เอกสารอ้างอิง

1. Michael P.Lukas, Distributed Control Systems: Their Evaluation and Design, 1986
2. YEW, Model SLPC,SLMC and SPLR Programmable Instruments Functions and Applications, Technical Information
3. FUJI, Compact Controller F User's Manual
4. TOSHIBA, One-loop Controller 211D User's Manual
5. สมบูรณ์ จงชัยกิจ, เอกสารประกอบการบรรยายเรื่อง การควบคุมแบบอัตโนมัติและการควบคุมแบบ PID, สมาคมเทคโนโลยี(ไทย-ญี่ปุ่น)
6. Karl J.Astrom and Bjorn Wittenmark, Computer Controlled System Theory and Design, Prentice-Hall, Inc. 1984.
7. Nicholas J.Krikelis and Spilios D.Fassois, Microprocessor Implementation of PID Controllers and Lead-Lag Compensators, IEEE trans. of Industrial Electronics, Vol.IE-31,NO.1, Feb 1984
8. Michael Andrews, Programming Microprocessor Interfaces for Control and Instrumentation, Prentice-Hall, 1982
9. John Zarrella, Language Translators, Microcomputer Applications, California, 1982
10. Joseph J.F. Cavanagh, Digital Computer Arithmetic Design and Implementation, McGraw-Hill, 1985
11. Robert J.Bibbero, Microprocessors in Instruments and Control, John Wiley & Sons, 1977
12. ALS, ALS CARD SYSTEM USER'S MANUAL 8088 SYSTEM, INDIA
13. Intel, PL/M-86 USER'S GUIDE, 1985
14. ____. 8086/8087/8088 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL, 1980
15. ____. iAPX 86,88 Family Utilities USER'S GUIDE, 1982
16. สมบูรณ์ จงชัยกิจ,กฤษดา วิศวธีรานนท์และอำนวย แสงวิโรจนพัฒน์, ตัวควบคุมเชิงเลขสำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่อง, การประชุมวิศวกรรมไฟฟ้า 9 สถาบัน ครั้งที่ 11, พ.ศ. 2532

ภาคผนวก

ภาคผนวก ก. ตารางเปรียบเทียบตัวควบคุมเชิงเลขในท้องตลาด

บริษัทผู้ผลิต	YEW	Shimadzu	Yametake	FOXBORO	Toshiba	Hithachi
Analog Input (1-5 V)	5	4	5	4	4	5
Digital Input	3	2	5	2	3	3
Analog Output : (1-5 V)	2	1	3	1	2	2
: (4-20 mA)	1	1	1	1	1	1
Digital Output	3	3	4	2	3	3
Basic Control Module (PID)	/	/	/	/	/	/
Cascade Control Module	/	X	X	X	/	/
Function (+,-,x,÷, ,timer, lead,lag,..)	/	/	X	/	/	/
Logical Function (AND,OR, NOT)	/	/	X	/	/	/

3.3.6 Bus connector

3.3.6.1 Bus connector (64 pin euroconnector) details - signal flow is referred to the processor card.

		ROW 'a'			ROW 'c'		
	PIN NO.	MNEMONIC	SIGNAL FLOW	DESCRIPTION	MNEMONIC	SIGNAL FLOW	DESCRIPTION
LOGIC	1	GND	IN	LOGIC GROUND	GND	IN	LOGIC GROUND
POWER	32	GND	IN	LOGIC GROUND	GND	IN	LOGIC GROUND
BUS	2	+5V	IN	LOGIC POWER	+5V	IN	LOGIC POWER
	31	+5V	IN	LOGIC POWER	+5V	IN	LOGIC POWER
	3	-12V	IN	ANALOG POWER	-12V	IN	ANALOG POWER
	4	+12V	IN	ANALOG POWER	+12V	IN	ANALOG POWER
	29	A.GND	IN	ANALOG COM	A.GND	IN	ANALOG COM
DATA BUS	5	DØ	IN/OUT	LOW-ORDER DATA BUS	D1	IN/OUT	LOW-ORDER DATA BUS
	6	D2	IN/OUT	LOW-ORDER DATA BUS	D3	IN/OUT	LOW-ORDER DATA BUS
	7	D4	IN/OUT	HIGH-ORDER DATA BUS	D5	IN/OUT	HIGH-ORDER DATA BUS
	8	D6	IN/OUT	HIGH-ORDER DATA BUS	D7	IN/OUT	HIGH-ORDER DATA BUS

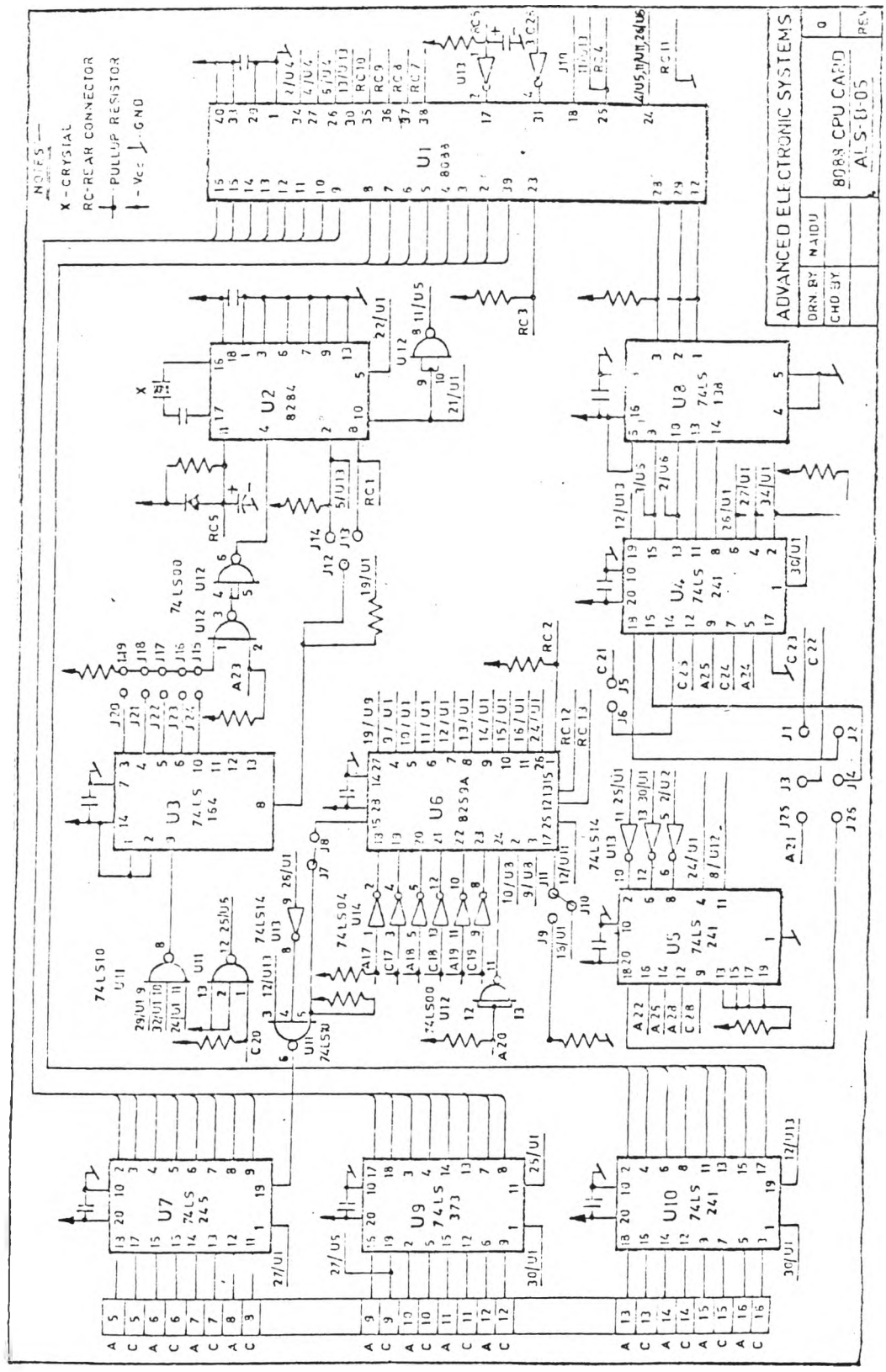
	PIN NO.	ROW 'a' MNEMONIC	SIGNAL FLOW	DESCRIPTION	ROW 'c' MNEMONIC	SIGNAL FLOW	DESCRIPTION
ADDRESS	9	AØ	OUT	LOW-ORDER ADDRESS BUS	A1	OUT	LOW-ORDER ADDRESS BUS
BUS	10	A2	OUT	-do-	A3	OUT	-do-
	11	A4	OUT	-do-	A5	OUT	-do-
	12	A6	OUT	-do-	A7	OUT	-do-
	13	A8	OUT	HIGH-ORDER ADDRESS BUS	A9	OUT	HIGH-ORDER ADDRESS BUS
	14	A10	OUT	-do-	A11	OUT	-do-
	15	A12	OUT	-do-	A13	OUT	-do-
	16	A14	OUT	-do-	A15	OUT	-do-

		ROW 'a'			ROW 'c'		
	PIN NO.	MNEMONIC	SIGNAL FLOW	DESCRIPTION	MNEMONIC	SIGNAL FLOW	DESCRIPTION
CONTROL BUS	17	$\overline{INT0}$	IN	Interrupt line	$\overline{INT1}$	IN	Interrupt line
	18	$\overline{INT2}$	IN	-do-	$\overline{INT3}$	IN	-do-
	19	$\overline{INT4}$	IN	-do-	$\overline{INT5}$	IN	-do-
	20	$\overline{INT6}$	IN	-do-	$\overline{INT7}$	IN	-do-
	21	$\overline{CCLK}$	OUT	Constant clock	--	--	reserved
	22	$\overline{INTA}$	OUT	Interrupt acknowledge	$\overline{INH2}$	OUT	Inhibit 2
	23	$\overline{XACK}$	IN	Reddy input	$\overline{INH1}$	OUT	Inhibit 1
	24	$\overline{IORC}$	OUT	I/o Read	$\overline{IOWC}$	OUT	I/o write
	25	$\overline{MRDC}$	OUT	Memory read	$\overline{MWTC}$	OUT	Memory write
	26	$\overline{BUSY}$	OUT	Bus busy	$\overline{BREQ}$	IN	Bus request
	27	$\overline{BPRN}$	IN	Bus priority in	$\overline{BPRO}$	OUT	Bus priority out
	28	$\overline{BCLK}$	OUT	Bus clock	$\overline{RESET}$	OUT	Initialisation signal
	30	--	--	reserved	--	--	reserved

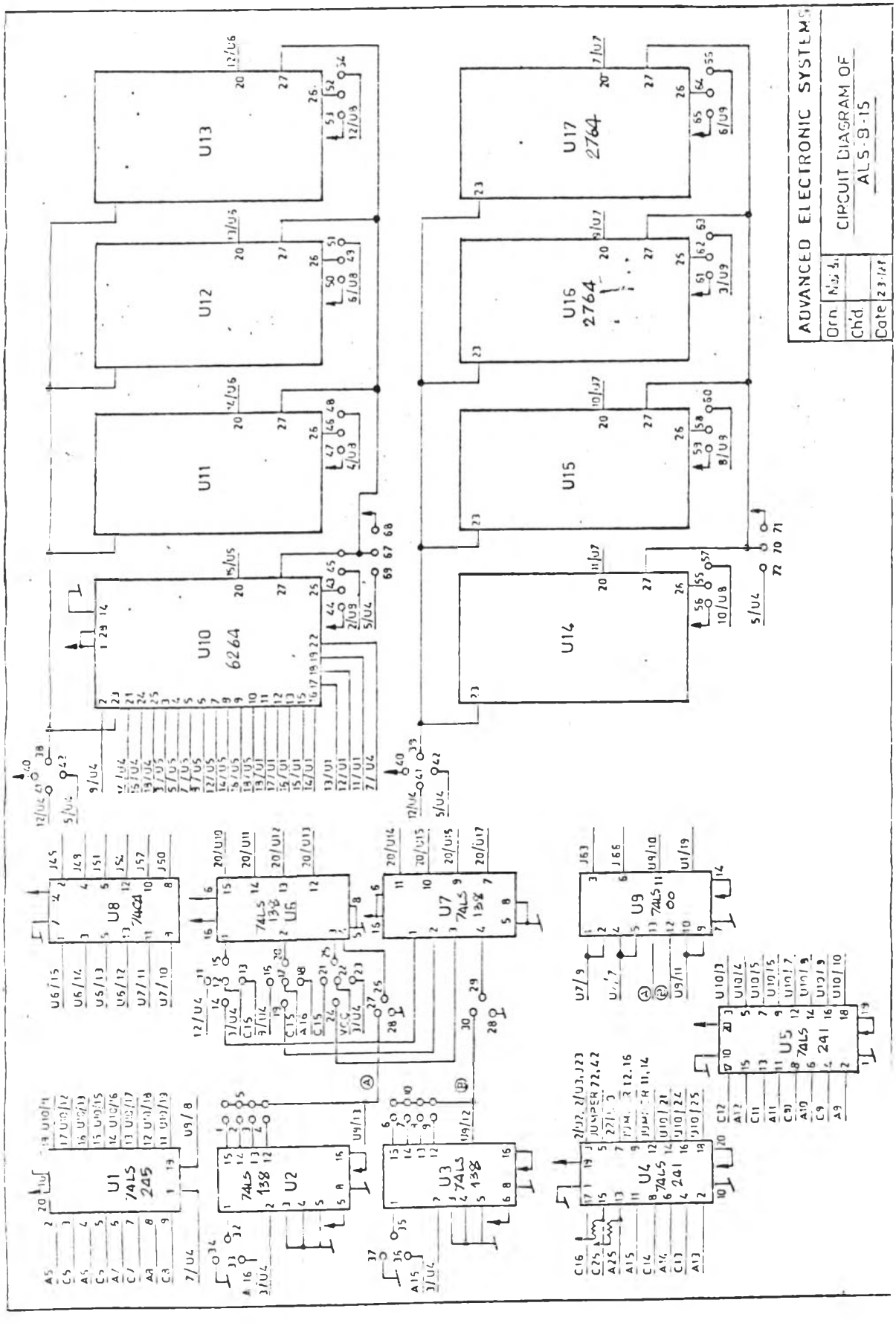
Note: 1) A bar over the mnemonic indicates that the signal is active when it is at a logic 'LO' level.

ภาคผนวก ค. วงจรของตัวควบคุมเชิงเลขชนิดโปรแกรมได้

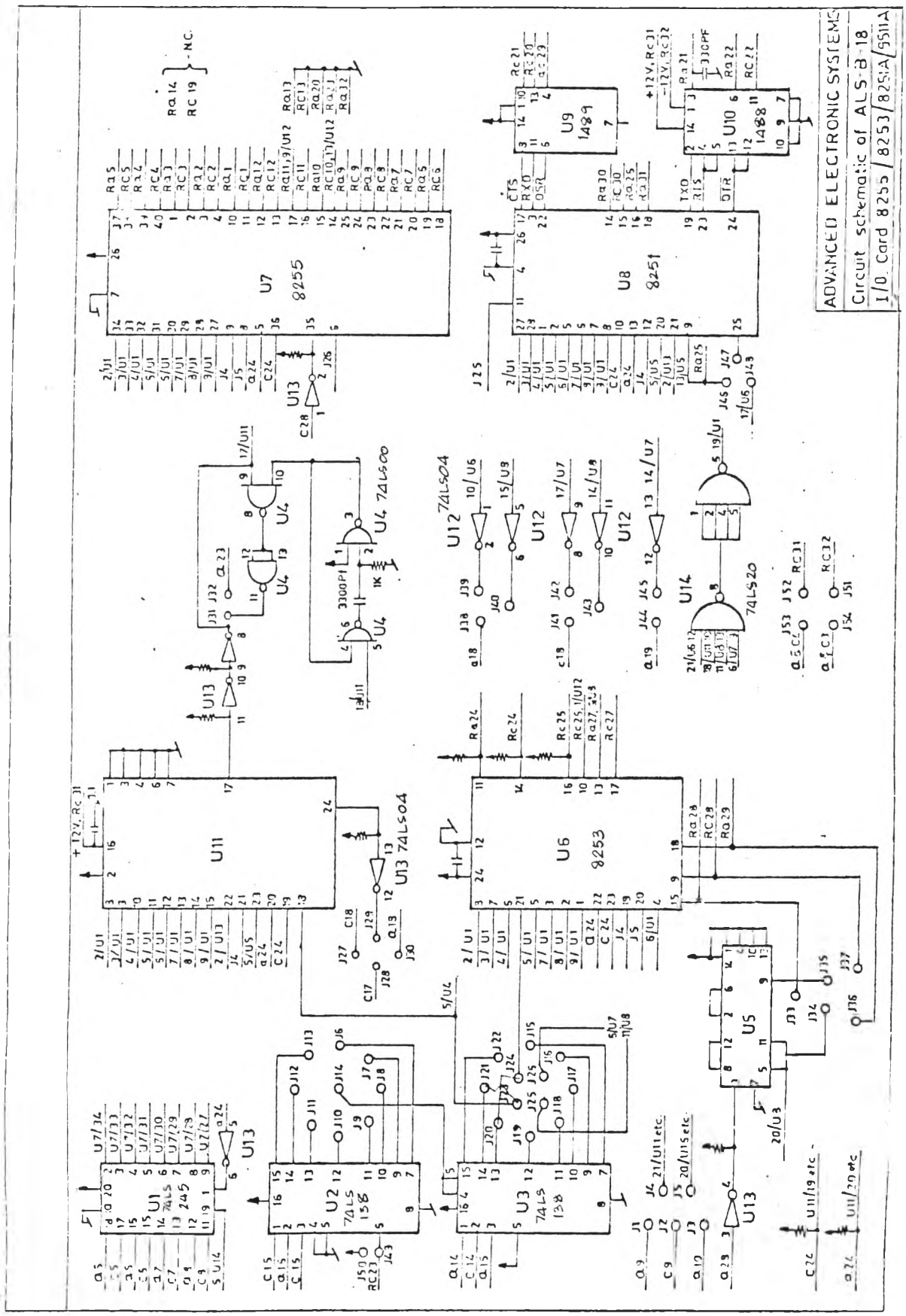
ค.1 วงจรส่วนประมวลผล



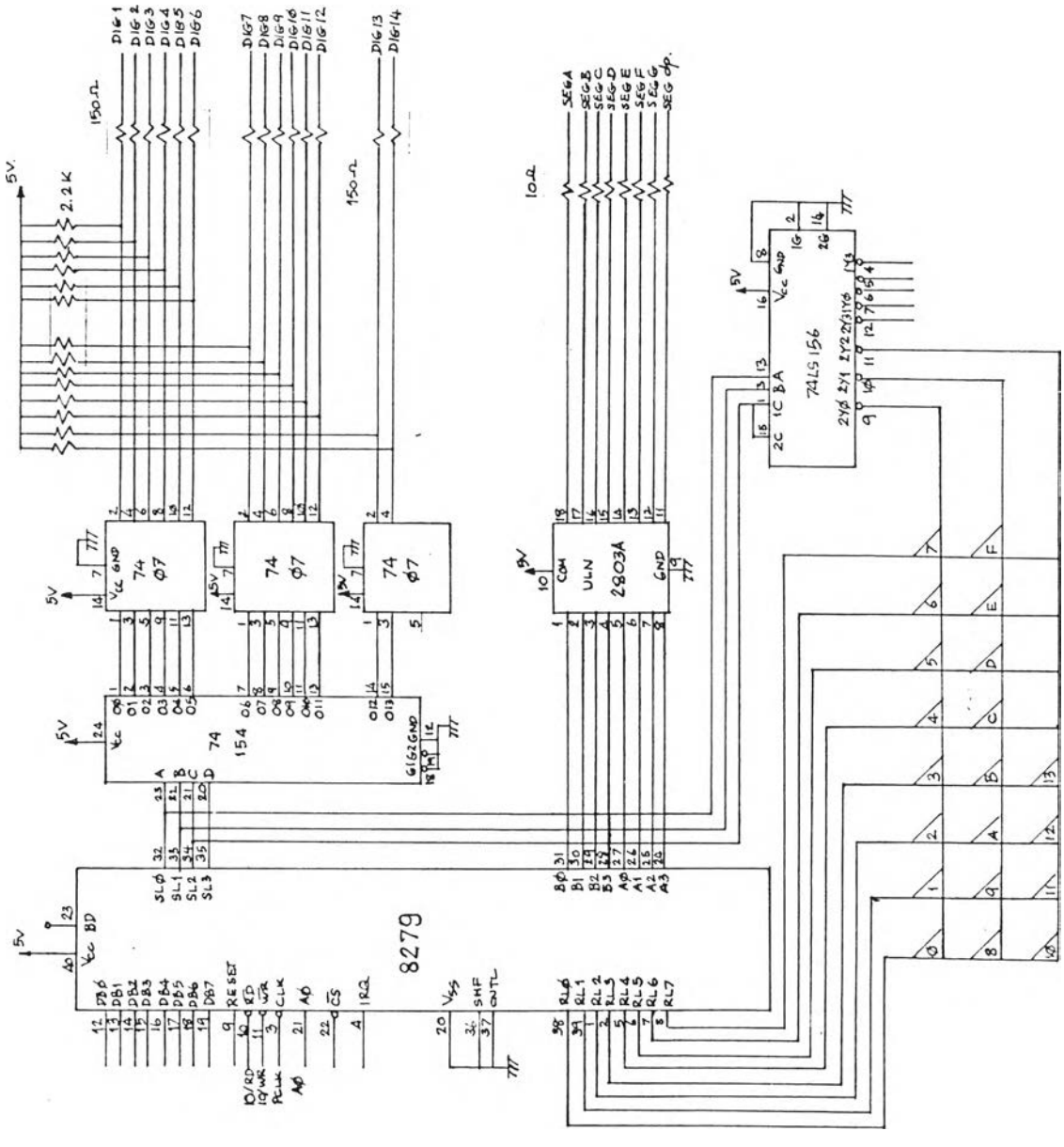
ค.2 วงจรส่วนหน่วยความจำ

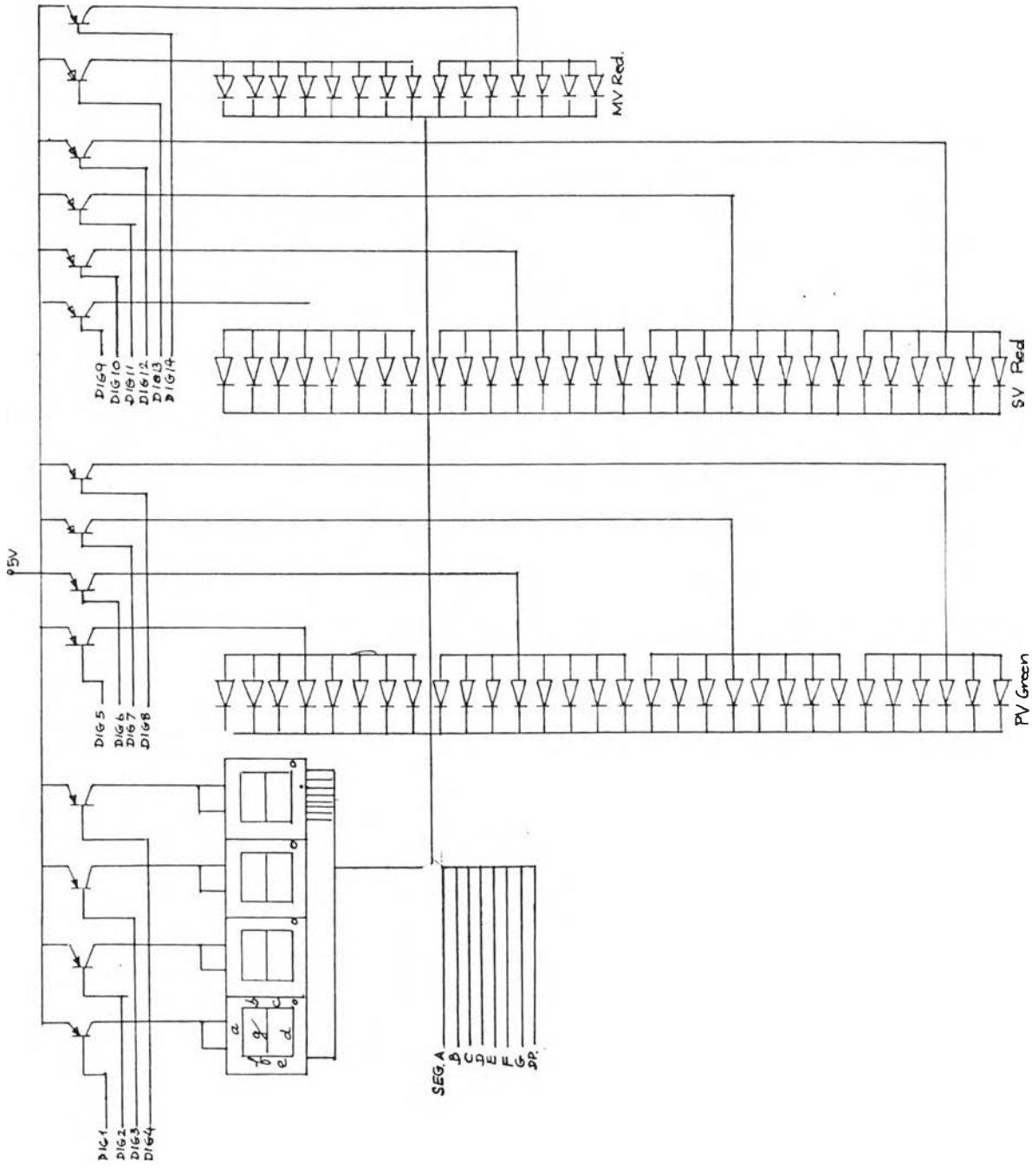


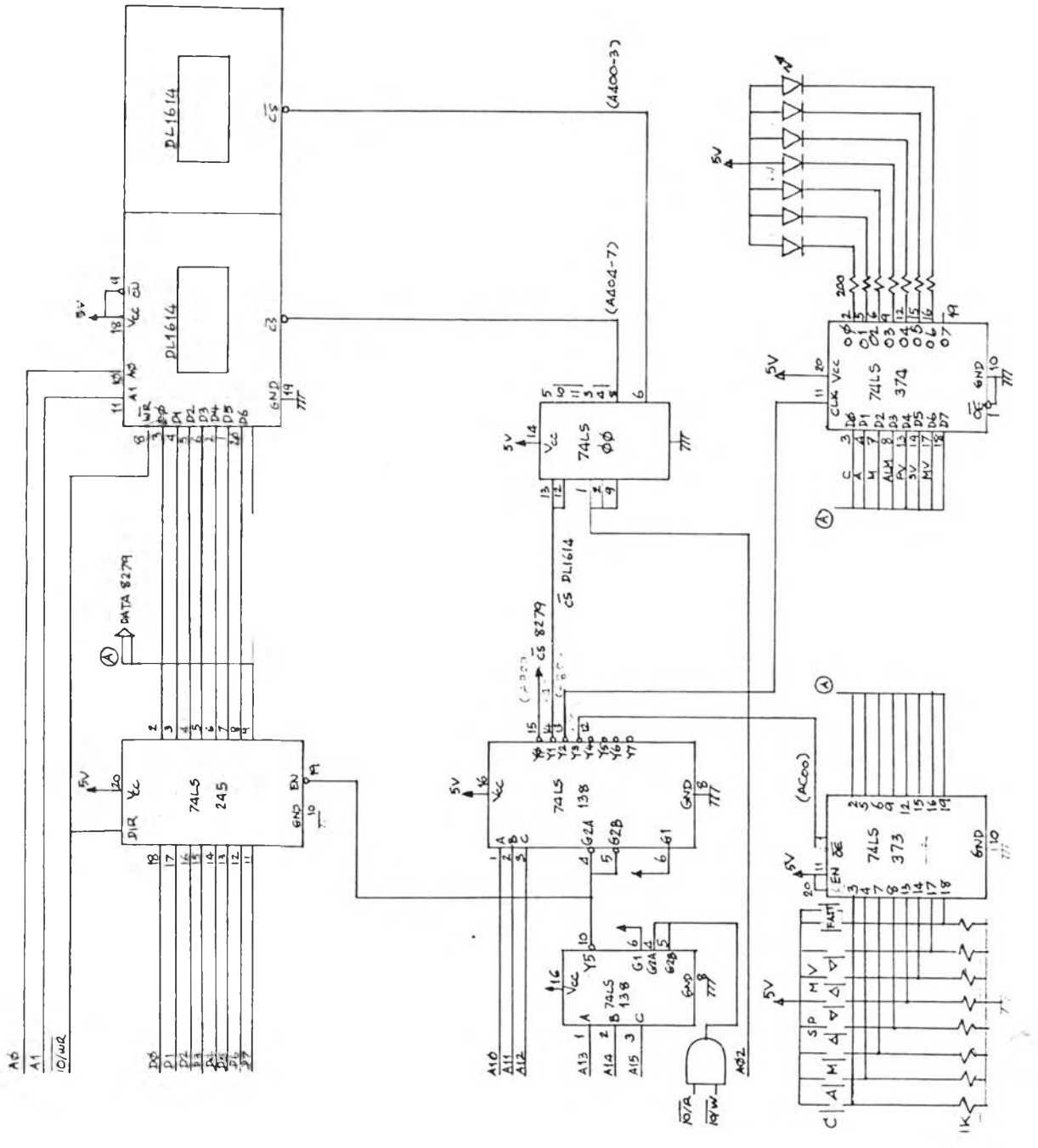
ค.3 วงจรส่วนตั้งเวลาและสื่อสาร



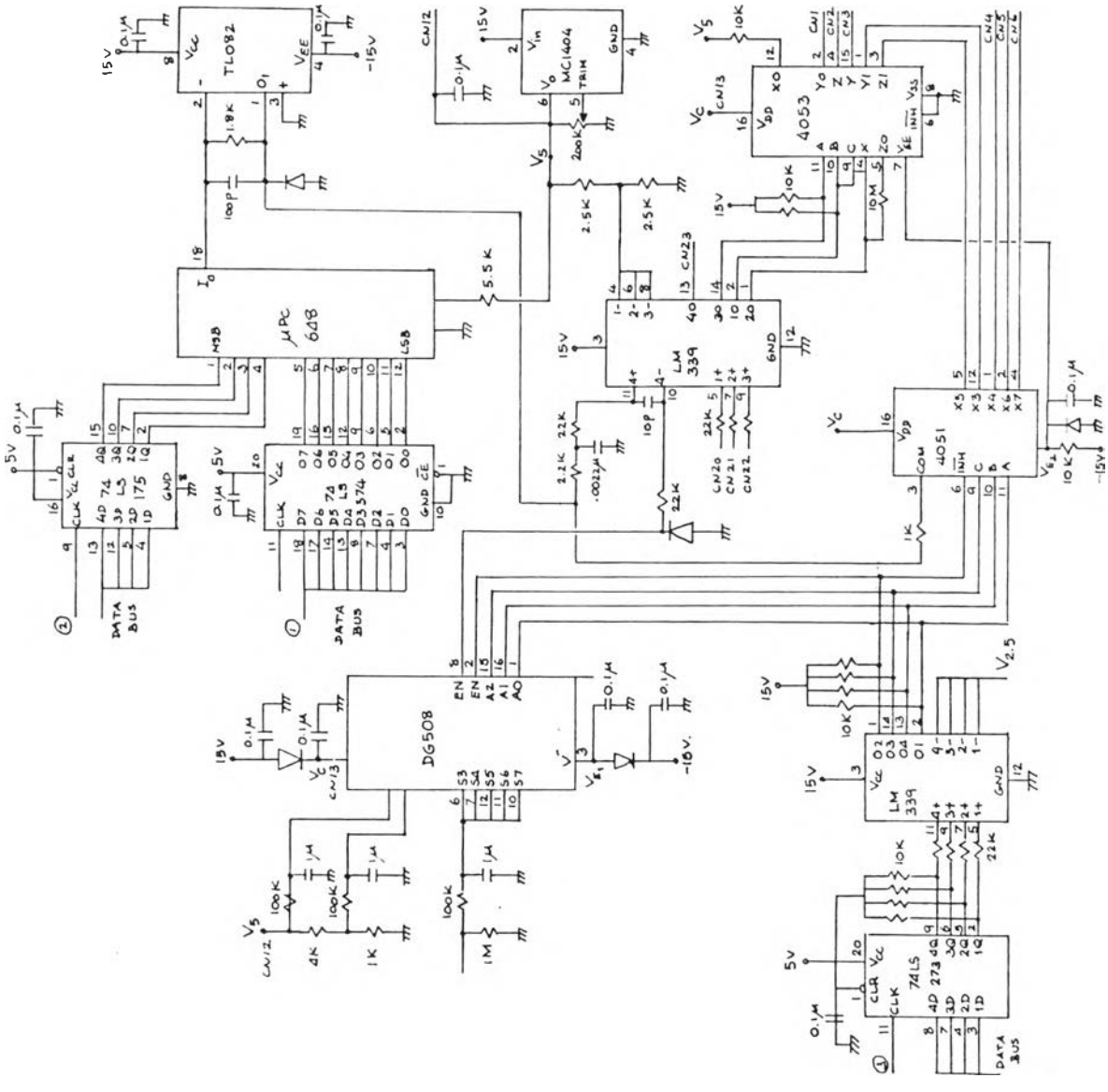
ค.4 วงจรส่วนแสดงผลและแป้นพิมพ์

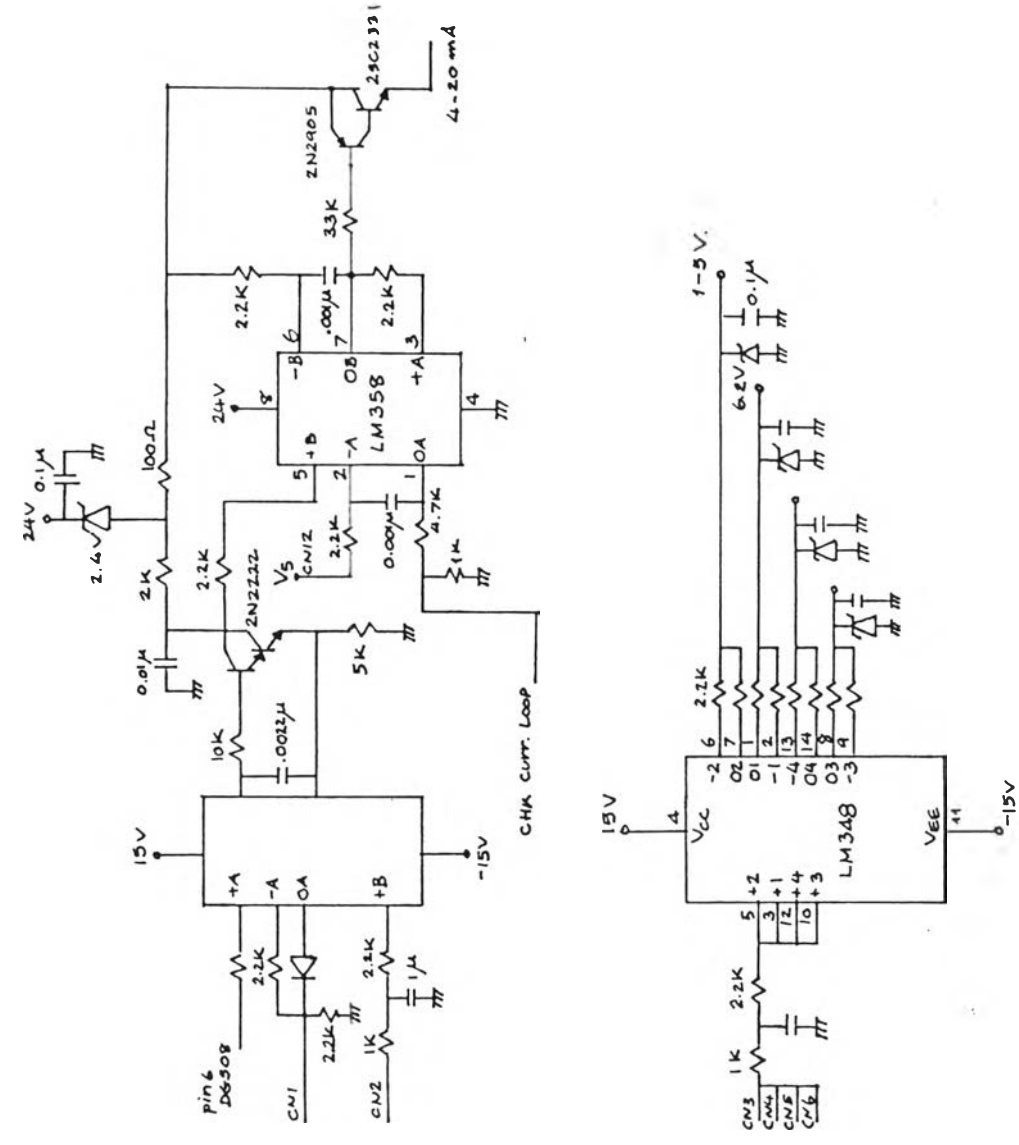


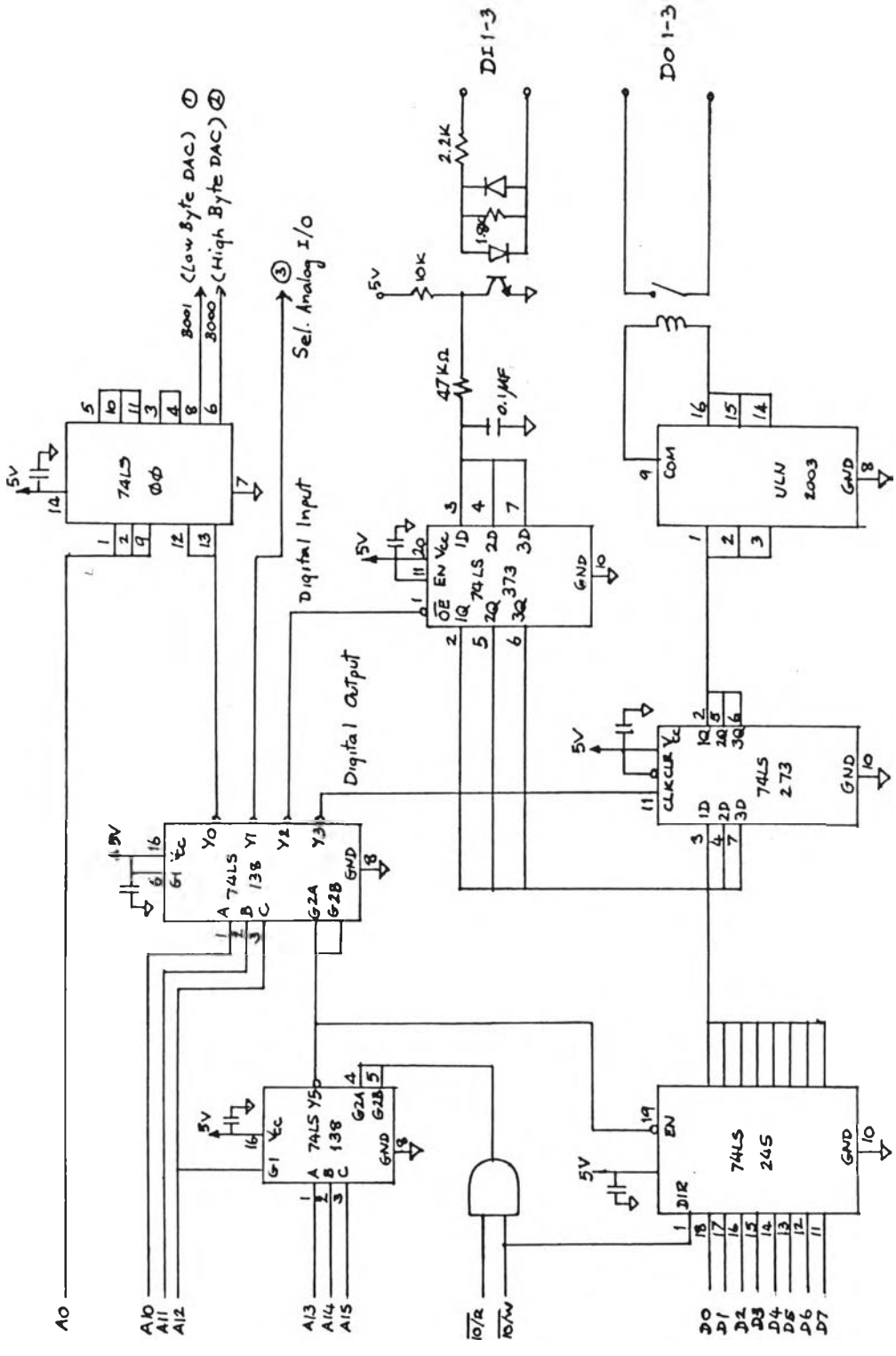

MV Red.
SV Red
PV Green
+5V
DIG9
DIG10
DIG11
DIG12
DIG13
DIG14
DIG5
DIG6
DIG7
DIG8
DIG1
DIG2
DIG3
DIG4
SEG A
B
C
D
E
F
G
DP
a
b
c
d
e
f
g


DL1614
DL1614
DATA 8279
74LS 245
74LS 138
74LS 138
74LS 00
74LS 374
74LS 373
(A400-3)
(A404-7)
(AC00)
CS 8279
CS DL1614
A0
A1
IO/M
RD
WR
A10
A11
A12
A13
A14
A15
A02
IO/M
RD/WR
5V
GND

ค.5 วงจรส่วนอินพุทและเอาท์พุท




24V
0.1μ
2.4V
100Ω
2K
2.2K
0.01μ
2N2222
10K
.002μ
5K
15V
-15V
+A
-A
OA
+B
Pin 6 DB308
2.2K
2.2K
CN1
2.2K
1K
CN2
2.2K
1μ
LM358
-B
OB
+A
-A
OA
+B
24V
8
4
5
2
1
6
7
3
2.2K
.001μ
2.2K
2N2905
2SC2331
33K
4-20 mA
V5
CN12
2.2K
0.001μ
4.7K
1K
CHK CURR. LOOP
1-5 V.
0.1μ
6.2V
2.2K
LM348
Vcc
VEE
-2
O2
O1
-1
-4
O4
O3
-3
+2
+1
+4
+3
15V
-15V
1K
2.2K
CN3
CN4
CN5
CN6


A0
A10
A11
A12
A13
A14
A15
IO/R
R/W
D0
D1
D2
D3
D4
D5
D6
D7
74LS 245
74LS 138
74LS 273
74LS 373
ULN 2003
74LS 00
5V
GND
2.2K
10K
47KΩ
0.1MF
DI1-3
DO1-3
Digital Input
Digital Output
Sel. Analog I/O
(Low Byte DAC)
(High Byte DAC)
8001
8000

ภาคผนวก ง. วิธีการโปรแกรมกำหนดรูปแบบการควบคุมลง ROM

เนื่องจากตัวควบคุมเชิงเลขชนิดโปรแกรมได้ ที่สร้างขึ้นยังไม่มี Compiler บนไมโครคอมพิวเตอร์ที่ทำหน้าที่เปลี่ยนรหัส mnemonic เป็นไฟล์ Intel Hex เพื่อเขียนลง EPROM โดยใช้ EPROM Programmer CLK3000 บนไมโครคอมพิวเตอร์

ดังนั้นจึงจำเป็นต้องทราบวิธีการเปลี่ยนรหัส mnemonic เป็นไฟล์ Intel Hex โดยวิธีการเขียนโปรแกรมภาษาแอสแซมบลี้ ซึ่งเป็นวิธีที่ใช้ในการเขียนโปรแกรมทดสอบตัวควบคุมที่สร้างขึ้น ขั้นตอนทั้งหมดมีรายละเอียดดังนี้

ง.1 เขียนรหัส mnemonic รายละเอียดดูในบทที่ 5

ง.2 เขียนแอสแซมบลี้ จากรหัส mnemonic นำมาเขียนเป็นภาษาแอสแซมบลี้ได้ โดยสามารแบ่งประเภทของฟังก์ชันออกเป็น 4 ประเภท ได้ดังนี้

ง.2.1 **ฟังก์ชันที่ไม่มีการส่งผ่านพารามิเตอร์** สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ง.1

Mnemonic	Assembly	Mnemonic	Assembly
ADD	ld bx,07fdah call bx	SUB	ld bx,0800dh call bx
MUL	ld bx,08040h call bx	DIV	ld bx,08073h call bx
ABS	ld bx,080beh call bx	SQR	ld bx,080a6h call bx
AND	ld bx,081cah call bx	OR	ld bx,081ech call bx
NOT	ld bx,0820eh call bx	CMP	ld bx,08220h call bx
HSL	ld bx,080d0h call bx	LSL	ld bx,0814dh call bx
FX	ld bx,07937h call bx		
BPID	ld bx,082cah call bx	CPID	ld bx,08550h call bx

รูปที่ ง.1 แสดงการแปลงฟังก์ชันที่ไม่มีการส่งผ่านพารามิเตอร์เป็นแอสแซมบลี้

ง.2.2 **ฟังก์ชันที่มีการส่งผ่านค่าตัวแปร** สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ง.2

Mnemonic	Assembly	Mnemonic	Assembly

LD DIn	mov al,n-1
	push ax
	mov bx,08272h
	call bx
		LD DOn	mov al,n-1
			push ax
			mov bx,082aeh
			call bx

LD FLn	mov al,n-1
	push ax
	mov bx,08290h
	call bx		

LD Xn	mov al,n-1
	push ax
	mov bx,0785ah
	call bx
		ST Yn	mov al,n-1
			push ax
			mov bx,078f0h
			call bx

**** n = channel no. & flag no.

รูปที่ ง.2 แสดงการแปลงฟังก์ชันที่มีการส่งผ่านค่าตัวแปรเป็นแอสแซมบลี้

ค่า n ในรูปที่ ง.2 คือค่าของช่องสัญญาณของอินพุทเอาท์พุททั้งอนาลอคและดิจิตอล หรือ ค่าของสถานะ flag ของฟังก์ชัน PID โดยที่ค่า n ในรหัส mnemonic จะต้องถูกลบด้วย 1 ในแอสแซมบลี้ เช่น ต้องการอ่านค่าอนาลอคอินพุทของช่องสัญญาณที่ 1 เขียนคำสั่งได้ดังนี้

```
mnemonic :      LD   X1
assembly :      mov  al,0
                push ax
                mov  bx,0785ah
                call bx
```

ง.2.3 ฟังก์ชันที่มีการส่งผ่านค่าตำแหน่ง แบ่งเป็น 2 ประเภท คือ

1. ส่งผ่านค่าตำแหน่งหนึ่งค่า สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ง.3

Mnemonic	Assembly	Mnemonic	Assembly

LD Pn	mov ax,addr
	push ax
	mov bx,07884h
	call bx
		ST Pn	mov ax,addr
			push ax
			mov bx,078b2h
			call bx

LEDn	mov ax,addr
	push ax
	mov bx,07b04h
	call bx
		LAGn	mov ax,addr
			push ax
			mov bx,07a0eh
			call bx

TIMn	mov ax,addr
	push ax
	mov bx,07f76h
	call bx
		GIF	mov ax,addr
			push ax
			mov bx,08bceh
			call bx

รูปที่ ง.3 แสดงการแปลงฟังก์ชันที่มีการส่งผ่านค่าตำแหน่งหนึ่งค่าเป็นแอสแซมบลี้

ค่า addr ในรูปที่ ง.3 คือค่าตำแหน่งของตัวแปรที่ใช้กับฟังก์ชัน โดยค่าตำแหน่งของแต่ละตัวแปรแสดงได้ดังรูปที่ ง.4

Variable	Address	Variable	Address
P1	018h	P2	01eh
P3	021h	P4	024h
P5	027h	P6	02ah
P7	02eh	P8	030h
P9	033h	P10	036h
P11	039h	P12	03ch
P13	03fh	P14	042h
P15	045h		
LED1	0182h	LED2	018fh
LAG1	019ch	LAG2	01a3h
LAG3	01aah	LAG4	01b1h
TIM1	02a2h	TIM2	02a6h
TIM3	02aah	TIM4	02aeh

รูปที่ ง.4 แสดงตำแหน่งของตัวแปรที่ใช้ในการส่งผ่านค่าตำแหน่งหนึ่งค่า

ตัวอย่างเช่น ต้องการอ่านค่า P1 เขียนคำสั่งดังนี้

```
mnemonic :   LD   P1
assembly :   mov  ax,01bh
             push ax
             mov  bx,07884h
             call bx
```

ตัวอย่าง ต้องการเรียกฟังก์ชัน LAG ตัวที่ 2

```
mnemonic :   LAG2
assembly :   mov   ax,01a3h
             push  ax
             mov   bx,07a0eh
             call  bx
```

2. ส่งผ่านค่าตำแหน่งสองค่า สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ง.5

```
-----------------------------------
  Mnemonic        Assembly
-----------------------------------
   DEDn         mov  ax,addr1
                push ax
                mov  ax,addr2
                push ax
                mov  bx,07c6dh
                call bx

-----------------------------------
```

รูปที่ ง.5 แสดงการแปลงฟังก์ชันที่มีการส่งผ่านค่าตำแหน่งสองค่าเป็นแอสแซมบลี้

ค่าตำแหน่งของ addr1 และ addr2 ของฟังก์ชันแสดงได้ดังรูปที่ ง.6

Variable	Address1	Address2
DED1	01b8h	01cah
DED2	0206h	0218h
DED3	0254h	02a6h

รูปที่ ง.6 แสดงตำแหน่งของตัวแปรที่ใช้กับฟังก์ชัน dead time

ตัวอย่าง การเรียกใช้ฟังก์ชัน dead time ที่ 1 เขียนคำสั่งได้ดังนี้

```
mnemonic :   DED1
assembly :   mov  ax,01b8h
             push ax
             mov  ax,01cah
             push ax
             mov  bx,07c6dh
             call bx
```

เมื่อได้ assembly แล้วนำมาเขียนโปรแกรม ซึ่งรูปร่างของโปรแกรมแสดงได้ดังรูปที่ ง.7

```
NAME  userlrom                    ---------------------
ASSUME CS:CODE,DS:DATA
DATA   SEGMENT WORD PUBLIC 'DATA'
DATA   ENDS                          โปรแกรมส่วนนี้
CODE   SEGMENT WORD PUBLIC 'CODE'    คงที่
userp  proc  far
       push  ax
       push  bx                      ---------------------
       mov   al,0h
       push  ax
       mov   bx,0785ah               โปรแกรมส่วนนี้ได้จาก
       call  bx  ; LD X1             การแปลง mnemonic เป็น
        .............                แอสแซมบลี้
        .............
       mov   bx,082cah
       call  bx  ; BPID              ---------------------
       pop   bx
       pop   ax                      โปรแกรมส่วนนี้
       ret                           คงที่
usep   ENDP
CODE   ENDS
       END                           ---------------------
```

รูปที่ ง.7 รูปร่างของโปรแกรม assembly ที่ใช้สร้าง Intel Hex

ง.3 Compile & Link ทำการ compile โดยใช้ compiler ASM86 และ link โดยใช้ iAPX 86,88 Utility มีขั้นตอนดังนี้

1. >ASM86 file.asm
2. >LINK86 file.obj to file.lnk
3. >LOC86 file.lnk to file.loc AD(SM(CODE(08000H)))
4. >OH86 file.loc to file.h

เมื่อหมดขั้นตอนนี้จะได้ file.h ซึ่งเป็นฟอร์แมทของ Intel hex 16 บิท

ง.4 แปลง Intel hex 16 บิท เป็น 8 บิท เนื่องจาก EPROM Programming ทำงานกับไฟล์ที่มีฟอร์แมทเป็น Intel hex 8 บิท ทำการแปลงโดยใช้โปรแกรม INTELH ซึ่งเขียนขึ้นเองดังนี้

```
>INTELH <ret>
>input file:file.h
>output file:file.hex
```

ง.5 เขียน EPROM โดยใช้ EPROM Programmer CLK3000 เขียน file.hex บน EPROM นำไปใส่ในบอร์ดหน่วยความจำตำแหน่ง 08000H เพื่อกำหนดรูปแบบการควบคุมให้กับตัวควบคุมเชิงเลข

ประวัติผู้เขียน

นายอำนวย แสงวิโรจนพัฒน์ เกิดเมื่อวันที่ 2 มกราคม พ.ศ. 2504 ที่กรุงเทพ จบการศึกษาชั้นปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า จากคณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ เมื่อปี พ.ศ. 2525 ปัจจุบันทำงานในตำแหน่งวิศวกรระดับ 5 อยู่ที่กองศูนย์ประมวลผลด้วยคอมพิวเตอร์ ฝ่ายวางแผนระบบไฟฟ้า การไฟฟ้าฝ่ายผลิตแห่งประเทศไทย