
  }
  return (y_value-OFFSET(x_offset+n))/SLOPE(x_offset+n);
}

init_max_engmode() {
  T_MAX_F = VALUE_Y(lin_offset[SENSOR_TYPE]+lin_set[SENSOR_TYPE]-1);
  MAX_TMAX_I = MAX_TMIN_I = T_MAX_I = (int) T_MAX_F;
  OP_0_F = 25;
  OP_0_I = 2500;
  OP_100_F=16.384;
  OP_100_I=1638;
}
```

```
/*
  PUTCHAR.C51 -- Put a character to standard output

  note : this routine is used standard output routine. So it should be
linked to used
         standard library at first. (now use C51L.lib)
*/

#pragma CODE SYMBOLS DEBUG  /* compiler option */

/* --> header files <-- */
#include <REG51.H>          /* CPU register define */

extern void wr_dat();             /* user defined display function */

char putchar (char c)  {
    wr_dat(c);   /* user defined routine */
    return c;
}
```

```
/*
    SA12.C51 -- 12 bits Successive Approximation

Include  : SA12.H   -- its header file
Descript :
  Specification : using with 12 bits A/D, D/A hardware.
                  input : standard voltage analog signal (1-5V)
                  output: modified digital data 0-4095 for (1-5V)
                  method:
                    first do 12 bits successive by CPU
                    next  convert from 0-5V range to 1-5V range digital
                    data.
*/

#pragma  CODE SYMBOLS DEBUG  /* compiler options */

/* --> header files <-- */
#include "83C51FA.H"         /* CPU register define */
#include "GLOBAL.H"          /* global define */
#include "SA12.H"            /* its header file */

/* --> variables define (not used in these module) */
unsigned ad_max_val; /* value from MAX_CHAN */
unsigned ad_min_val; /* value from MIN_CHAN */

delay() {
  data unsigned char i;

  for(i=0; i<2; i++);
}

/* -------- 12 bits a/d successive approximation function -------- */
unsigned sa12(unsigned char channel) {
  data unsigned         result;
                    /* mask bits, enable mux and clear MSB nibble */
  result = 0;
  P2  |= 0xf0;       /* clear mux to no value first */
  delay();
  P2  = 0;
  P2 |= channel;     /* set channel */
  P0  = 0x0;         /* reset P0 and delay mux */
  P2B7 = ON;         /* turn on mux */
                     /* successive approximation begin here */
  delay();
  P2B3 = 1;
  delay();
  if(COMP)
    P2B3 = 0;
  P2B2 = 1;
  delay();
  if(COMP)
    P2B2 = 0;
```

```
  P2B1 = 1;
  delay();
  if(COMP)
    P2B1 = 0;
  P2B0 = 1;
  delay();
  if(COMP)
    P2B0 = 0;
  P0B7 = 1;
  delay();
  if(COMP)
    P0B7 = 0;
  P0B6 = 1;
  delay();
  if(COMP)
    P0B6 = 0;
  P0B5 = 1;
  delay();
  if(COMP)
    P0B5 = 0;
  P0B4 = 1;
  delay();
  if(COMP)
    P0B4 = 0;
  P0B3 = 1;
  delay();
  if(COMP)
    P0B3 = 0;
  P0B2 = 1;
  delay();
  if(COMP)
    P0B2 = 0;
  P0B1 = 1;
  delay();
  if(COMP)
    P0B1 = 0;
  P0B0 = 1;
  delay();
  if(COMP)
    P0B0 = 0;
  result |= (P2 & 0xf);
  result <<= 8;
  result |= P0;
  P2 |= P0 |= 255;
  return result;
}
```

```
/*
  SYSTIME.C51 -- system timer (sampling timer)
*/

#pragma CODE SYMBOLS DEBUG  /* compiler options */

/* --> header files <-- */
#include <STDIO.H>          /* standard input output */
#include "83C51FA.H"        /* hardware header */
#include "LCD.H"            /* display header */
#include "GLOBAL.H"         /* global defining */
#include "PID.H"            /* PID defining */
#include "SYSTIME.H"        /* its header */

/* - internal variable declaration */
  /* -- hardware control variable */
    unsigned char         nv_flag;   /* NV RAM flag */
  /* -- PID variables */
    float              PROP;  /* PROPortional */
    float     INTE, INTE_1;  /* INTEgral present and last event  */
    float      DER,  DER_1;  /* DERivative present and last event */
    float             sv_1;  /* sv last event */
    float             pv_1;  /* pv last event */
    float             dv_1;  /* dv last event */
    float         pid_k[4];  /* PID gain array (kp,ki,kd1,kd2) */
    float        pid_kd[4];  /* PID gain array mirror */
    float        k_temp[2];  /* PID gain tempory buffer transfer storage
*/
  /* -- general counter */
    data unsigned char j;    /* counter */

  /* -- Output buffer structure */
    data union {
      struct{
        unsigned char high; /* High nibble of 12 bits D/A */
        unsigned char low;  /* 8 bits left */
      } bite;               /* byte pattern format */
      int word;             /* word pattern format */
    } mv_out;               /* name of the variable */

/* - internal function declaration */
  void save_vtab();
  void restore_vtab();
  void init_system_timer();
  void send_output();

forground_routine(void) interrupt 6 using 1 {

  if(CCF0) {                /* CCON.0 PCA module 0 interrupt */
    EC      = RESET;        /* Disable PCA interrupt */
    CCF0    = RESET;        /* Clear Module 0's interrupt flag */
    /* --> Turn on watchdog timer at first time */
```

```
    if((CCAPM4&0x7f)!=0x48)
      CCAPM4 = 0x48;
    /* --> set next match of watchdog timer */
    wd_flag = WD_RESET;        /* reset watchdog flag */
    CCAP4L  = CL+LOW_WMATCH;   /* load low and turn off comparator */
                               /* load high and turn on comparator */
    CCAP4H  = CH+(unsigned char) CY + HIG_WMATCH;
    wd_flag = WD_SET;          /* set watchdog flag */
    /* --> set_next_match of system timer */
    CCAP0L += LOW_MATCH;
    CCAP0H += (unsigned char) CY + HIG_MATCH;
    /* --> Get PV */
                           /* linearlize and convert */
    if(SENSOR_TYPE <= 6)
      PV_F = linear_and_convert((float) GET_VAR(PV_CHAN));
    else
      PV_F = (convert(PV_RANGE100, PV_MIN, (float)
GET_VAR(PV_CHAN))*1.05);
    if((PV_F<=pv_1+0.4)&&(PV_F>=pv_1-0.4))  /* Rounding error */
      PV_F = pv_1;
    /* --> Get SV */
    if(LOCAL_FLAG)                         /* LOCAL */
      SV_F = (float) SV_I*0.1;
    else                                   /* REMOTE */
      SV_F = (convert(SV_RANGE100,SV_MIN, (float)
GET_VAR(SV_CHAN))*1.05);
    if((SV_F<=sv_1+0.4)&&(SV_F>=sv_1-0.4)) /* Rounding error */
      SV_F = sv_1;
    /* --> Calculate DV */
    DV_F = (DIRECT_SW)? SV_F-PV_F:PV_F-SV_F;
    /* --> Check PV on range */
    if((PV_F>=LL_F)&&(PV_F<=HH_F)) {
      NO_SEVERE_ALARM = SET;     /* no severe alarm */
      if((PV_F>=AL_F)&&(PV_F<=AH_F))
        NO_NORMAL_ALARM = SET;
      else
        NO_NORMAL_ALARM = RESET;
      ala_period_cnt = 0;
    }
    else {
      NO_SEVERE_ALARM = RESET;   /* severe alarm */
      if(++ala_period_cnt==5) {
        ala_swap = ~ala_swap;
        ala_period_cnt = 0;
      }
    }
    /* --> Check miss of PV or SV ( input loose ) */
    if((PV_F>=-0.5)&&(PV_F<=100.5)&&(SV_F>=-0.5)&&(SV_F<=100.5)) {
      err_flag = RESET;
      ad_test  = OK;     /* enable A/D test */
    }
    else {
```

```
  err_type = IP_LOOSE;
  err_set();
  ad_test  = NOK;     /* Bypass A/D test */
}
/* --> calculate PROP at first */
if(!busy)
  PROP  = (PID1_SW)? (KP*PV_F):(KP*DV_F);
else
  PROP = (PID1_SW)? (KPD*PV_F):(KPD*DV_F);
/* --> Calculate MV */
if(MAN_FLAG) {                          /* "OS" = "MA:"nual */
  MV_F = ((float) MV_I)*0.1;
  INTE = MV_F - PROP;
}
else {                                  /* "OS" = "AU:"to */
  if(!busy) {
    INTE  = INTE_1 + KI*(DV_F+dv_1);
    DER   = KD1*DER_1 + KD2*(PV_F-pv_1);
  }
  else {
    INTE  = INTE_1 + KID*(DV_F+dv_1);
    DER   = KD1D*DER_1 + KD2D*(PV_F-pv_1);
  }
  MV_F  = PROP+INTE+DER;
}
/* --> Check MV out of range */
if(MV_F < MV_MIN) {                      /* lower limit */
  INTE  = INTE_1 + MV_MIN-MV_F;
  MV_F  = MV_MIN;
}
else {
  if(MV_F > MV_MAX) {                   /* upper limit */
    INTE  = INTE_1 + MV_MAX-MV_F;
    MV_F   = MV_MAX;
  }
}
/* --> Send output to plant */
send_output();
/* --> Save STATUS variables */
for(j=0; j<4; j++) {
  if(vtab[STATUS].flag[j]==READ) {
    vtab[STATUS].par[j].ipar = (int) (vtab[STATUS].par[j].fpar*10);
    tmp[j] = (!dmode)? vtab[STATUS].par[j].ipar:tmp[j];
    /* Not echo if display is not in status mode or bar display */
  }
}
/* --> Save for next calculation */
sv_1   = SV_F;
                  /* integral term */
INTE_1 = INTE;      /* save to INTn-1 */
dv_1   = DV_F;      /* save to DVn-1 */
                  /* difference term */
```

```
    DER_1  = DER;       /* save to DERn-1 */
    pv_1   = PV_F;      /* save to PVn-1 */
    /* --> display the result if "M" = "STATUS" */
    if(!dmode) {
      ddisplay = DDAT_CHANGE;
      update = OK;
    }
    else
      update = NOK;
    if(!mode_change) {
      cast_flag[STATUS] = CAST;
      mode_change = -1;
    }
    /* --> save all variables to their buffer */
    save_vtab();
    /* --> Enable PCA interrupt (all timer interrupt) */
    EC = SET;
  }
  else {                          /* event timer routine */
    if(CCF1) {       /* event timer interrupt */
      CCF1 = RESET;
      CCAPM1 &= 0xf6;   /* disable event timer interrupt */
      fast_flag = ON;   /* turn of fast flag */
    }
  }
}

void send_output() { /* copy the current MV to plant */
    /* --> modified MV on range */
    mv_out.word = (int) (OP_100_F*(MV_F+OP_0_F));
    /* --> sent MV */
    P2 = mv_out.bite.high;
    P0 = mv_out.bite.low;
    /* --> Activate holding circuit - on->off */
    for(j=0; j<2; j++);  /* D/A delay */
    OUTEN = 0;
    for(j=0; j<10; j++); /* Holding circuit delay */
    OUTEN   = 1;
}

void init_system_timer() {
          /* Set time base to timer0 overflow */
  CH = 0;
  CL = 1;           /* Reset PCA time base counter */
      /*   Module 0 in Software Timer mode .1 sec                    */
      /*   turn on comparator and matching condition                 */
      /*                  -   ECOM0 CAPP0 CAPN0 MAT0 TOG0 PWM0 ECCF0 */
  CCAPM0 = 0x49;    /*  0     1     0     0     1    0    0    1   */
      /*   initialized system timer Compare CAPture register pair    */
  CCAP0L = 0x90;    /* write low byte with matching data  low(400) */
  CCAP0H = 0x01;    /* high(400)                                   */
      /*   Module 4 in Watchdog timer mode .1 sec                    */
```

```
    /*   turn on comparator and matching condition                    */
    /*                  -   ECOM4 CAPP4 CAPN4 MAT4 TOG4 PWM4 ECCF4 */
  CCAPM4 = 0;       /*  0     0     0     0     0    0    0    0   */
    /*   initialized watchdog Compare CAPture register pair          */
  CCAP4L = 0xff;    /*  reset watchdog timer to maximum             */
  CCAP4H = 0xff;
    /*   set PCA Counter Mode register                               */
    /*               CIDL  WDTE    -     -     -  CPS1 CPS0  ECF   */
  CMOD   = 0x44;    /*  0     1     0     0     0    1    0    0   */
    /*               Turn on PCA interrupt                          */
  EC     = ON;      /*  IE.6 (IE=0xA8, IE.6=0xAE)                   */
    /*               Turn on global interrupt                       */
  EA     = ON;      /*  IE.7 (IE=0xA8, IE.7=0xAF)                   */
    /*               Turn on PCA timer                              */
  CR     = ON;      /*  CCON.6 (CCON=0xD8, CCON.6=0xDE)             */
}

void save_vtab() {
  unsigned char i,j;

  for(i=0; i<MAXDMODE; i++) {
    for(j=0; j<MAXDPAR; j++) {
      if(i==STATUS)
        vtabs[STATUS].flag[j] = vtab[STATUS].flag[j];
      vtabs[i].min[j] = vtab[i].min[j];
      vtabs[i].max[j] = vtab[i].max[j];
      vtabs[i].par[j].fpar = vtab[i].par[j].fpar;
      vtabs[i].par[j].ipar = vtab[i].par[j].ipar;
    }
  }
}

void restore_vtab() {
  unsigned char i,j;

  for(i=0; i<MAXDMODE; i++) {
    for(j=0; j<MAXDPAR; j++) {
      if(i==STATUS)
        vtab[STATUS].flag[j] = vtabs[STATUS].flag[j];
      vtab[i].min[j] = vtabs[i].min[j];
      vtab[i].max[j] = vtabs[i].max[j];
      vtab[i].par[j].fpar = vtabs[i].par[j].fpar;
      vtab[i].par[j].ipar = vtabs[i].par[j].ipar;
    }
  }
}
```

```
/*
  TIMEBASE.C51 -- PCA timebase initialization
*/

#pragma CODE SYMBOLS DEBUG  /* compiler options */

/* --> header files <-- */
#include "83C51FA.H"        /* CPU register define */
#include "GLOBAL.H"         /* global define */

void init_timebase(void) { /* hardware dependent */
     TMOD  = 2;        /* set timer 0 to 8 bit auto-reload */
     TH0   = 6;        /* 250 usec set */
     ET0   = 0;        /* disable timer 0 interrupt */
     TR0   = ON;       /* turn on timer 0 */
}

/*
  function : init_event_timer - Initialized event timer
  description :
    pattern of CCAPM1 - PCA Module Compare/CAPture register module 1
    bit  ¦  7 ¦   6  ¦  5   ¦   4  ¦   3  ¦   2  ¦   1  ¦   0  ¦
    name ¦  - ¦ ECOM1¦CCAPP1¦ CAPN1¦ MAT1 ¦ TOG1 ¦ PWM1 ¦ ECCF1 ¦
    set  ¦  0 ¦   1  ¦  0   ¦   0  ¦   0  ¦   0  ¦   0  ¦   0  ¦
    note :
     bit : 6 - enable comparator function
     bit : 3 - matching condition switch disable at first
     bit : 0 - module 1 interrupt disable at first
*/

void init_event_timer(void) {
  CCAPM1 = 0x40;
}
```

```
/*
   XRAMCHK.C51 -- RAM test program prevents crashes

  Function : xramchk();
  argument pass : by register
                  R4:R5 = start address
                  R6:R7 = end address
                  (bank0 selceted)
  Called : main
  return   : 0 : no error
             1 : pattern 0x55 (even bits set) test error
             2 : pattern 0xaa (odd bits set) test error
             3 : walking ones test error
             4 : memory range test error
  descript : Test external RAM. Argument are passed at the internal RAM
             location by variable "startadd" and "endadd".

*/

#pragma  CODE SYMBOLS DEBUG /* compiler options */

/* --> header files <-- */
#include "83C51FA.H"        /* CPU register define */
#include <ABSACC.H>         /* absolute address access */
#include "PID.H"            /* controller define */
#include "XRAMCHK.H"        /* its header file */

unsigned char xramchk() {
  data unsigned           i;   /* address counter */
  data unsigned char    tmp;   /* temporary storage */
  data unsigned char bumper;   /* start address bumper */
  data unsigned char  shift;   /* variable for walking ones test */
  data unsigned char    err;   /* error report */

  err = 0;                     /* at first no error */
  bumper = XBYTE[STARTADD];    /* bump start address before test */
  XBYTE[STARTADD] = BUMP_DAT;  /* 0x5a is bump data */
  for(i=STARTADD+1; i<=ENDADD; i++) {
    tmp = XBYTE[i];            /* save data */
    XBYTE[i]=EVEN_DAT;
    if(XBYTE[i]==EVEN_DAT) {
      XBYTE[i]=ODD_DAT;
      if(XBYTE[i]==ODD_DAT) {
        shift = 1;
        while(shift) {
          XBYTE[i]=shift;
          if(XBYTE[i]==shift) {
            shift <<=1;
            continue;
          }
          else {
            err = 3;
```

```
            break;
          }
        }
      }
      else
        err = 2;
    }
    else
      err = 1;
    XBYTE[i] = tmp;   /* return its data */
    if(err)
      break;
    else
      continue;
  }
  /* test first byte */
  if(!err)  /* if not error test first byte - else bypass this process
*/
    if(XBYTE[STARTADD]!=BUMP_DAT)
      err = 4;
  XBYTE[STARTADD] = bumper;
  return err;
}
```

```
/*
   83C51FA.H -- CPU registers declaration
   Declaration Register for 83C51FA, 83C51FB, 83C51FC
   device
 */

/*
   Device list
     ROMless
       1. 80C51FA
     EPROM version
       1. 87C51FA -- 8k
       2. 87C51FB -- 16k
       3. 87C51FC -- 32k
*/
/*  BYTE Registers  */
sfr P0    = 0x80; /* port 0 */
sfr SP    = 0x81; /* internal Stack Pointer */
sfr DPL   = 0x82; /* Data Pointer low Byte  */
sfr DPH   = 0x83; /* Data Pointer high Byte */
sfr PCON  = 0x87; /* Power CONtrol register */
sfr TCON  = 0x88; /* Timer/counter CONtrol register */
sfr TMOD  = 0x89; /* Timer/counter MODe control register */
sfr TL0   = 0x8A; /* Timer 0 Low byte */
sfr TL1   = 0x8B; /* Timer 1 Low byte */
sfr TH0   = 0x8C; /* Timer 0 High byte */
sfr TH1   = 0x8D; /* Timer 1 High byte */
sfr P1    = 0x90; /* port 1 */
sfr SCON  = 0x98; /* Serial port CONtrol register */
sfr SBUF  = 0x99; /* Serial BUFfer */
sfr P2    = 0xA0; /* port 2 */
sfr IE    = 0xA8; /* Interrupt Enable register */
sfr SADDR = 0xA9; /* for multiprocessor communication */
sfr P3    = 0xB0; /* port 3 */
sfr IP    = 0xB8; /* Interrupt Priority register */
sfr SADEN = 0xB9; /* for multiprocessor communication */
sfr T2CON = 0xC8; /* Timer/counter 2 CONtrol register */
sfr T2MOD = 0xC9; /* Timer 2 Mode Control Register */
sfr RCAP2L= 0xCA; /* Timer/counter 2 CAPture Register low byte */
sfr RCAP2H= 0xCB; /* Timer/counter 2 CAPture Register high byte */
sfr TL2   = 0xCC; /* Timer/counter 2 Low byte */
sfr TH2   = 0xCD; /* Timer/counter 2 High byte */
sfr PSW   = 0xD0; /* Program Status Word */
sfr CCON  = 0xD8; /* PCA Counter Control Register */
sfr CMOD  = 0xD9; /* PCA Counter Mode Register */
sfr CCAPM0= 0xDA; /* PCA Module 0 Compare/CAPture Register */
sfr CCAPM1= 0xDB; /* PCA Module 1 Compare/CAPture Register */
sfr CCAPM2= 0xDC; /* PCA Module 2 Compare/CAPture Register */
sfr CCAPM3= 0xDD; /* PCA Module 3 Compare/CAPture Register */
sfr CCAPM4= 0xDE; /* PCA Module 4 Compare/CAPture Register */
sfr ACC   = 0xE0; /* ACCumulator */
sfr CL    = 0xE9; /* PCA Count value Low byte */
```

```
sfr CCAP0L= 0xEA; /* PCA Compare/CAPture value module 0 Low byte */
sfr CCAP1L= 0xEB; /* PCA Compare/CAPture value module 1 Low byte */
sfr CCAP2L= 0xEC; /* PCA Compare/CAPture value module 2 Low byte */
sfr CCAP3L= 0xED; /* PCA Compare/CAPture value module 3 Low byte */
sfr CCAP4L= 0xEE; /* PCA Compare/CAPture value module 4 Low byte */
sfr B     = 0xF0; /* register B */
sfr CH    = 0xF9; /* PCA Count value High byte */
sfr CCAP0H= 0xFA; /* PCA Compare/CAPture value module 0 Low byte */
sfr CCAP1H= 0xFB; /* PCA Compare/CAPture value module 1 Low byte */
sfr CCAP2H= 0xFC; /* PCA Compare/CAPture value module 2 Low byte */
sfr CCAP3H= 0xFD; /* PCA Compare/CAPture value module 3 Low byte */
sfr CCAP4H= 0xFE; /* PCA Compare/CAPture value module 4 Low byte */

/*  BIT Registers  */
/*  TCON  */
sbit TF1   = 0x8F;  /* Timer 1 overFlow flag */
sbit TR1   = 0x8E;  /* Timer 1 Run control   */
sbit TF0   = 0x8D;  /* Timer 0 overFlow flag */
sbit TR0   = 0x8C;  /* Timer 0 Run control   */
sbit IE1   = 0x8B;  /* External interrupt 1  */
sbit IT1   = 0x8A;  /* Interrupt 1 type control bit */
sbit IE0   = 0x89;  /* External interrupt 0  */
sbit IT0   = 0x88;  /* Interrupt 0 type control bit */
/* P0 */
sbit P0B7   = 0x87;
sbit P0B6   = 0x86;
sbit P0B5   = 0x85;
sbit P0B4   = 0x84;
sbit P0B3   = 0x83;
sbit P0B2   = 0x82;
sbit P0B1   = 0x81;
sbit P0B0   = 0x80;

/*  SCON  */
sbit SM0   = 0x9F;  /* Serial port Mode bit specifier 0 when PCON.6 is
reset */
sbit FE    = 0x9F;  /* Framming Error when PCON.6 is set */
sbit SM1   = 0x9E;  /* Serial port Mode bit specifier 1 */
sbit SM2   = 0x9D;  /* multiprocessor communication enable */
sbit REN   = 0x9C;  /* Received ENable */
sbit TB8   = 0x9B;  /* 9th bit Transmitted for mode 2 & 3 */
sbit RB8   = 0x9A;  /* SCON.2 */
sbit TI    = 0x99;  /* Transmit interrupt flag */
sbit RI    = 0x98;  /* Receive interrupt flag */
/* P1 */
sbit P1B7 = 0x97;
sbit CEX4 = 0x97; /* External I/O for Compare/Capture Module 4 */
sbit P1B6 = 0x96;
sbit CEX3 = 0x96; /* External I/O for Compare/Capture Module 3 */
sbit P1B5 = 0x95;
sbit CEX2 = 0x95; /* External I/O for Compare/Capture Module 2 */
sbit P1B4 = 0x94;
```

```
sbit CEX1 = 0x94; /* External I/O for Compare/Capture Module 1 */
sbit P1B3 = 0x93;
sbit CEX0 = 0x93; /* External I/O for Compare/Capture Module 0 */
sbit P1B2 = 0x92;
sbit ECI  = 0x92; /* External Count Input to the PCA */
sbit P1B1 = 0x91;
sbit T2EX = 0x91; /* Timer/Counter 2 Capture/Reload Trigger and
Direction Control */
sbit P1B0 = 0x90;
sbit T2   = 0x90; /* External Count Input to Timer/Counter 2 */

/* P2 */
sbit P2B7   = 0xA7;
sbit P2B6   = 0xA6;
sbit P2B5   = 0xA5;
sbit P2B4   = 0xA4;
sbit P2B3   = 0xA3;
sbit P2B2   = 0xA2;
sbit P2B1   = 0xA1;
sbit P2B0   = 0xA0;

/*  IE  */
sbit EA    = 0xAF;  /* Enable All (global enable) */
sbit EC    = 0xAE;  /* PCA Enable */
sbit ET2   = 0xAD;  /* Timer 2 Enable */
sbit ES    = 0xAC;  /* Serial port Enable */
sbit ET1   = 0xAB;  /* Timer 1 Enable */
sbit EX1   = 0xAA;  /* eXternal 1 Enable */
sbit ET0   = 0xA9;  /* Timer 0 Enable */
sbit EX0   = 0xA8;  /* eXternal 0 Enable */

/*  IP  */
                    /* bit 7 for future use */
sbit PPC   = 0xBE;  /* PCA Priority */
sbit PT2   = 0xBD;  /* Timer 2 Priority */
sbit PS    = 0xBC;  /* Serial port Priority */
sbit PT1   = 0xBB;  /* Timer 1 Priority */
sbit PX1   = 0xBA;  /* eXternal 1 Priority */
sbit PT0   = 0xB9;  /* Timer 0 Priority */
sbit PX0   = 0xB8;  /* eXternal 0 Priority */
/*  P3  */
sbit P3B7  = 0xB7;
sbit RD    = 0xB7;  /* ReaD pin */
sbit P3B6  = 0xB6;
sbit WR    = 0xB6;  /* WRite pin */
sbit P3B5  = 0xB5;
sbit T1    = 0xB5;  /* Timer 1 pin */
sbit P3B4  = 0xB4;
sbit T0    = 0xB4;  /* Timer 0 pin */
sbit P3B3  = 0xB3;
sbit INT1  = 0xB3;  /* external INTerrupt 1 pin */
sbit P3B2  = 0xB2;
```

```
sbit INT0  = 0xB2;  /* external INTerrupt 0 pin */
sbit P3B0  = 0xB0;
sbit TXD   = 0xB1;  /* serial Transmit Data pin */
sbit P3B1  = 0xB1;
sbit RXD   = 0xB0;  /* serial Receive Data pin */


/*  T2CON  */
sbit TF2   = 0xCF;  /* Timer 2 overflow Flag */
sbit EXF2  = 0xCE;  /* Timer 2 EXternal Flag  */
sbit RCLK  = 0xCD;  /* Receive CLocK flag */
sbit TCLK  = 0xCC;  /* Transmit CLocK flag */
sbit EXEN2 = 0xCB;  /* Timer 2 external ENable flag */
sbit TR2   = 0xCA;  /* Timer 2 Run control */
sbit C_T2  = 0xC9;  /* Counter/Timer 2 select */
sbit CP_RL2= 0xC8;  /* Capture/Reload flag */

/* CCON : PCA Counter Control Register */
sbit CF    = 0xDF;  /* Counter overflow Flag */
sbit CR    = 0xDE;  /* Run Control bit */
                    /* bit 5 for future use */
sbit CCF4  = 0xDC;  /* Module 4 interrupt Flag */
sbit CCF3  = 0xDB;  /* Module 3 interrupt Flag */
sbit CCF2  = 0xDA;  /* Module 2 interrupt Flag */
sbit CCF1  = 0xD9;  /* Module 1 interrupt Flag */
sbit CCF0  = 0xD8;  /* Module 0 interrupt Flag */
/*  PSW  */
sbit CY    = 0xD7;  /* CarrY flag */
sbit AC    = 0xD6;  /* Auxiliary Carry flag */
sbit F0    = 0xD5;  /* Flag 0 avialable to the user for general purpose
*/
sbit RS1   = 0xD4;  /* Register bank Selector bit 1 */
sbit RS0   = 0xD3;  /* Register bank Selector bit 0 */
                    /* bit 2 for future use */
sbit OV    = 0xD2;  /* OVer flow flag */
sbit P     = 0xD0;  /* Parity flag */
```

```
/*
   global.h -- global variable declaration



 */

/* string defining */
  #define ON              1
  #define OFF             0
  #define OK              1
  #define NOK             0
  #define SET             1
  #define RESET           0
  #define WD_SET          0xa5
  #define WD_RESET        0
  #define NONVOLATILED    0xa5
  #define CHANGE          1
  #define NOCHANGE        0
  #define MANUAL          255       /* byte defining */
  #define AUTO            0
  #define TEST            0xff
  #define TESTED          0
  #define ACTIVE          1         /* for active bar */
  #define INERT           0
  #define ECHO            0         /* for head of bar */
  #define ECHOED          1
  #define CAST            0
  #define CASTING         255       /* convert integer variable to float
*/
                                    /* variable writeable status */
  #define READ              0       /* READ only */
  #define RDWR            255       /* ReaD/WRite */
```

```
/*
   PID.H -- Controller declaration file
*/

/* --> global define <-- */
  /* - enviroment defining */
    #define MAXBMODE    2     /* max. bar display */
    #define MAXBPAR     4     /* max. bar display */
    #define MAXDPAR     4     /* max. data-display parameter */
    #define MAXDMODE    4     /* max. data-display mode */
    #define NDMODE      3     /* number of data-mode mode */
    #define EMODE       3     /* first number of engineering mode */
  /* - mode member defining */
    #define STATUS    0       /* input parameter */
    #define ALARM     1       /* theirs high/low limit define */
    #define TUNING    2       /* PID parameter */
    #define ENG       3       /* output condition */

  /* - changing in data-display elements */
    #define DMODE_CHANGE    1
    #define DPAR_CHANGE     2
    #define DDAT_CHANGE     3
  /* - mode member changing status */
    #define STATUS_CHANGE   0
    #define ALARM_CHANGE    1
    #define TUNING_CHANGE   2
    #define EMODE_CHANGE    3
  /* - PID symbols declaration */
      /* -- status */
        /* --- mode member */
          #define   SV          0
          #define   PV          1
          #define   DV          2
          #define   MV          3
        /* --- varible */
          /* floating point storage */
          #define   SV_F        vtab[STATUS].par[SV].fpar
          #define   PV_F        vtab[STATUS].par[PV].fpar
          #define   DV_F        vtab[STATUS].par[DV].fpar
          #define   MV_F        vtab[STATUS].par[MV].fpar
          /* integer storage */
          #define   SV_I        vtab[STATUS].par[SV].ipar
          #define   PV_I        vtab[STATUS].par[PV].ipar
          #define   DV_I        vtab[STATUS].par[DV].ipar
          #define   MV_I        vtab[STATUS].par[MV].ipar
          /* max, min defining */
          #define   MV_MAX      100
          #define   MV_MIN      0
      /* --- flag */
          #define   LOCAL_FLAG  vtab[STATUS].flag[SV]   /* remote/local
*/
```

```
        #define   MAN_FLAG    vtab[STATUS].flag[MV]   /* auto/manual
*/
    /* -- alarm */
      /* --- mode members */
        #define   AL          0
        #define   AH          1
        #define   LL          2
        #define   HH          3
      /* --- variable */
        /* floating point */
        #define   AL_F        vtab[ALARM].par[AL].fpar
        #define   AH_F        vtab[ALARM].par[AH].fpar
        #define   LL_F        vtab[ALARM].par[LL].fpar
        #define   HH_F        vtab[ALARM].par[HH].fpar
        /* integer */
        #define   AL_I        vtab[ALARM].par[AL].ipar
        #define   AH_I        vtab[ALARM].par[AH].ipar
        #define   LL_I        vtab[ALARM].par[LL].ipar
        #define   HH_I        vtab[ALARM].par[HH].ipar
        /* max, min */
        #define   MAX_AL_I    vtab[ALARM].max[AL]
        #define   MIN_AL_I    vtab[ALARM].min[AL]
        #define   MAX_AH_I    vtab[ALARM].max[AH]
        #define   MIN_AH_I    vtab[ALARM].min[AH]
        #define   MAX_LL_I    vtab[ALARM].max[LL]
        #define   MIN_HH_I    vtab[ALARM].min[HH]
      /* -- output */
        #define  NO_NORMAL_ALARM    P3B4   /* cpu port 3 bit 4 */
        #define  NO_SEVERE_ALARM    P3B3   /* cpu port 3 bit 3 */
    /* -- tuning */
      /* --- mode members */
        #define   PB          0
        #define   TI          1
        #define   TD          2
        #define   BI          3
      /* --- variable */
        /* floating point */
        #define   PB_F        vtab[TUNING].par[PB].fpar
        #define   Ti_F        vtab[TUNING].par[TI].fpar
        #define   Td_F        vtab[TUNING].par[TD].fpar
        #define   BIAS_F      vtab[TUNING].par[BI].fpar
        /* integer */
        #define   PB_I        vtab[TUNING].par[PB].ipar
        #define   Ti_I        vtab[TUNING].par[TI].ipar
        #define   Td_I        vtab[TUNING].par[TD].ipar
        #define   BIAS_I      vtab[TUNING].par[BI].ipar
    /* -- engineering */
      /* --- mode members */
        #define   T_MIN       0
        #define   T_MAX       1
        #define   OP_0        2
        #define   OP_100      3
```

```
      /* --- variable */
         /* floating point */
         #define   T_MIN_F     vtab[ENG].par[T_MIN].fpar
         #define   T_MAX_F     vtab[ENG].par[T_MAX].fpar
         #define   OP_0_F      vtab[ENG].par[OP_0].fpar
         #define   OP_100_F    vtab[ENG].par[OP_100].fpar
         /* integer */
         #define   T_MIN_I     vtab[ENG].par[T_MIN].ipar
         #define   T_MAX_I     vtab[ENG].par[T_MAX].ipar
         #define   OP_0_I      vtab[ENG].par[OP_0].ipar
         #define   OP_100_I    vtab[ENG].par[OP_100].ipar
         /* min,max */
         #define   MAX_TMIN_I  vtab[ENG].max[T_MIN]
         #define   MIN_TMIN_I  vtab[ENG].min[T_MIN]
         #define   MAX_TMAX_I  vtab[ENG].max[T_MAX]
         #define   MIN_TMAX_I  vtab[ENG].min[T_MAX]
         #define   T_MIN_FLAG  vtab[ENG].flag[T_MIN]
         #define   T_MAX_FLAG  vtab[ENG].flag[T_MAX]
  /* - PID terms gain defining */
    /* variables declared in module SYSTIME */
    #define   KP    pid_k[0]     /* proportional gain */
    #define   KI    pid_k[1]     /* integral gain */
    #define   KD1   pid_k[2]     /* derivative gain 1 */
    #define   KD2   pid_k[3]     /* derivative gain 2 */
    /* mirror of PID gain */
    #define   KPD   pid_kd[0]    /* proportional gain */
    #define   KID   pid_kd[1]    /* integral gain */
    #define   KD1D  pid_kd[2]    /* derivative gain 1 */
    #define   KD2D  pid_kd[3]    /* derivative gain 2 */

/* hardware dependent defining */
  /* - input multiplexer (dg508) */
    #define   MAX_CHAN       0x00 /* 5 V input channel */
    #define   MIN_CHAN       0x10 /* 1 V input channel */
    #define   PV_CHAN        0x20 /* PV input */
    #define   SV_CHAN        0x30 /* SV input */
    #define   GET_VAR(chan)  sa12(chan)
    #define   READ_AD(chan)  sa12(chan)
  /* - PID status DIP switch defining */
    /* - Controller set */
    #define   LOCAL_SW       P1B7     /* P1.7 */
    #define   DIRECT_SW      P1B6     /* P1.6 */
    #define   PID1_SW        P1B5     /* P1.5 */
    /* - Output set when error */
    #define   OP_SET         P1B4     /* P1.4 */
    /* - PID select sensor switch defining */
    #define   SENSOR_TYPE    (P1&0x0f)  /* P1.3 - P1.0 */
  /* - error defining */
    #define   COUNT_OUT_DAT  150 /* watchdog fault counter */
    /* -- error type defining */
    #define   NV_RAM         0
    #define   WATCHDOG       1
```

```
    #define   IP_LOOSE       2
    #define   AD_FAIL        3
  /* - keyboard message defining */
    /* modified if hardware are modified */
    #define  ISAUTOMAN      0xef /* bit DA4, AUTO key is pressed */
    #define  ISMODE         0xf7 /* bit DA3, MODE key is pressed */
    #define  ISPARA         0xfb /* bit DA2, PARameter key is pressed */
    #define  ISUP           0xfd /* bit DA1, UP key is pressed */
    #define  ISDOWN         0xfe /* bit DA0, DOWN key is pressed */
    #define  ISMODEPARA     0xf3 /* bit DA2 & DA3 , MODE & PAR are
pressed */
  /* - external RAM check range defining */
    #define  STARTADD   DWORD[0x4]   /* start address passing location
*/
    #define  ENDADD     DWORD[0x6]   /* end address passing location */
    /* note : include <absacc.h> first */
/*
  -- PID variable table --
  This structure is used by :
    Files : PID.C51
            KEYBOARD.C51
*/
typedef struct {
  char           *mchr;   /* mode message for display */
  unsigned char  nparm;   /* number of parameter of mode */
  char  *pchr[MAXDPAR];   /* parameter message for display */
  char           *fchr;   /* data format */
  unsigned char flag[MAXDPAR]; /* it's flag */
  int   min[MAXDPAR];          /* its minimum */
  int   max[MAXDPAR];          /* its maximum */
  struct {
    float fpar;
    int   ipar;
  }     par[MAXDPAR];          /* its value */
} pstruct;                     /* PID structure */

typedef struct {
  unsigned char flag[MAXDPAR];
  int   min[MAXDPAR];          /* minimum */
  int   max[MAXDPAR];          /* maximum */
  struct {
    float fpar;
    int   ipar;
  }     par[MAXDPAR];          /* its value */
} p_shadow;

/* external structure
   format : extern struct_type var_name;   comment_module_name
*/
  extern pstruct vtab[];    /* backgnd */
  extern p_shadow vtabs[];  /* keyboard */
```

```
/*
   BACKGND.H -- Header file of Background task

   Purpose:  Main function
   Call   :
     variable - see external variable
     function - see external function
   Called :
     none
   Description :
     This module is the main module which implementes the background
   routine of the controller. It calls many functions which are in other
   modules and also implements the way to control user-interface,
hardware
   checking (A/D, NV RAM) and PID coefficients calculation.
     see more:
       user-interface concept
       hardware_checking
       PID coefficients calculation
*/

/* - external variable */
  extern char              *err_msg; /* display */
  extern float                 INTE; /* systime */
  extern float               INTE_1; /* systime */
  extern float                  DER; /* systime */
  extern float                DER_1; /* systime */
  extern float                 sv_1; /* systime */
  extern float                 pv_1; /* systime */
  extern float                 dv_1; /* systime */
  extern unsigned char       nv_flag; /* systime */
  extern unsigned char      ddisplay; /* keyboard*/
  extern unsigned char  mode_change; /* keyboard */
  extern p_shadow            vtabs[]; /* keyboard */
  extern unsigned char  cast_flag[]; /* keyboard */
  extern unsigned char      err_type; /* keyboard */
  extern unsigned char      sens_buf; /* linear */
  extern unsigned         ad_max_val; /* sa12 */
  extern unsigned         ad_min_val; /* sa12 */

/* - external function */
  extern  unsigned char cal_coeff(); /* cal_coef */
  extern  display_routine();        /* display */
  extern  init_lcd();               /* lcd */
  extern  wr_inst();                /* lcd */
  extern  cal_sens_parameter();     /* linear */
  extern  init_max_engmode();       /* linear */
  extern  keyboard_routine();       /* keyboard */
  extern  store_int2float();        /* keyboard */
  extern  err_set();                /* keyboard */
  extern  unsigned sa12();          /* sa12 */
  extern  init_system_timer();      /* systime */
```

```
extern  restore_vtab();          /* systime */
extern  send_output();           /* systime */
extern  init_timebase();         /* timebase */
extern  init_event_timer();      /* timebase */
extern  unsigned char xramchk(); /* xramchk */
```

```
/*
  CALCOEF.H -- PID calculate coefficients  header file
*/

/* --> enviroment defining */
    #define   ALPHA   0.2     /* filter coefficient (1/10 to 1/3) */
    #define   TS      0.1     /* sampling time interval (100 msec) */

    #define   Kd_DENOMINATOR  (2*ALPHA*gamma+1)
    #define   Kd1_NUMERATOR   (2*ALPHA*gamma-1)
    #define   Kd2_NUMERATOR   2*gamma*KP

/*
   EXTERNAL VARIABLE
   format :    extern variable_type variable_name; comment_from_module
*/
extern pstruct vtab[];   /* backgnd */
extern float pid_k[4];   /* systime */
extern float pid_kd[4];  /* systime */
```

```
/*
   DISPLAY.H -- Display routine header file

   FUNCTION : display_routine()
   CALL     : display_bar(), display_data()
   CALLED   : main()
   EX.var   : unsigned char bpar1,bpar2
              unsigned char bar1_head_flag,bar2_head_flag
              bit update,bar_flag
   LOCAL    : none
   DESCRIPTION :
     This module display the message by sending to the standard output
     (LCD). By synchronize with module keyboard the "update" is used to
     signal the display to do the routine. If the job is done,the
"update"
     is reset.
     The routine of the function is done by determining the "bar_flag"
for
     display bar or the data. The "bar1_head_flag" and "bar2_head_flag"
is
     used in bar mode. (see more detail in "disp_bar" module).
*/

/* local defining */
  #define ROW1      0     /* LCD 1st row first address */
  #define ROW2      0x40  /*     2nd row first address */
  #define ALA_LOC   0     /* ALArm display location */
  #define MAN_LOC  0x40   /* auto/MANual display location */

/* alarm constant defining */
  #define SEVERE_ALARM    0xf7
  #define NORMAL_ALARM    0xef
  #define SEVERE_NORMAL   0xe7

/* external function
   format : [func_type] func_name(); comment_module_name
 */
  extern display_bar();     /* disp_bar */
  extern display_data();    /* disp_dat */
  extern wr_inst();         /* lcd */
  extern wr_dat();          /* lcd */
  extern move_bar();        /* lcd */

/* external variable
  format : var_type var_name; comment_module_name
 */
  extern bit              update; /* backgnd */
  extern bit            bar_flag; /* backgnd */
  extern bit           fast_flag; /* backgnd */
  extern bit            err_flag; /* backgnd */
  extern bit           swap_flag; /* backgnd */
  extern unsigned char  ddisplay; /* keyboard */
```

```
extern data unsigned char bpar1; /* keyboard */
extern data unsigned char bpar2; /* keyboard */
extern data unsigned char bmode; /* keyboard */
```

```
/*
   DISP_BAR.H  -- DISPLAY BAR HEADER
*/

/* --> LCD message defining */
  /* --> LCD bar cell character address */
    #define BLACK          0    /* character generated in CG RAM of LCD
*/
  /* - LCD 2 rows 20 characters/row */
    #define ROW_OFFSET  0x40    /* absolute first location */
    #define MID_REL        6    /* relative middle position */
    #define OFFSET         0    /* 1st location to display in each row
*/
    #define MAXBARCELL    40
  /*  EXTERNAL VARIABLE
       format :  extern var_type var_name; comment_module_name
   */
    extern pstruct                vtab[]; /* backgnd */
    extern data unsigned char      bmode; /* keyboard */
    extern unsigned char       adisplay; /* keyboard */
    extern unsigned char bar1_head_flag; /* display */
    extern unsigned char bar2_head_flag; /* display */
  /* - external function
       format : extern func_type func_name; comment_module_name
   */
     extern void wr_inst();          /* lcd */
     extern void wr_dat();           /* lcd */
     extern unsigned char rd_dat();  /* lcd */
```

```
/*
  DISP_DAT.H -- Display data header file

 FUNCTION : display_data(), report_error()
 Call     : printf() - for sending formatted message to the standard o/p
            put_char() - for sending ascii character to the standard o/p
            wr_inst() - for controlling the display (LCD)
            wr_dat() - for echo the character to the display
 Called   : display()
 EX.var   :
   - pstruct       vtab[]
   - int           tmp[]
   - unsigned char dmode,dpar,ddisplay
 Local var:
   int display_data.quotient,remain1
   sfr report_error.PCON,EA
 Description :
   The module is for display the variables value. The format is as
follow
        mode :
        group: value
*/

/* local defining */
   #define  STR_DMODE      4      /* - string of mode position */
   #define  STR_DPAR    0x44      /* - string of parameter position */
   #define  STR_MAN     0x40      /* - string of auto indicator position
*/

/* external variable declaration
   format : extern var_type var_name; comment_module_name
 */
  extern pstruct vtab[];           /* backgnd */
  extern int tmp[MAXDPAR];         /* backgnd */
  extern bit fast_flag;            /* backgnd */
  extern data unsigned char dmode; /* keyboard */
  extern data unsigned char  dpar; /* keyboard */
  extern unsigned char   ddisplay; /* keyboard */

/* external function declaration
   format : extern func_type func_name; comment_module_name
 */
  extern wr_inst();
  extern wr_dat();
```

```
/*
  KEYBOARD.H -- keyboard header file
*/

/* --> local define message <-- */
  /* - timer message */
    /* -- Start of FAST key time (1 sec = 4000 counts) */
      #define HIG_T_SFAST     0x0f  /* high byte of 4000 */
      #define LOW_T_SFAST     0xa0  /* low byte of 4000 */
  /* - keyboard status message */
      #define NOPRESS         255   /* NO key PRESS status */
      #define RELEASE         255   /* the same meaning as NOPRESS */
      #define DUMMY           0     /* keyboard dummy do nothing */

/* external variable
   format : extern var_type var_name; comment_module_name
 */
  extern int             tmp[MAXDPAR]; /* backgnd */
  extern pstruct               vtab[]; /* backgnd */
  extern bit                fast_flag; /* backgnd */
  extern bit               evtimer_on; /* backgnd */
  extern bit                   update; /* backgnd */
  extern bit                 bar_flag; /* backgnd */
  extern bit                 err_flag; /* backgnd */
  extern bit                swap_flag; /* backgnd */
  extern bit                input_cal; /* backgnd */
  extern bit                 done1sec; /* backgnd */
  extern unsigned char        bhead[]; /* display */



/* external function
   format : extern [func_type] func_name; comment_module_name
 */
   extern move_bar();               /* dips_bar */
   extern void wr_inst();           /* lcd */
   extern cal_sens_parm();          /* linear */

/* internal variable declaration */
p_shadow vtabs[4];
```

```
/*
   LCD.H -- LCD display header file
*/

/* --> LCD global defining <-- */
  /* - port defining */
    #define   LCD_INST_WR   XBYTE[0x8004]  /* instruction port write */
    #define   LCD_INST_RD   XBYTE[0x8006]  /* instruction port read */
    #define   LCD_DAT_WR    XBYTE[0x8005]  /* data port write */
    #define   LCD_DAT_RD    XBYTE[0x8007]  /* data port read */
  /* - message defining */
    #define HOME            0              /* first location of dd ram
*/
  /* - LCD type defining */
    #define NBR_ROW         2              /* for 2 rows 20 characters
*/
    #define NBR_COL         20
  /* - function defining */
    /* -- general defined */
      #define CLEAR_DISPLAY          wr_inst(1)   /* clear display */
      #define DISPLAY_CONTROL(word) wr_inst(word)    /* display control
*/

      #define MOV_CURSOR(location)  wr_inst(location|0x80)
                                                  /* same meaning*/
      #define RETURN_HOME           wr_inst(2)   /* cursor return home
*/
      #define SHIFT_CURSOR_RIGHT    wr_inst(20)  /* shift cursor right
*/
      #define SHIFT_CURSOR_LEFT     wr_inst(16)  /* shift cursor left */
      #define SET_CGRAM(address)    wr_inst(address|0x40) /* set cg ram
address */
      #define SET_DDRAM(address)    wr_inst(address|0x80)
      #define SET_ENTRY(mode)       wr_inst(mode)    /* entry mode set
*/
      #define SET_FUNCTION(mode)    wr_inst(mode)    /* function mode
set */
                                                  /* set dd ram address
*/
      #define SHIFT_DISPLAY_LEFT    wr_inst(24)  /* shift address
counter 1
                                                    left position */
      #define SHIFT_DISPLAY_RIGHT   wr_inst(28)  /* shift address
counter 1
                                                    right position */
      #define WRITE_CG(data)        wr_dat(data)
      #define READ_CG(data)         rd_dat(data)
      #define WRITE_DD(data)        wr_dat(data)
      #define READ_DD(data)         rd_dat(data)
    /* -- for 2 rows 20 columns LCD only */
      #define SAVE_BAR              move_bar(1)
      #define RESTORE_BAR           move_bar(0)
```

```
/*
   LINEAR.H -- Sensor data table
*/

/*
      voltage          temp          slope         offset
        (V)          (deg.C)
*/
code table sensor_table[] = {
    /*   Type B linearlizing data */
    2.2000E-04,         220,     6.783919598E+05,      71,
    6.1400E-04,         355,     3.426395939E+05,     145,
    1.1920E-03,         490,     2.335640138E+05,     212,
    1.9120E-03,         620,     1.805555556E+05,     275,
    2.7820E-03,         750,     1.494252874E+05,     334,
    4.1260E-03,         920,     1.264880952E+05,     398,
    5.6800E-03,        1090,     1.093951094E+05,     469,
    7.7360E-03,        1290,     9.727626459E+04,     537,
    1.3585E-02,        1800,     8.719439220E+04,     615,
    /*   Type E linearlizing data */
    7.3390E-03,         115,     1.566970977E+04,       0,
    1.2314E-02,         185,     1.407035176E+04,      12,
    2.2989E-02,         325,     1.311475410E+04,      24,
    5.2711E-02,         695,     1.244869121E+04,      39,
    7.6358E-02,        1000,     1.289804203E+04,      15,
    /* Type J linearlizing data */
    1.1887E-02,         220,     1.850761336E+04,       0,
    2.9358E-02,         535,     1.802987808E+04,       6,
    3.5764E-02,         645,     1.717140181E+04,      31,
    4.2922E-02,         760,     1.606594021E+04,      70,
    /*    Type K linearlizing data   */
    1.7453E-02,         425,     2.435111442E+04,       0,
    3.1007E-02,         745,     2.360926664E+04,      13,
    3.8122E-02,         920,     2.459592410E+04,     -18,
    4.7171E-02,        1155,     2.596972041E+04,     -70,
    5.4807E-02,        1370,     2.815610267E+04,    -173,
    /*Type R  linearlizing data */
    6.8500E-04,         115,     1.678832117E+05,       0,
    2.0640E-03,         265,     1.087744743E+05,      40,
    4.5800E-03,         510,     9.737678855E+04,      64,
    7.4600E-03,         760,     8.680555556E+04,     112,
    1.0503E-02,        1000,     7.886953664E+04,     172,
    2.1006E-02,        1760,     7.236027802E+04,     240,
    /*  Type S linearlizing data */
    4.6700E-04,          75,     1.605995717E+05,       0,
    1.4400E-03,         200,     1.284686536E+05,      15,
    3.7430E-03,         450,     1.085540599E+05,      44,
    6.2740E-03,         700,     9.877518767E+04,      80,
    9.4700E-03,         990,     9.073842303E+04,     131,
    1.8612E-02,        1760,     8.422664625E+04,     192,
    /*   Type T linearlizing data */
    2.6430E-03,          64,     2.421490730E+04,       0,
```

```
5.7120E-03,         130,     2.150537634E+04,       7,
9.5530E-03,         205,     1.952616506E+04,      18,
1.4261E-02,         290,     1.805437553E+04,      33,
2.0869E-02,         400,     1.664648910E+04,      53,
/* Pt100 RTD (spare) */
/* Standard current */
};
```

```
/*
  SA12.H -- 12 bit Successive Approximation header file
    module : SA12
    function :  unsigned sa12(), void delay()
  Description : This file is user defined for his hardware. The hardware
is
    for 8 channel analog Multiplexer which device name is DG508.
  Note : channel decoded by msb nibble bit 6,5,4 .
*/
#define  COMP      P3B2              /* comparator result bit */
```

```
/*
  SYSTIME.H -- system timer (sampling timer) header file
*/

/* Defining information */
/* - output enable bit */
  #define  OUTEN              P3B5
/* - matching data defining for system timer (module 0 of PCA) */
  #define  LOW_MATCH          0x90    /* low(400) */
  #define  HIG_MATCH          0x01    /* high(400) */
/* - matching data defining for watchdog timer (module 4 of PCA) */
  #define  LOW_WMATCH         0x91    /* low(401) */
  #define  HIG_WMATCH         0x01    /* high(401) */
/* - digital value converted to % constant */
  #define  SV_RANGE100        0.030517578
  #define  PV_RANGE100        0.030517578
  #define  SV_MIN             819.2
  #define  PV_MIN             819.2
/* - % min and max value of D/A */
  #define  MIN_VAL            0x32a
  #define  MAX_VAL            0xfff
/* - error flag */
  #define  LOOSE_ERROR        loose_flag
/* - variable declaration */
   extern data unsigned char     wd_flag; /* backgnd */
   extern data bit              bar_flag; /* backgnd */
   extern data bit             fast_flag; /* backgnd */
   extern data bit               ad_test; /* backgnd */
   extern data bit               update; /* backgnd */
   extern data bit              err_flag; /* backgnd */
   extern int                tmp[MAXDPAR]; /* backgnd */
   extern unsigned char             busy; /* calcoef */
   extern bit                   ala_swap; /* display */
   extern unsigned char   ala_period_cnt; /* display */
   extern char                *err_msg[]; /* display */
   extern unsigned char         err_type; /* display */
   extern data unsigned char       dmode; /* keyboard */
   extern data unsigned char        dpar; /* keyboard */
   extern unsigned char         ddisplay; /* keyboard */
   extern data unsigned char      keybuf; /* keyboard */
   extern unsigned char         ddisplay; /* keyboard */
   extern p_shadow               vtabs[]; /* keyboard */
   extern unsigned char      cast_flag[]; /* keyboard */
   extern unsigned char      mode_change; /* keyboard */

/* - function declaration */
   extern void                  err_set(); /* keyboard */
   extern float     linear_and_convert(); /* linear */
   extern float                 convert(); /* linear */
   extern unsigned                 sa12(); /* sa12 */
/* INFORMATION : */
/* ------------------------------------------------
```

```
  Function : init_system_timer();

  Initilized the system timer to 100 msec (scantime)
  hardware dependent

  Generate an interrupt in software every 100 msec
  Oscillator frequency =    12 MHz
  PCA clock input      =    250 usec
      clock source     =    timer 0 auto reload mode
      clock output     :    from module 0 of PCA
  Calculate reload value for compare registers :
      100 msec
    --------------     =    400 counts
    250 usec/count
  Used registers
    CMOD   --  PCA Counter MODe register
    CCON   --  PCA Counter CONtrol register
    CCAPM0 --  PCA Compare/CAPture Module n register
    CH:CL  --  PCA 16-bits timebase set value register
  NOTE :
    Modify init_system_timer(), forground_routine() for the
    different hardware.
 */

/* ------------------------------------------
    Function : forground_routine();
    Type     : interrupt service routine
               using interrupt no. 6 of MCS-51 hardware
               and register blank1
    Job      : PID calculation
    Local variable:
      unsigned char j;        -  general counter
      float      prop;        -  propor tional term
      union    mv_out;        -  output buffer structure
    Description :
 */
```

```
/*
  XRAMCHK.H --  External RAM check header file
*/

#define  EVEN_DAT   0x55    /* even bits test pattern */
#define  ODD_DAT    0xaa    /* odd bits test pattern */
#define  BUMP_DAT   0x5a    /* bumping pattern */
```

```
# BACK -- Compact Digital PID Controller make file
#   This file is used for developping the all PID software.
#   The methods are compile, link, use ICE and also compile PAL software
#   When use this file the command is
#     MAKE BACK command_line
#   For ex.
#     MAKE BACK MAKEBACK
#   Compiling, linking and make the software output

#macro definitions
INCS = 83C51FA.H GLOBAL.H PID.H LCD.H KEYBOARD.H SA12.H CALCOEF.H\
       DISPLAY.H DISP_BAR.H DISP_DAT.H BACKGND.H REG51F.INC SYSTIME.H\
       XRAMCHK.H LINEAR.H
OBJS = timebase.obj keyboard.obj  sa12.obj calcoef.obj\
       display.obj disp_bar.obj disp_dat.obj backgnd.obj systime.obj\
       xramchk.obj linear.obj lcd.obj startup.obj

#default rules with special target (.SUFFIXES)
.SUFFIXES: .obj .c51 .a51

#dependency and command lines
MAKEBACK : $(OBJS)
  L51 @backgnd.l51
  OHS51 backgnd
  COPY  backgnd ..\ice51fx
  COPY  backgnd.m51 ..\ice51fx
  copy  backgnd.hex ..\ice51fx

ICE :
  cd..\ice51fx
  i51fx
  cd ..\source

TEST :
  cd..\ice51fx
  ict51fx
  cd ..\source

PALASM :
  cd \palasm2
  palasm
  cd \yong\source

SDT :
  cd \orcad\sdt
  draft
  cd \yong\source
PCB :
  cd \orcad\pcb
  pcb
  cd \yong\source
```

```
#default rules
.c51.obj :
  c51 $*.c51 LARGE PAGELENGTH(60)
  plmlst $*

.a51.obj :
  a51 $*.a51 PAGELENGTH(60)
  plmlst $*

$(OBJS) : $(INCS)
```

# ภาคผนวก ค

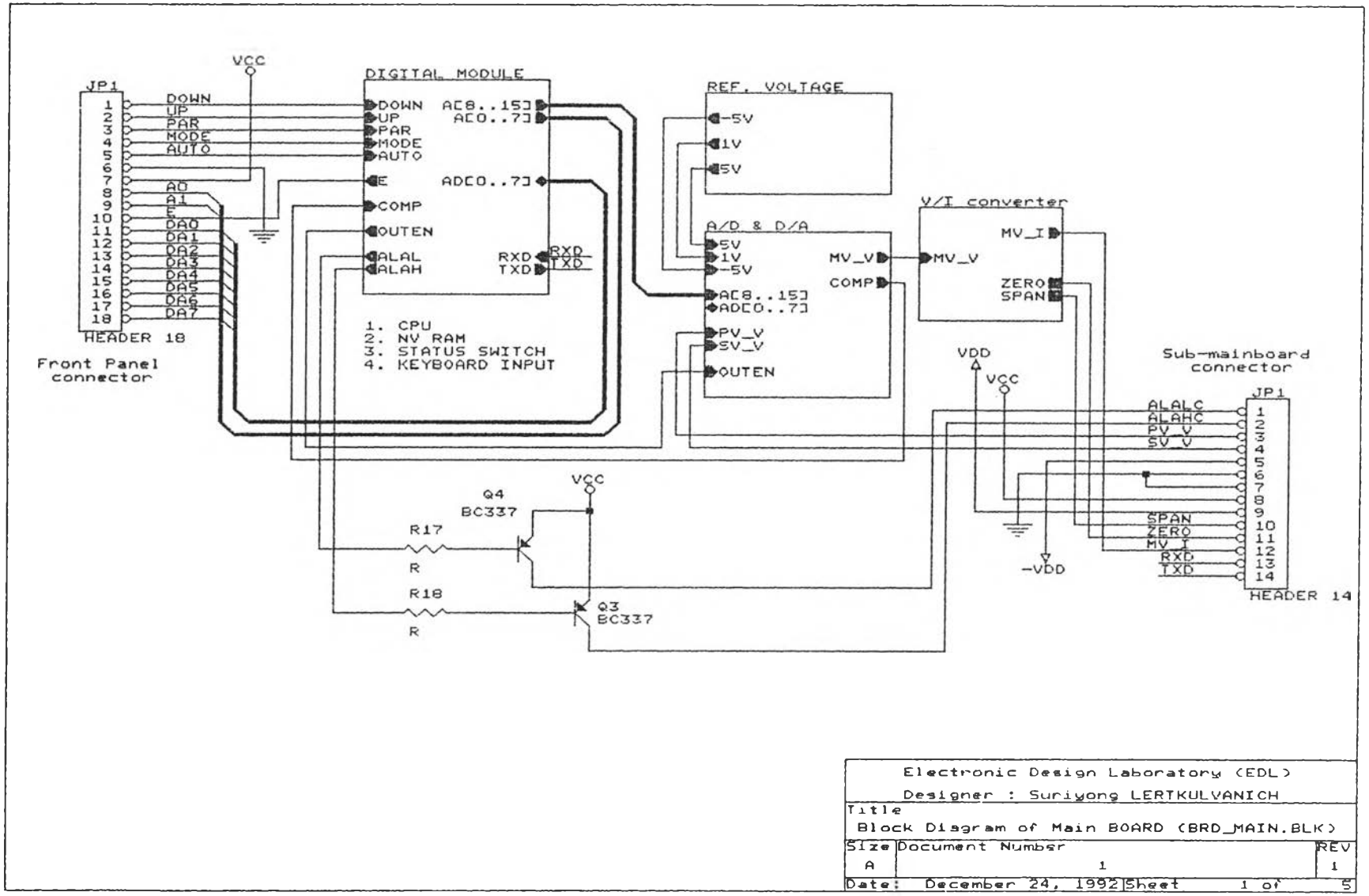
# โปรแกรมของ GAL

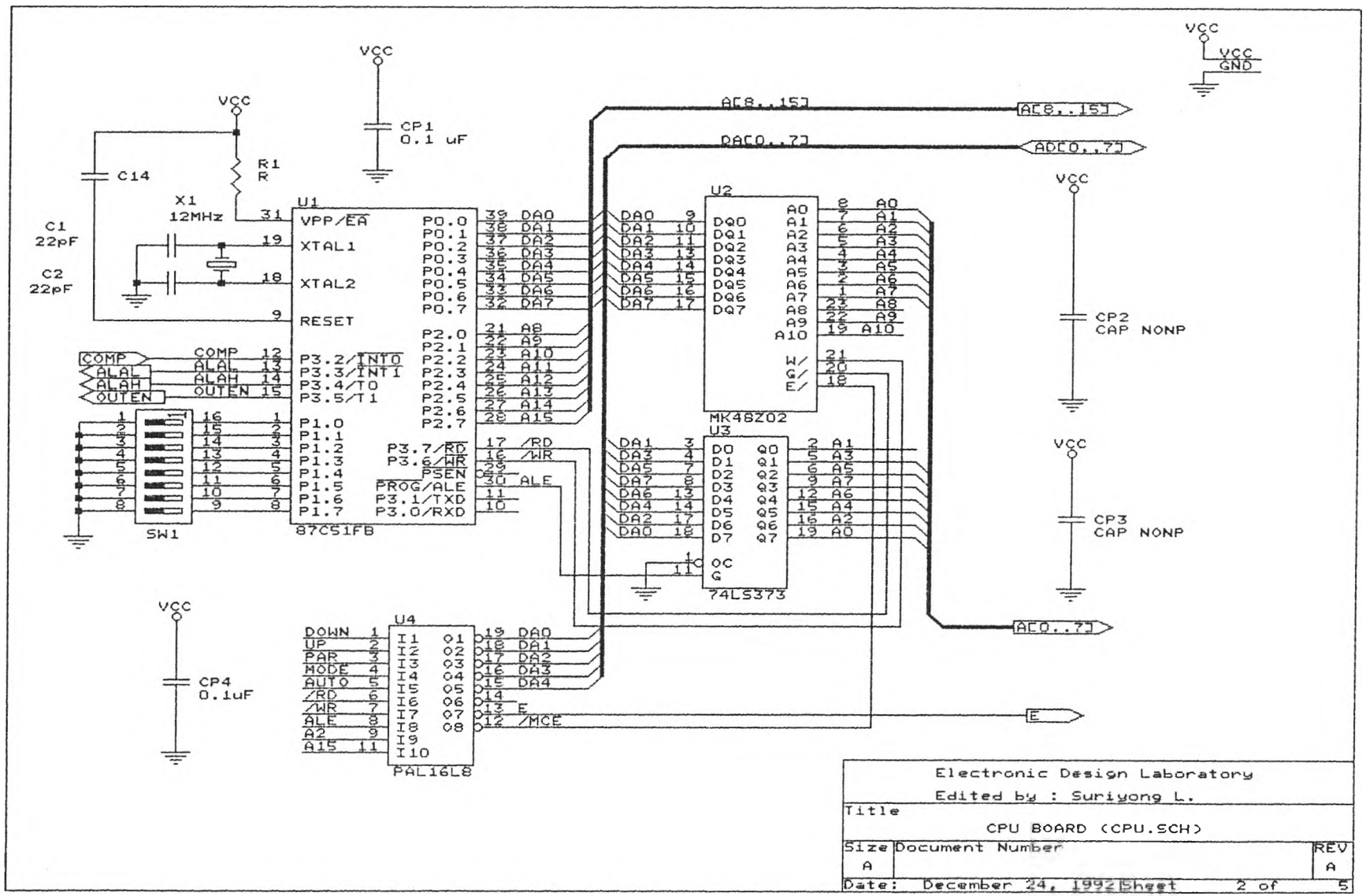
```
Title     Y_PAL
Pattern   Y_PAL.pds
Revision  A
Author    Suriyong LERTKULVANICH
Company   Electronic Design Laboratory, Chula, BKK.

CHIP      PAL1  PAL16L8
; 1    2      3      4      5      6       7       8       9       10
/RD    /WR    ALE    A15    A2     DOWN    UP      PAR     MODE    GND
;11    12     13     14     15     16      17      18      19       20
AUTO   DA0    DA1    DA2    DA3    DA4     NC      E       /MCE    VCC

;Pin description
;Input
;/RD      -- CPU ReaD
;/WR      -- CPU WRite
;ALE      -- Address Latch Enable
;A15      -- CPU address MSB bit
;A2       -- CPU address bit 2
;DOWN     -- DOWN key read bit
;UP       -- UP key read bit
;PAR      -- PARameter key read bit
;MODE     -- MODE key read bit
;AUTO     -- AUTO/manual key read bit

;Output
;DA[0..3] -- DIP switch input data to CPU
;E        -- lcd Enable signal
;/MCE     -- Memory Chip Enable and i/o select

EQUATIONS
E     = (RD+WR)*A15*A2
/MCE  = A15
/MCE.TRST = ALE
DA0   = DOWN
DA1   = UP
DA2   = PAR
DA3   = MODE
DA4   = AUTO
;Tristate command
DA0.TRST = RD*/A2*A15   ;High Z when /RD=1 ,A2 =1 or A15 = 0
DA1.TRST = RD*/A2*A15
DA2.TRST = RD*/A2*A15
DA3.TRST = RD*/A2*A15
DA4.TRST = RD*/A2*A15

SIMULATION

TRACE_ON /WR /RD  ALE  A15  A2  E  /MCE  DA0  DOWN
SETF     /WR /RD /ALE  /A15 /A2 DOWN UP PAR MODE AUTO
    FOR J := 3 TO 8 DO                    ; TEST DOWN INPUT
      BEGIN
```

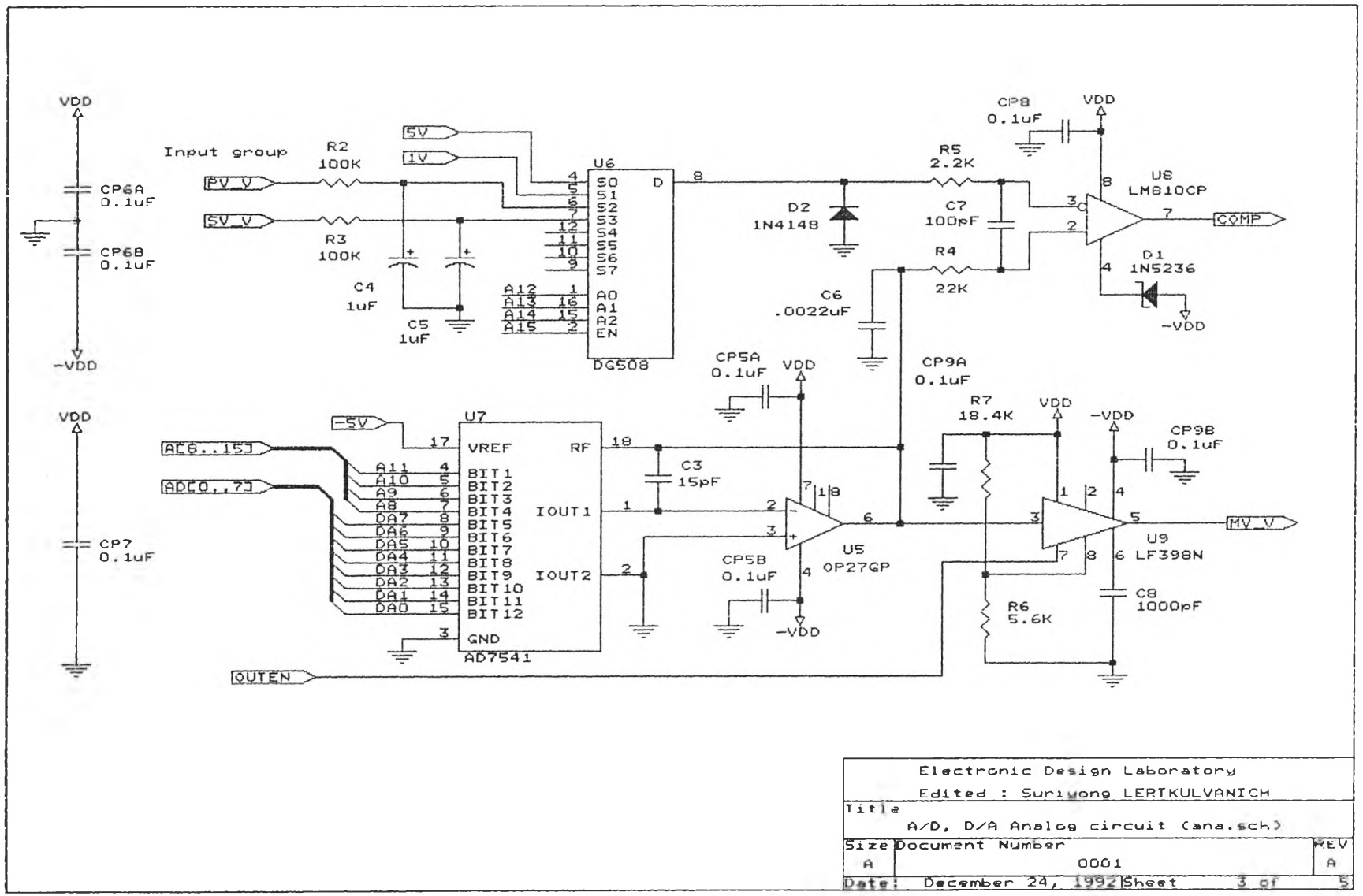
```
        SETF       RD    A15 /A2  DOWN
      END
    FOR J := 3 TO 8 DO
      BEGIN
        SETF       RD    A15 /A2 /DOWN
      END
                                                ; TEST E
    SETF  /WR  RD  A15 /A2
    SETF  /WR  RD  A15  A2
    SETF  /WR  RD /A15 /A2
    SETF  /WR  RD /A15  A2
    SETF   WR /RD  A15 /A2
    SETF   WR /RD  A15  A2
    SETF   WR /RD /A15 /A2
    SETF   WR /RD /A15  A2
                                                ; TEST /MCE
    SETF           /A15     /ALE
    SETF           /A15      ALE
    SETF            A15     /ALE
    SETF            A15      ALE
SETF       /WR /RD ALE A15 A2      ; inactive all
TRACE_OFF
```

# ภาคผนวก ง

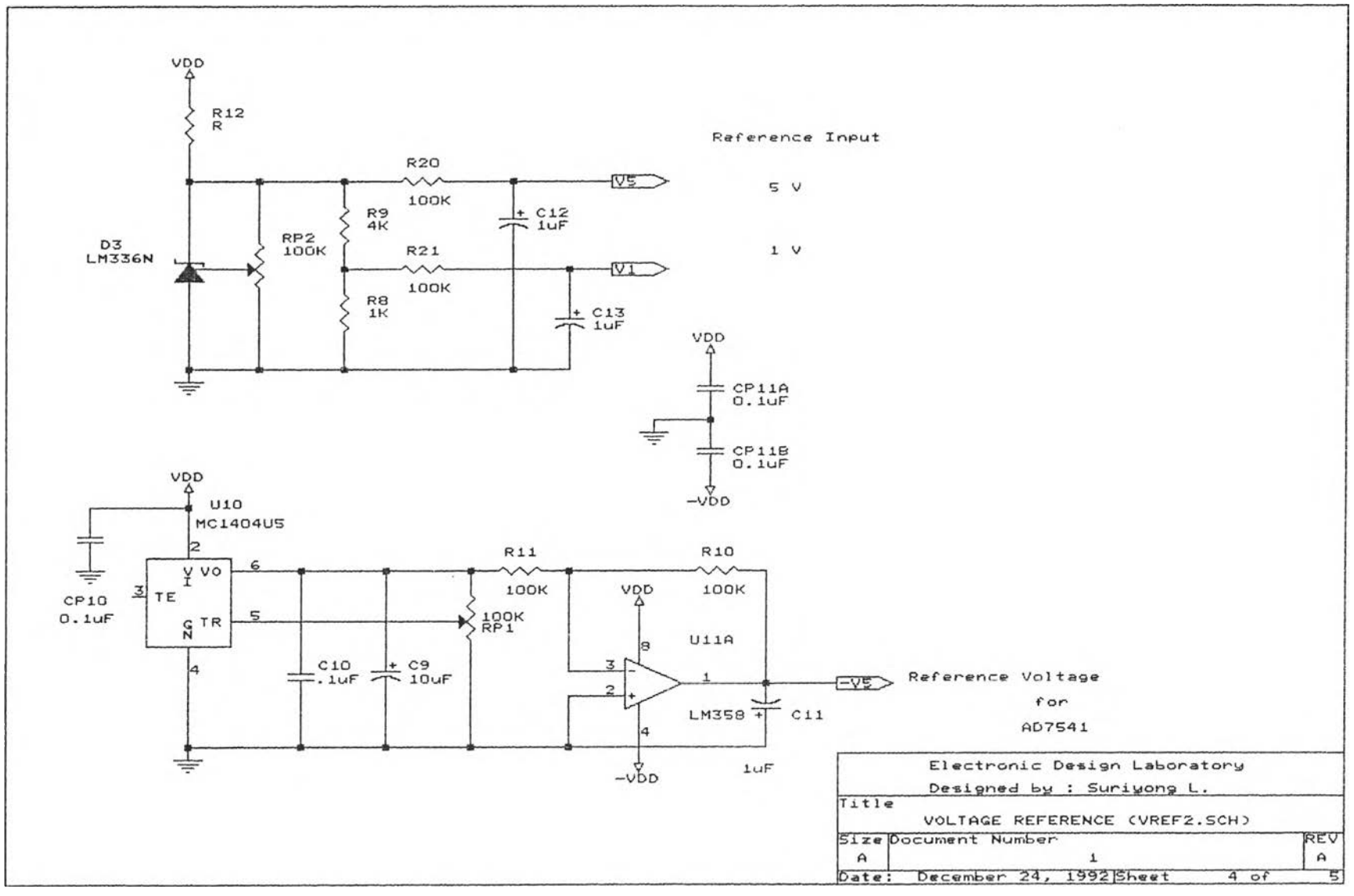
# วงจรชองตัวควบคุม PID เชิงเลขขนาดกะทัดรัด

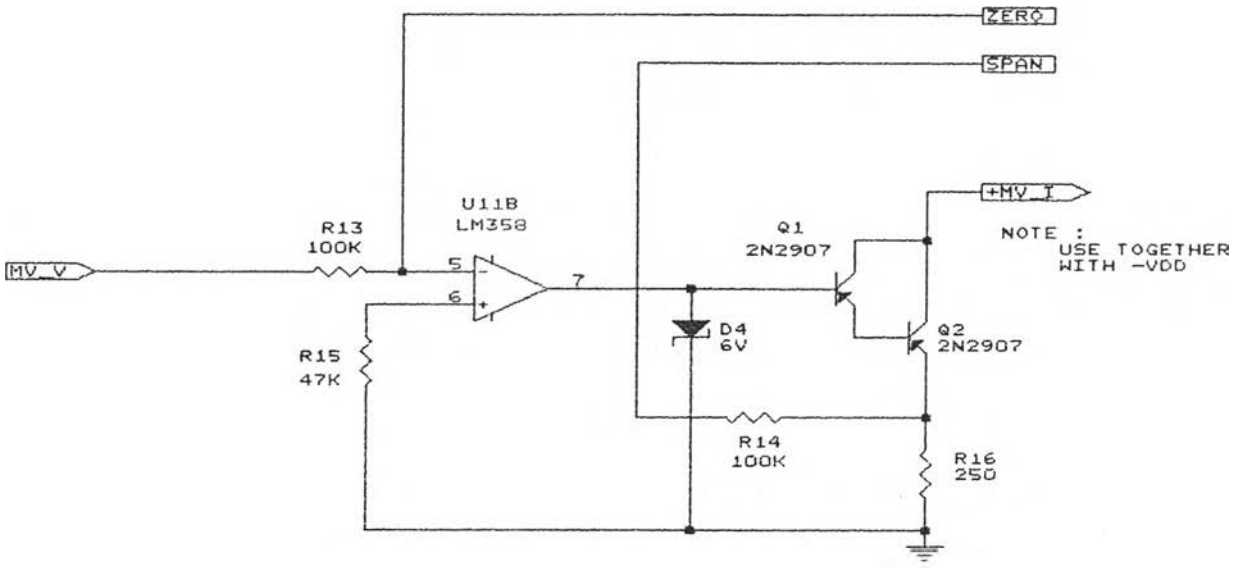

JP1
DOWN
UP
PAR
MODE
AUTO
AD
A1
E
DA0
DA1
DA2
DA3
DA4
DA5
DA6
DA7
HEADER 18
Front Panel connector
VCC
DIGITAL MODULE
DOWN
UP
PAR
MODE
AUTO
E
COMP
OUTEN
ALAL
ALAH
AC8..15]
AC0..7]
ADC0..7]
RXD
TXD
1. CPU
2. NV RAM
3. STATUS SWITCH
4. KEYBOARD INPUT
REF. VOLTAGE
-5V
1V
5V
A/D & D/A
5V
1V
-5V
AC8..15]
ADC0..7]
PV_V
SV_V
OUTEN
MV_V
COMP
V/I converter
MV_I
MV_V
ZERO
SPAN
VDD
VCC
-VDD
Sub-mainboard connector
JP1
ALALC
ALAHC
PV_V
SV_V
SPAN
ZERO
MV_I
RXD
TXD
HEADER 14
Q4
BC337
R17
R
R18
R
Q3
BC337
Electronic Design Laboratory (EDL)
Designer : Suriyong LERTKULVANICH
Title
Block Diagram of Main BOARD (BRD_MAIN.BLK)
Size
A
Document Number
1
REV
1
Date: December 24, 1992
Sheet 1 of 5


VCC
CP1
0.1 uF
C14
R1
R
X1
12MHz
C1
22pF
C2
22pF
U1
VPP/EA
XTAL1
XTAL2
RESET
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.7/RD
P3.6/WR
PSEN
PROG/ALE
P3.1/TXD
P3.0/RXD
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
87C51FB
COMP
ALAL
ALAH
OUTEN
SW1
A[8..15]
DA[0..7]
U2
MK48Z02
U3
74LS373
OC
G
CP2
CAP NONP
CP3
CAP NONP
A[0..7]
E
/RD
/WR
ALE
/MCE
U4
PAL16L8
DOWN
UP
PAR
MODE
AUTO
CP4
0.1uF
VCC
GND
Electronic Design Laboratory
Edited by : Suriyong L.
Title
CPU BOARD (CPU.SCH)
Size
A
Document Number
REV
A
Date: December 24, 1992
Sheet 2 of 5




| | | |
|---|---|---|
| Electronic Design Laboratory | | |
| Edited : Suriwong LERTKULVANICH | | |
| Title: A/D, D/A Analog circuit (ana.sch) | | |
| Size: A | Document Number: 0001 | REV: A |
| Date: December 24, 1992 | Sheet 3 of 5 | |


VDD
R12
R
Reference Input
R20
100K
V5
5 V
R9
4K
C12
1uF
D3
LM336N
RP2
100K
R21
100K
V1
1 V
R8
1K
C13
1uF
VDD
CP11A
0.1uF
CP11B
0.1uF
-VDD
VDD
U10
MC1404U5
2
V VO
I
3
TE
G TR
N
6
5
4
CP10
0.1uF
C10
.1uF
C9
10uF
R11
100K
100K
RP1
R10
100K
VDD
8
U11A
3
2
1
4
LM358
C11
1uF
-VDD
-VE
Reference Voltage
for
AD7541
Electronic Design Laboratory
Designed by : Suriyong L.
Title
VOLTAGE REFERENCE (VREF2.SCH)
Size
A
Document Number
1
REV
A
Date: December 24, 1992
Sheet 4 of 5




From : Circuit For Electronic Engineers

Edited by : Samuel Weber

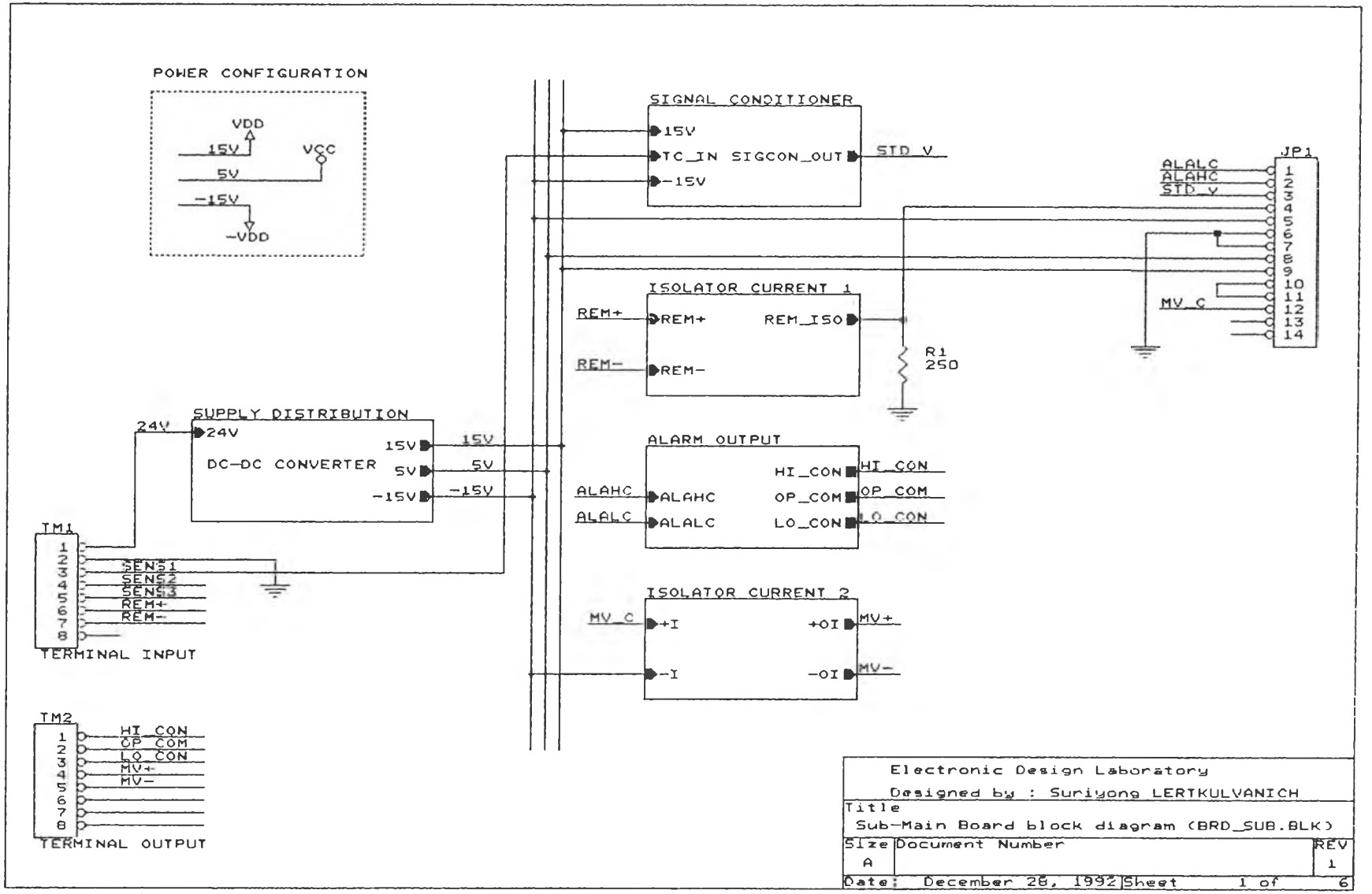
Item : 23 Instrument circuits

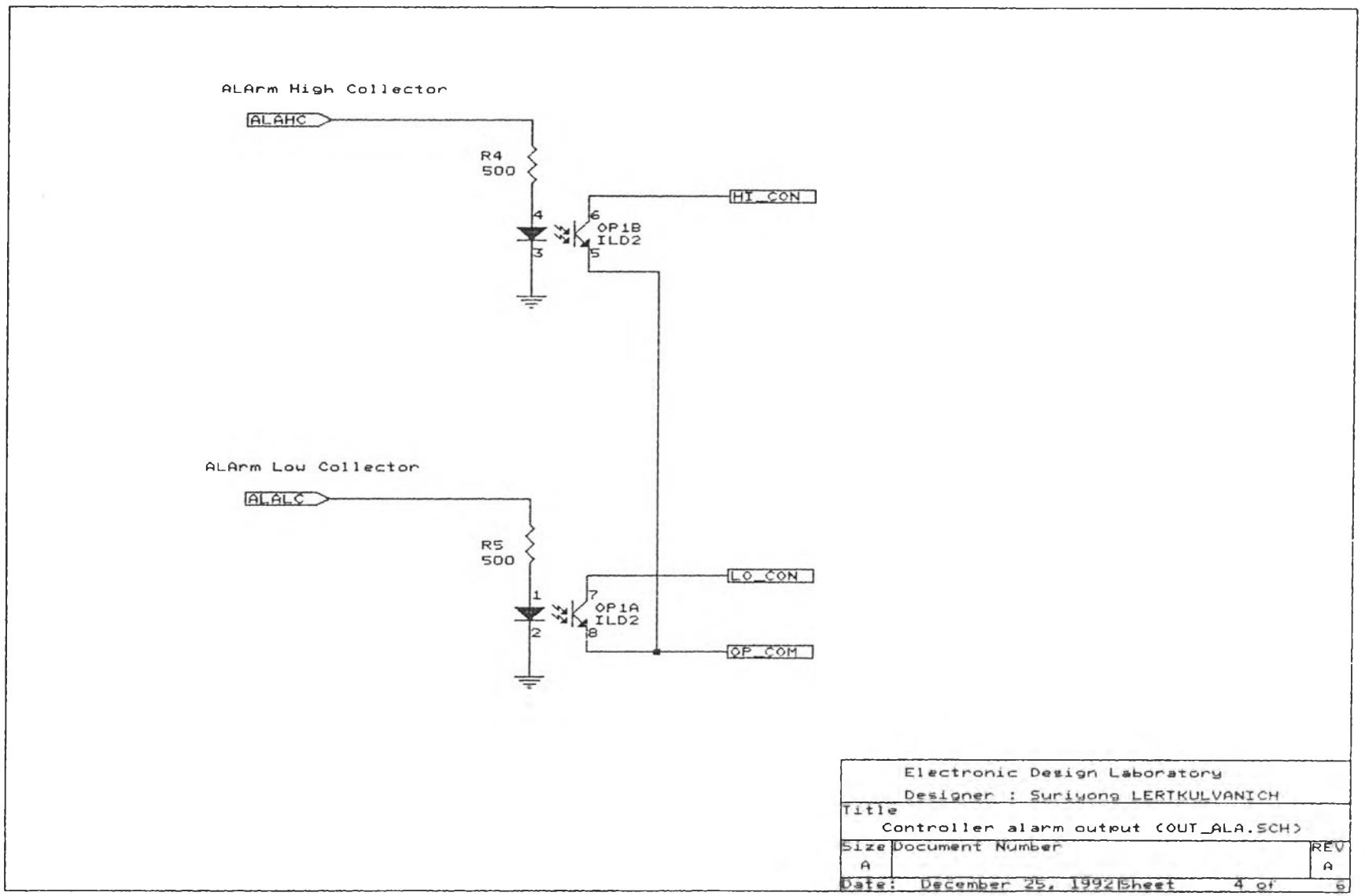
Title : Voltage-to-current converter
for process-control systems

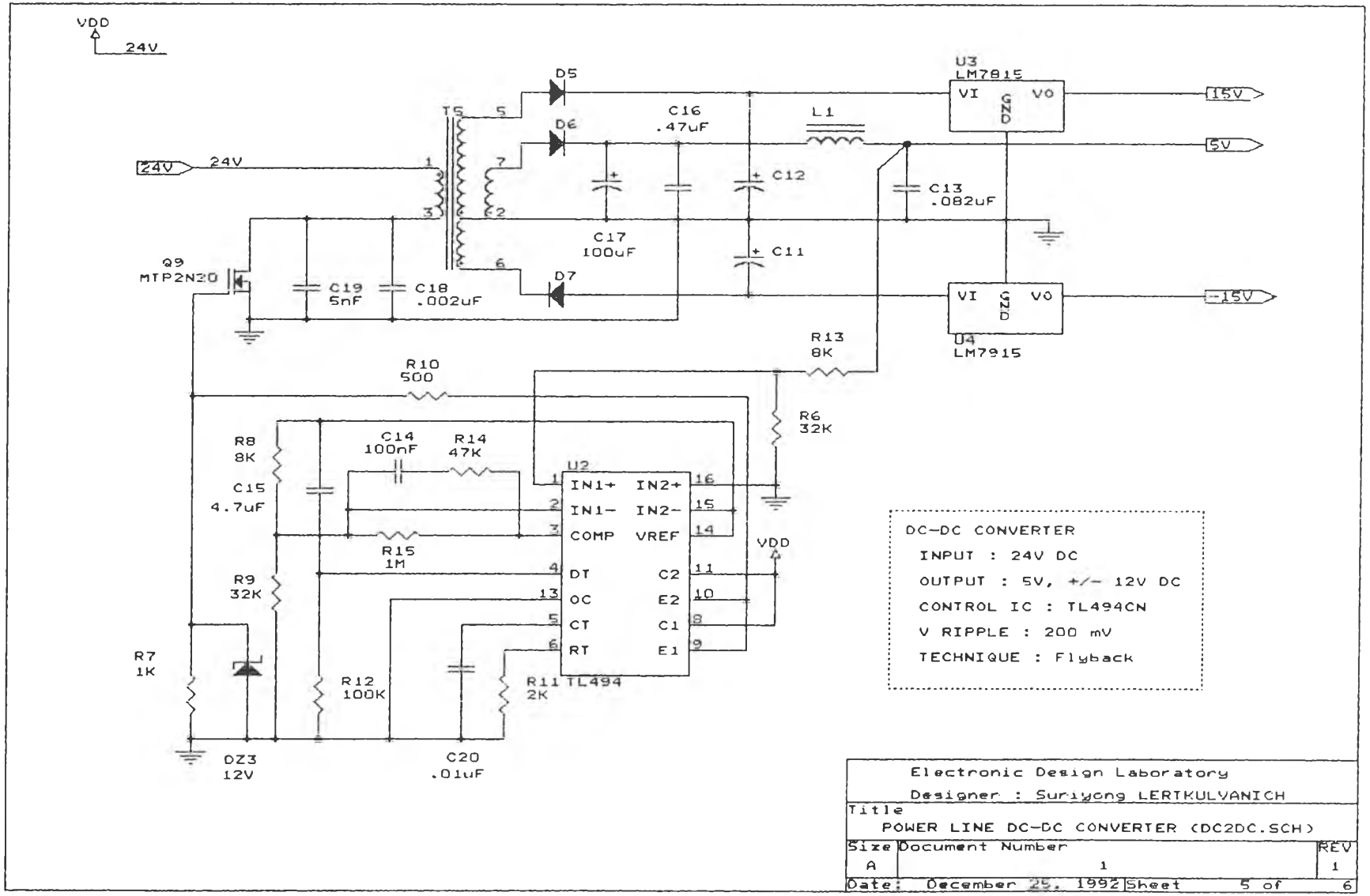
by : Harry L. Trietley, Jr

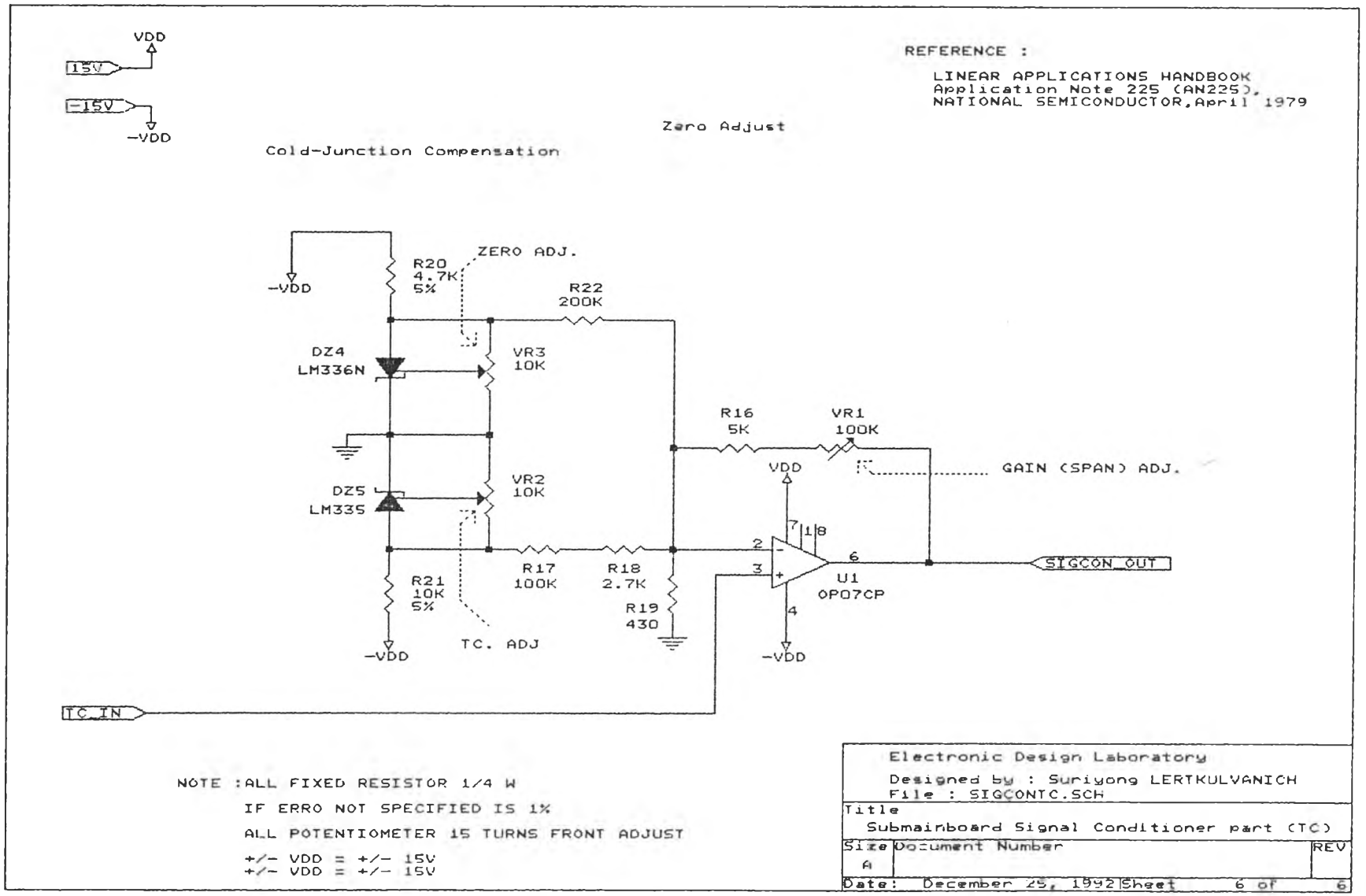
P 170 McGraw Hill 1977

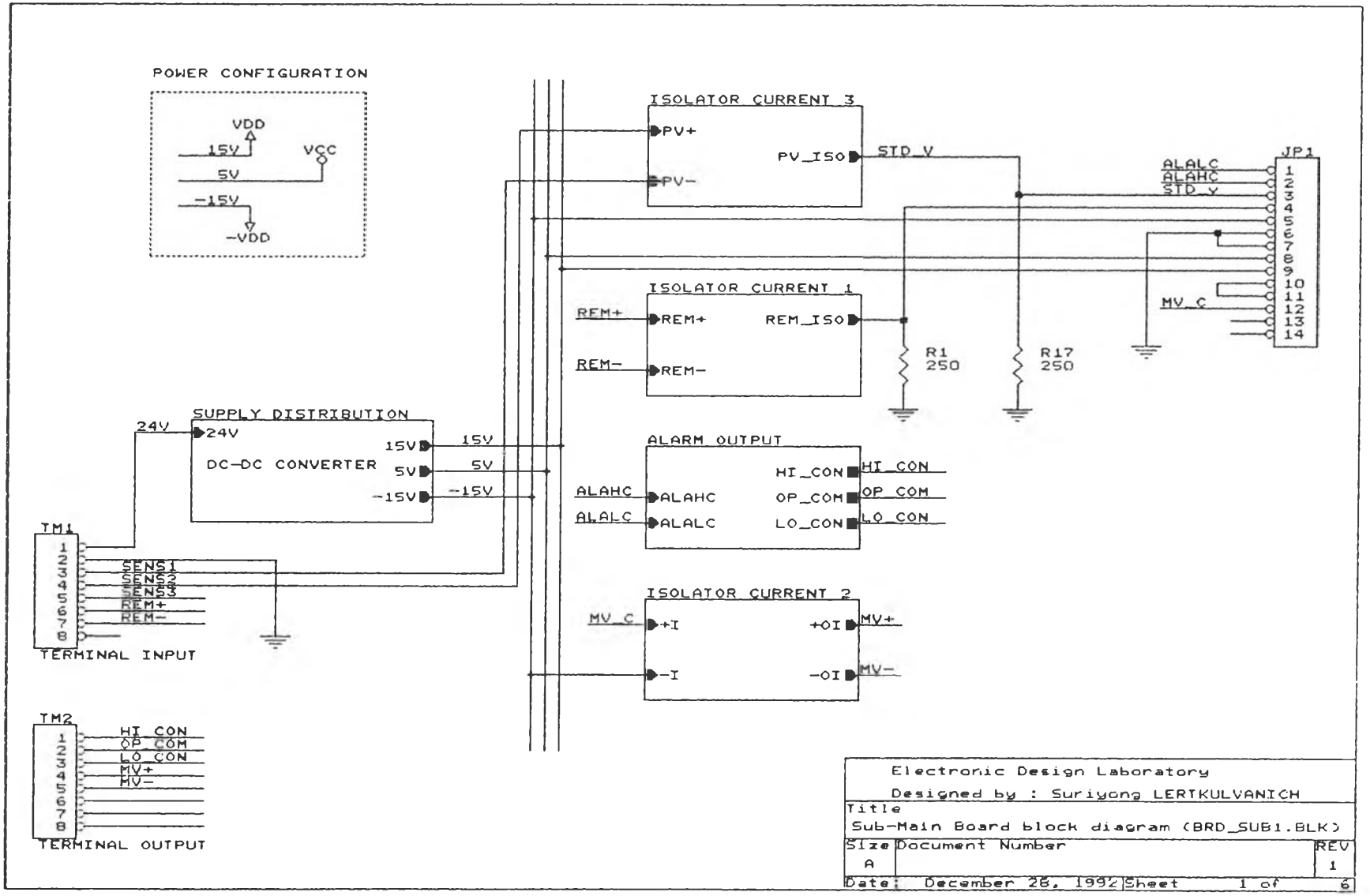
| | | |
|---|---|---|
| Electronic Design Laboratory<br>Written by : Suriyong L.<br>Owner : Suriyong L. | | |
| Title<br>V/I converter (floating ground type) | | |
| Size<br>A | Document Number | REV<br>1 |
| Date: December 24, 1992 | Sheet 5 of 5 | |

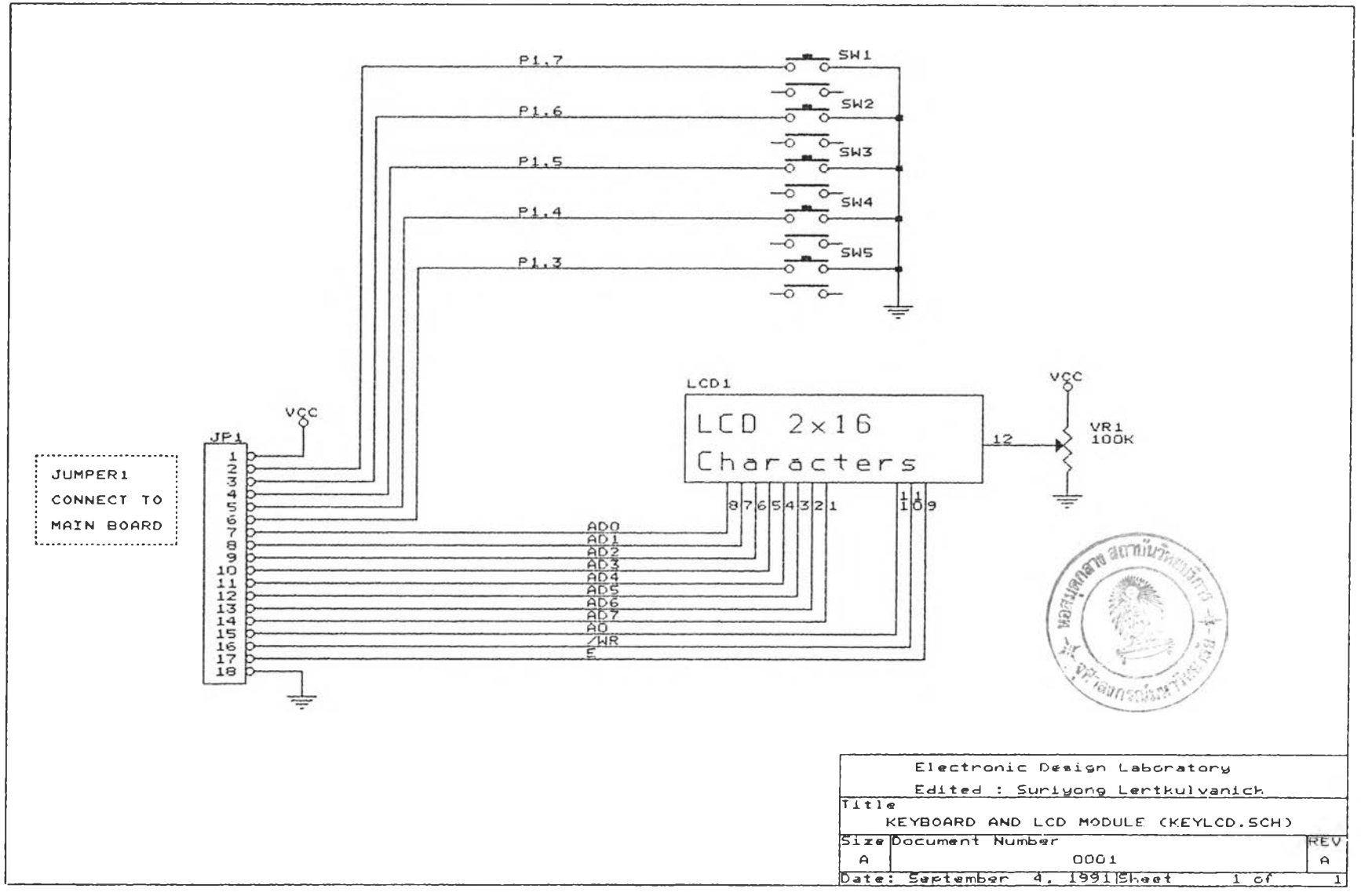

POWER CONFIGURATION
VDD
15V
VCC
5V
-15V
-VDD
SIGNAL CONDITIONER
15V
TC_IN SIGCON_OUT
-15V
STD_V
JP1
ALALC
ALAHC
STD_V
MV_C
ISOLATOR CURRENT 1
REM+
REM+
REM_ISO
REM-
REM-
R1
250
SUPPLY DISTRIBUTION
24V
24V
DC-DC CONVERTER
15V
5V
-15V
15V
5V
-15V
ALARM OUTPUT
ALAHC
ALALC
HI_CON
OP_COM
LO_CON
HI_CON
OP_COM
LO_CON
TM1
SENS1
SENS2
SENS3
REM+
REM-
TERMINAL INPUT
ISOLATOR CURRENT 2
MV_C
+I
-I
+OI
-OI
MV+
MV-
TM2
HI_CON
OP_COM
LO_CON
MV+
MV-
TERMINAL OUTPUT
Electronic Design Laboratory
Designed by : Suriyong LERTKULVANICH
Title
Sub-Main Board block diagram (BRD_SUB.BLK)
Size
A
Document Number
REV
1
Date: December 28, 1992
Sheet 1 of 6


ALArm High Collector
ALAHC
R4
500
4
3
6
5
OP1B
ILD2
HI_CON
ALArm Low Collector
ALALC
R5
500
1
2
7
8
OP1A
ILD2
LO_CON
OP_COM
Electronic Design Laboratory
Designer : Suriyong LERTKULVANICH
Title
Controller alarm output (OUT_ALA.SCH)
Size
A
Document Number
REV
A
Date: December 25, 1992
Sheet 4 of 5


VDD
24V
D5
D6
T5
C16
.47uF
L1
U3
LM7815
VI
GND
VO
15V
5V
24V
24V
C12
C13
.082uF
C17
100uF
C11
Q9
MTP2N20
C19
5nF
C18
.002uF
D7
VI
GND
VO
-15V
U4
LM7915
R13
8K
R10
500
R6
32K
R8
8K
C14
100nF
R14
47K
U2
C15
4.7uF
IN1+
IN2+
IN1-
IN2-
COMP
VREF
R15
1M
VDD
R9
32K
DT
C2
OC
E2
CT
C1
RT
E1
R7
1K
R12
100K
R11
2K
TL494
DZ3
12V
C20
.01uF
DC-DC CONVERTER
INPUT : 24V DC
OUTPUT : 5V, +/- 12V DC
CONTROL IC : TL494CN
V RIPPLE : 200 mV
TECHNIQUE : Flyback
Electronic Design Laboratory
Designer : Suriyong LERTKULVANICH
Title
POWER LINE DC-DC CONVERTER (DC2DC.SCH)
Size
A
Document Number
1
REV
1
Date: December 25, 1992
Sheet 5 of 6


VDD
15V
-15V
-VDD
REFERENCE :
LINEAR APPLICATIONS HANDBOOK
Application Note 225 (AN225),
NATIONAL SEMICONDUCTOR,April 1979
Zero Adjust
Cold-Junction Compensation
R20
4.7K
5%
ZERO ADJ.
R22
200K
-VDD
DZ4
LM336N
VR3
10K
R16
5K
VR1
100K
GAIN (SPAN) ADJ.
VDD
DZ5
LM335
VR2
10K
R17
100K
R18
2.7K
R19
430
R21
10K
5%
TC. ADJ
-VDD
-VDD
U1
OP07CP
SIGCON_OUT
TC_IN
NOTE :ALL FIXED RESISTOR 1/4 W
IF ERRO NOT SPECIFIED IS 1%
ALL POTENTIOMETER 15 TURNS FRONT ADJUST
+/- VDD = +/- 15V
+/- VDD = +/- 15V
Electronic Design Laboratory
Designed by : Suriyong LERTKULVANICH
File : SIGCONTC.SCH
Title
Submainboard Signal Conditioner part (TC)
Size
A
Document Number
REV
Date: December 25, 1992 Sheet 6 of 6


POWER CONFIGURATION
VDD
15V
VCC
5V
-15V
-VDD
ISOLATOR CURRENT 3
PV+
PV-
PV_ISO
STD_V
ISOLATOR CURRENT 1
REM+
REM-
REM_ISO
R1
250
R17
250
JP1
ALALC
ALAHC
STD_V
MV_C
SUPPLY DISTRIBUTION
24V
DC-DC CONVERTER
15V
5V
-15V
ALARM OUTPUT
ALAHC
ALALC
HI_CON
OP_COM
LO_CON
TM1
SENS1
SENS2
SENS3
REM+
REM-
TERMINAL INPUT
ISOLATOR CURRENT 2
MV_C
+I
-I
+OI
-OI
MV+
MV-
TM2
HI CON
OP COM
LO CON
MV+
MV-
TERMINAL OUTPUT
Electronic Design Laboratory
Designed by : Suriyong LERTKULVANICH
Title
Sub-Main Board block diagram (BRD_SUB1.BLK)
Size
A
Document Number
REV
1
Date: December 28, 1992
Sheet 1 of 6


P1.7
P1.6
P1.5
P1.4
P1.3
SW1
SW2
SW3
SW4
SW5
VCC
JP1
JUMPER1
CONNECT TO
MAIN BOARD
LCD1
LCD 2x16
Characters
VCC
VR1
100K
12
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A0
/WR
E
Electronic Design Laboratory
Edited : Suriyong Lertkulvanich
Title
KEYBOARD AND LCD MODULE (KEYLCD.SCH)
Size
A
Document Number
0001
REV
A
Date: September 4, 1991
Sheet 1 of 1

# ประวัติผู้เขียน

นาย สุริยงค์ เลิศกุลวาณิชย์ เกิดวันที่ 4 ตุลาคม พ,ศ, 2508 ที่กรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า จากคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าธนบุรีในปีการศึกษา 2531 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต ณ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย