รายการอ้างอิง

ภาษาไทย

นิรันดร์ เลิศนิมิตรธรรม. 2537. การประยุกต์ใช้นิวรอลเน็ตเวอร์กสำหรับการตรวจรู้ก๊าซ. วิทยานิพนธ์ปริญญามหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย.

ภาษาอังกฤษ

Floyd, Micheal. 1996. Developing Visual Basic 4 Communications Applications. United States of America : The Coriolis Group, Inc.

Aitken, Peter. 1995. Visual Basic 4 Programing Explorer. United States of America : The Coriolis Group, Inc.

Driscoll, Frederick F. 1992. Data Communications. United States of America : Saunders College Publishing

Texas Instruments Incorporated. 1990. TMS320C5x User's Guide. Digital Signal Processing Products Texas Instruments Incorporated.

Texas Instruments Incorporated. 1994. TMS320C5x DSP Starter Kit User's Guide. Digital Signal Processing Products Texas Instruments Incorporated.

Figaro Engineering Incorporated. 1993. Figaro Gas Sensor. Figaro Engineering Incorporated

ภาคผนวก

ภาคผนวก ก

รายละเอียดของก๊าซพาห์ โซเลนอยด์วาล์วและก๊าซเซนเซอร์

ข้อมูลแสดงรายละเอียดของก๊าซพาห์แสดงได้ในตาราง ก.1 ข้อมูลของโซเลนอยด์วาล์วแสดงได้ในตาราง ก.2 ส่วนรายละเอียดของก๊าซเซนเซอร์แสดงได้ในตาราง ก.3

ตาราง ก.1 ข้อมูลของก๊าซออกซิเจนและก๊าซไนโตรเจนที่ใช้เป็นก๊าซพาห์

ก๊าซ	ความบริสุทธิ์	ก๊าซเจือปน	บริษัทผู้ผลิต
ออกซิเจน	99.5%	ความชื้น < 100ppm	ไทยอินดัสเตรียลก๊าซ
ไนโตรเจน	99.5%	ความชื้น < 100ppm ออกซิเจน < 0.3%	ไทยอินดัสเตรียลก๊าซ

ตาราง ก.2 ข้อมูลของโซเลนอยด์วาล์ว

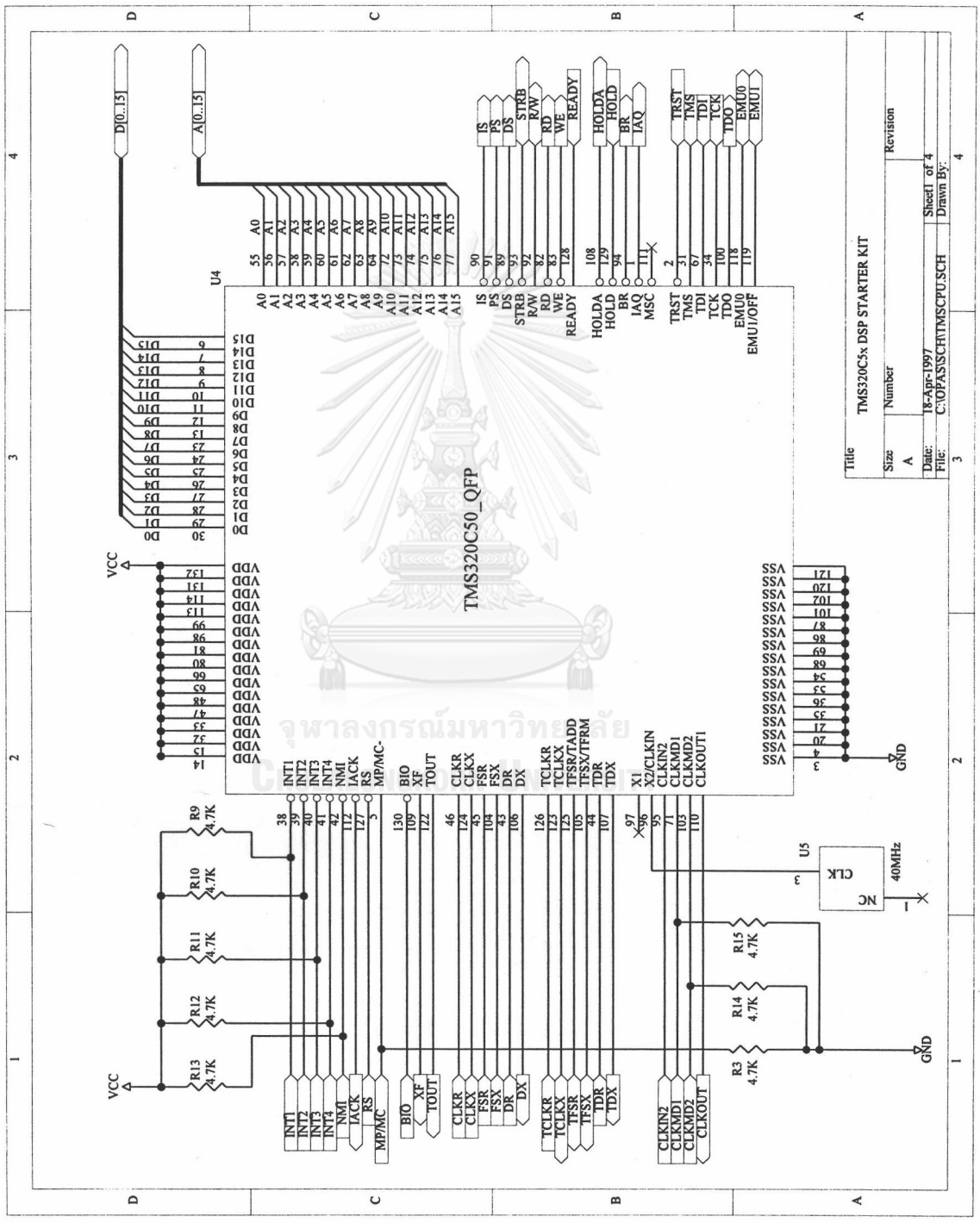
รายละเอียด	คุณสมบัติ
แหล่งจ่ายไฟ	AC 220V
สถานะภาพขณะไม่มีไฟเลี้ยง	ปิด
สารส่งผ่าน	อากาศ,น้ำ,น้ำมัน (ต่ำกว่า 50 cst)
ความดันในการทำงาน kgf/cm^2 (kPa)	0 ถึง 15 (0 ถึง 1500)
ความดันต้านทาน kgf/cm^2 (kPa)	30 (3000)
อุณหภูมิของเหลว	-10 ถึง 40 องศาเซลเซียส
อุณหภูมิสภาพแวดล้อม	-20 ถึง 50 องศาเซลเซียส
วัตถุดิบที่ใช้ทำโซเลนอยด์วาล์ว	อัลลูมิเนียม
ขนาดของท่อเชื่อมต่อ	PT(1/8)

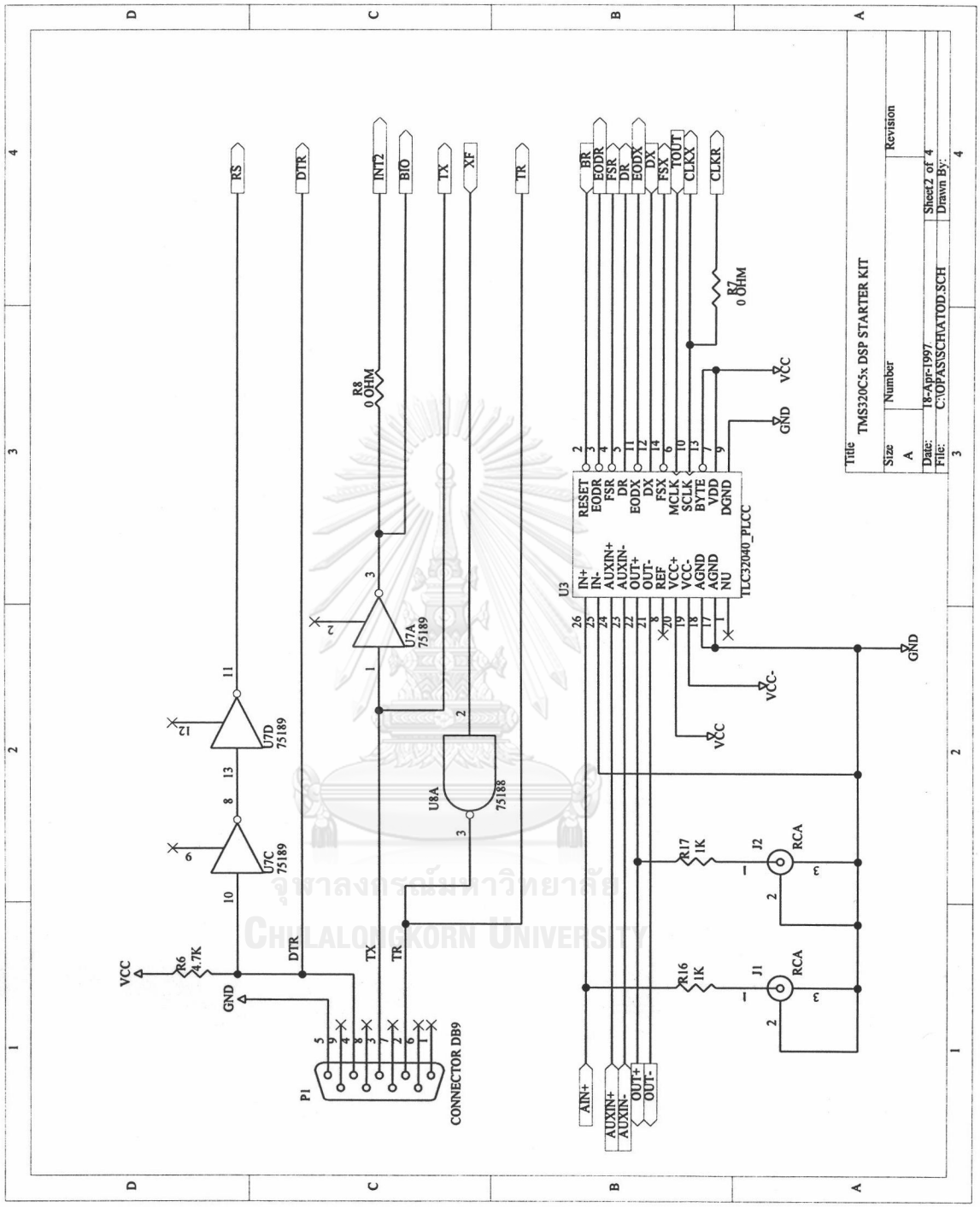
ตาราง ก.3 ข้อมูลของหัวตรวจวัดก๊าซทั้งสาม

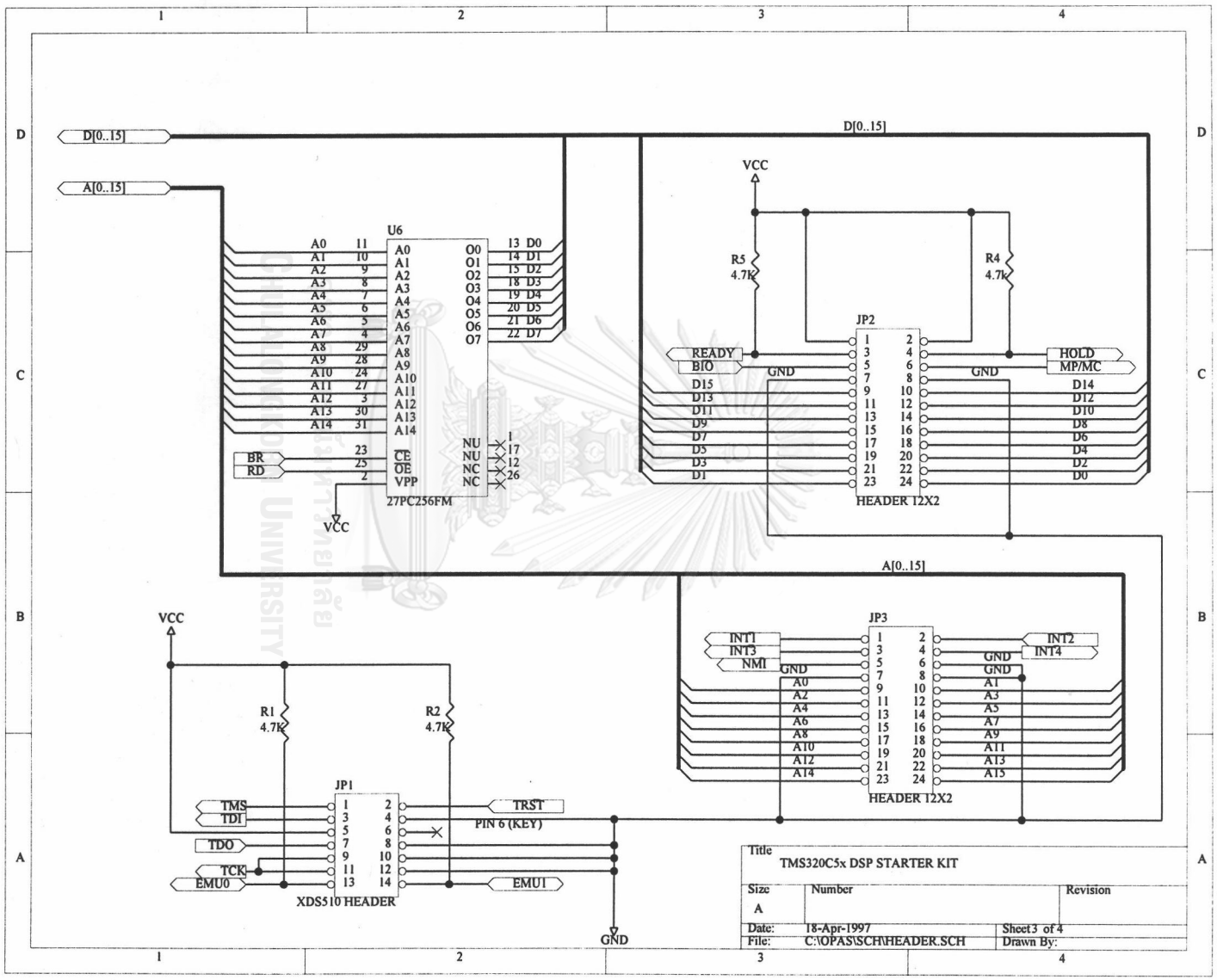
หัวตรวจวัดก๊าซ / รายละเอียด	TGS-800	TGS-813	TGS-822
แรงดันสูงสุดที่สามารถให้แก่วงจรไฟฟ้า	24 VAC หรือ DC		
แรงดันของขดลวดความร้อน	5.0 VAC หรือ DC		
กำลังไฟสูงสุด (P_S)	15 mW		
ผลตอบสนองไวต่อ	สิ่งเจือปนในอากาศ เช่น ควันบุหรี่, ไอระเหย เป็นต้น	ก๊าซติดไฟทั่วๆ ไป	สารละลายอินทรีย์เช่น แอลกอฮอล์, โทลูอีน, ไซลีน เป็นต้น

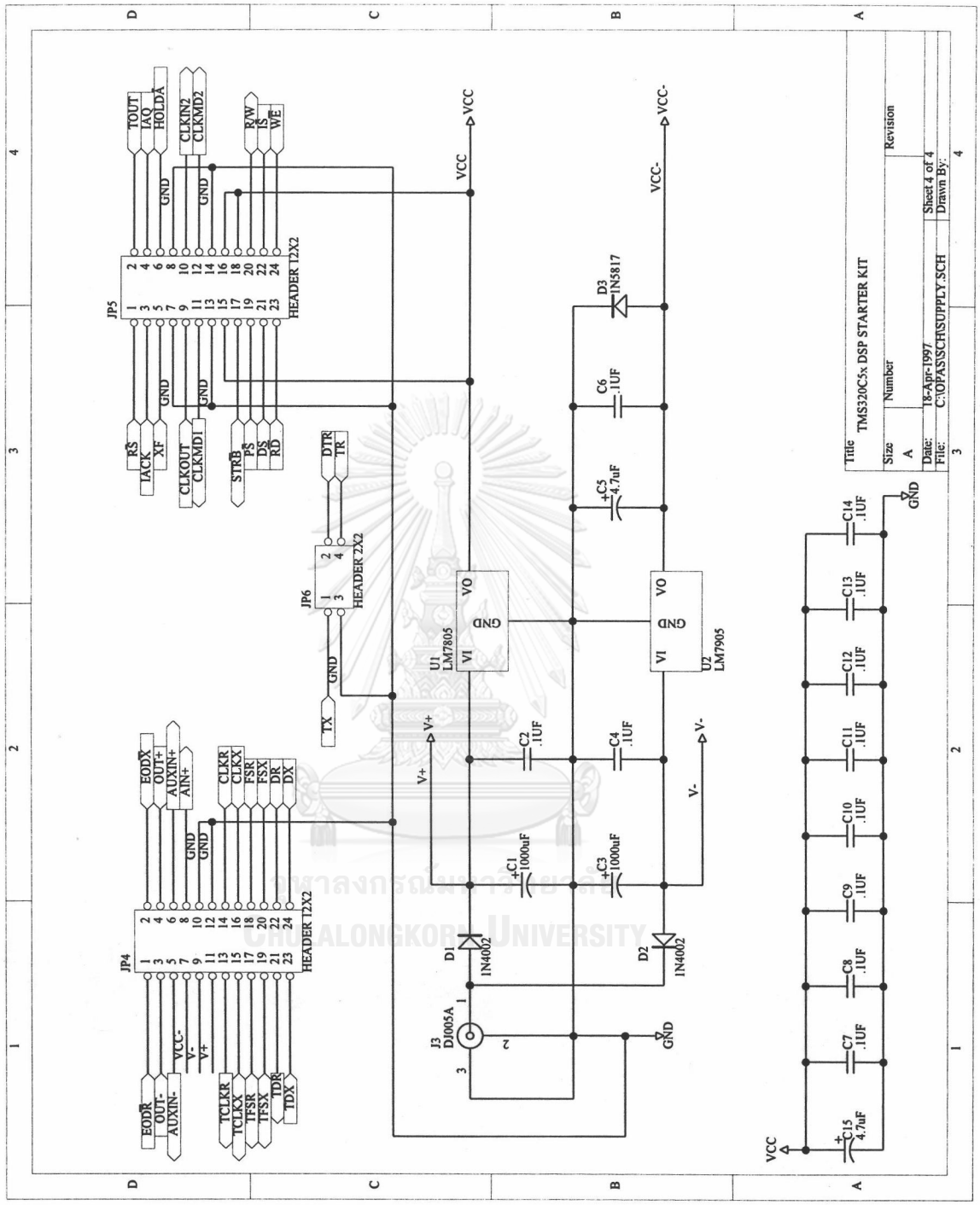
ภาคผนวก ข

วงจรของชุด TMS320C5x DSP Starter Kit



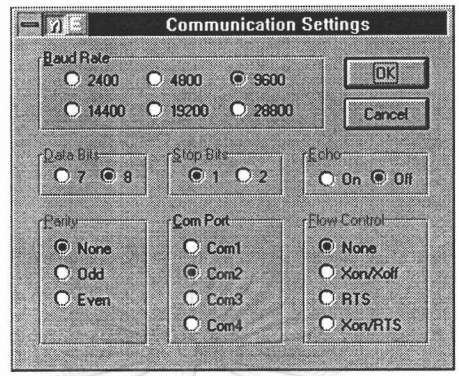



D[0..15]
A[0..15]
U6
27PC256FM
BR
RD
VCC
R5
4.7k
R4
4.7k
JP2
READY
BIO
GND
HOLD
MP/MC
HEADER 12X2
R1
4.7k
R2
4.7k
JP1
TMS
TDI
TDO
TCK
EMU0
TRST
PIN 6 (KEY)
EMU1
XDS510 HEADER
JP3
INT1
INT3
NMI
INT2
INT4
HEADER 12X2
Title
TMS320C5x DSP STARTER KIT
Size
A
Number
Revision
Date: 18-Apr-1997
Sheet3 of 4
File: C:\OPAS\SCH\HEADER.SCH
Drawn By:

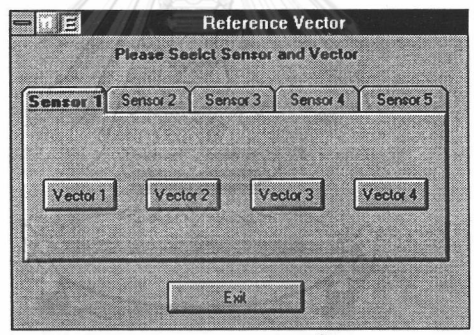

JP4
HEADER T2X2
JP5
HEADER T2X2
JP6
HEADER 2X2
U1
LM7805
U2
LM7905
D1
1N4002
D2
1N4002
D3
1N5817
J3
DJ005A
C1
1000uF
C2
.1UF
C3
1000uF
C4
.1UF
C5
4.7uF
C6
.1UF
C7
.1UF
C8
.1UF
C9
.1UF
C10
.1UF
C11
.1UF
C12
.1UF
C13
.1UF
C14
.1UF
C15
4.7uF
VCC
VCC-
GND
V+
V-
TMS320C5x DSP STARTER KIT
Size: A
Number
Revision
Date: 18-Apr-1997
Sheet 4 of 4
File: C:\DPASS\CH5\SUPPLY.SCH
Drawn By:

ภาคผนวก ค

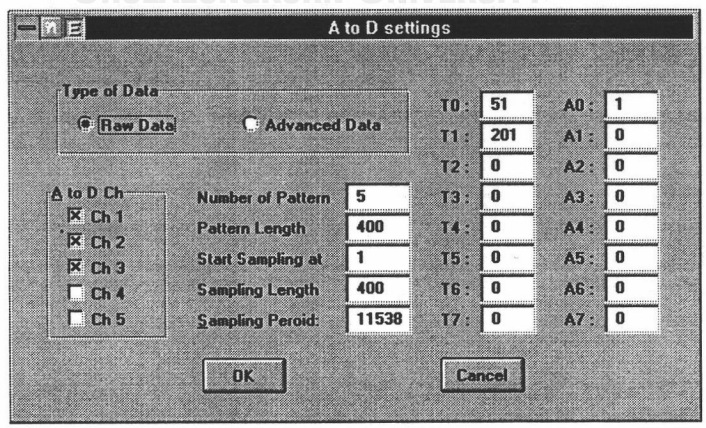
ตัวอย่างหน้าจอแสดงการทำงานของเครื่องคอมพิวเตอร์แม่



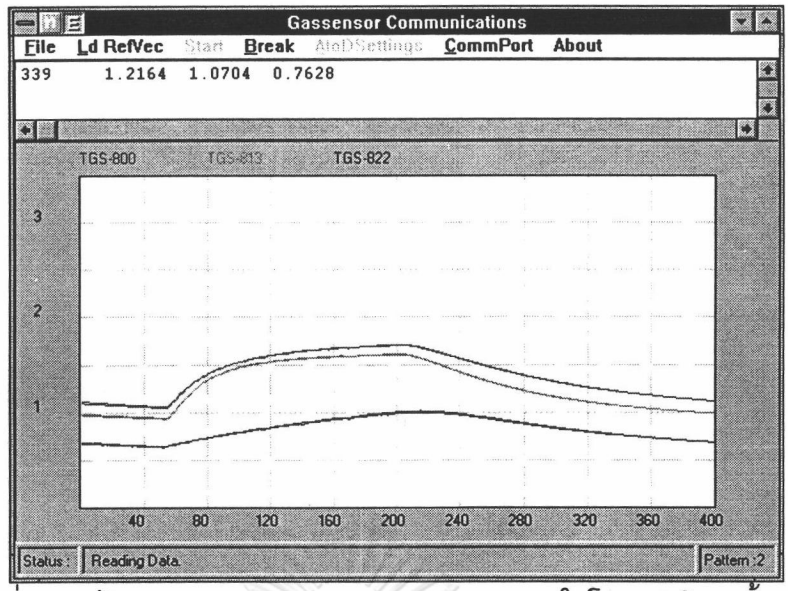
รูปที่ ค.1 หน้าจอแสดงการทำงานกำหนดเลือกพอร์ตอนุกรมที่ใช้



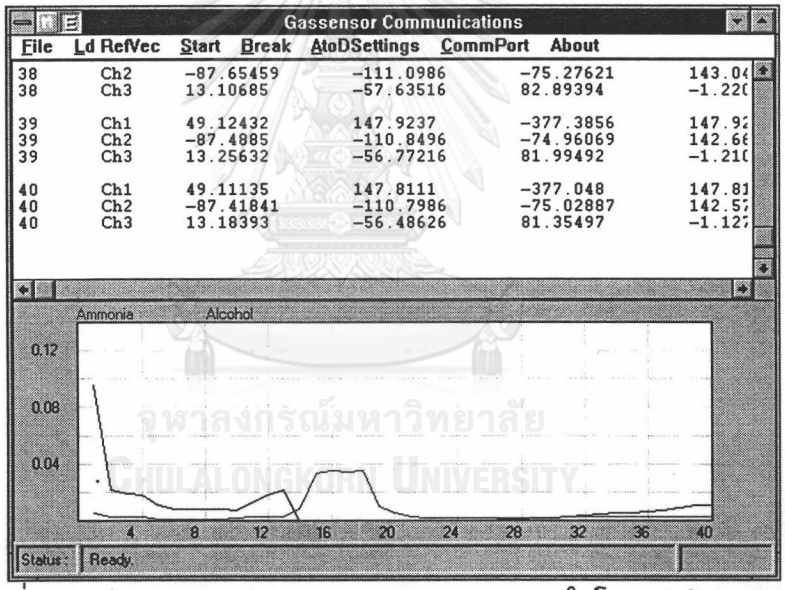
รูปที่ ค.2 หน้าจอแสดงการทำงานเลือกโหลดค่าเวกเตอร์อ้างอิง



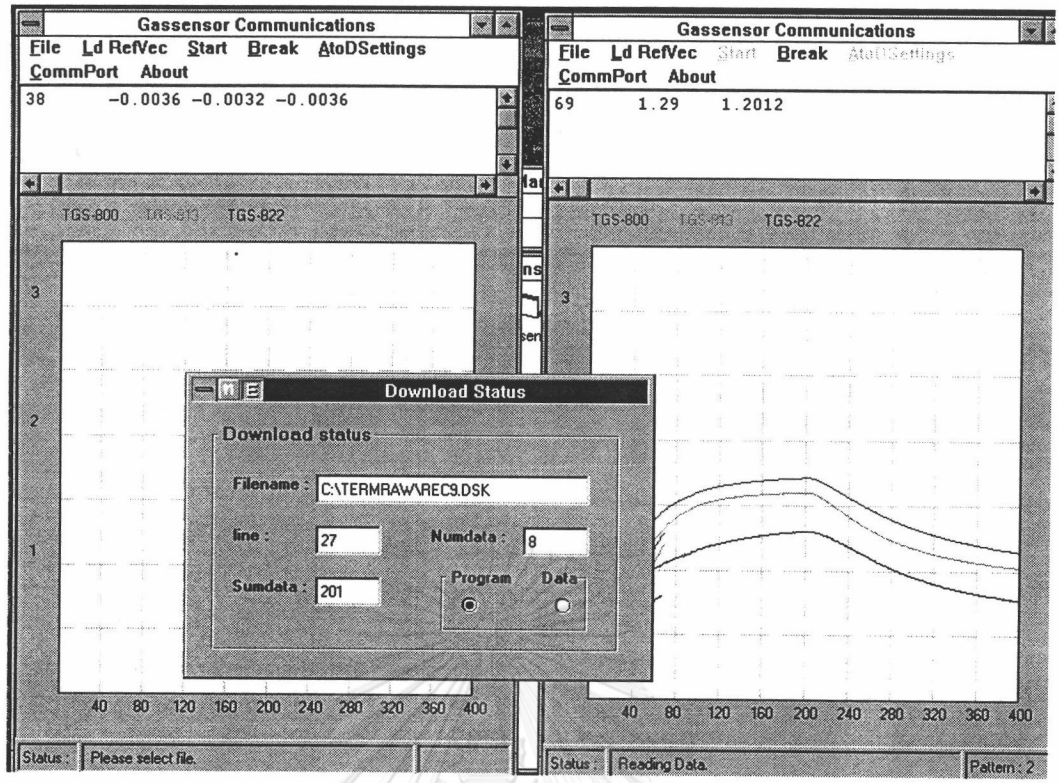
รูปที่ ค.3 หน้าจอแสดงการกำหนดค่าพารามิเตอร์การทำงานของ remote module



รูปที่ ค.4 หน้าจอแสดงการทำงานของ remote module ในโหมดส่งข้อมูลพื้นฐาน



รูปที่ ค.5 หน้าจอแสดงการทำงานของ remote module ในโหมดส่งข้อมูลพิเศษ



รูปที่ ค.6 หน้าจอแสดงการทำงานสองโปรแกรมพร้อมกัน

ภาคผนวก ง

โปรแกรมของ Matlab (M file) ที่ใช้ในการหาเมตริกซ์ปรับเทียบ

ในภาคผนวกนี้จะแสดงรายละเอียดโปรแกรมของ Matlab ที่เขียนขึ้นมาเพื่ออ่านค่าข้อมูลที่ได้จาก remote module และโปรแกรมที่ใช้ในการสร้างเมตริกซ์ปรับเทียบ

ไฟล์ dis2000.m

```
% DIS2000.m Program for read Gas sensor data (new version).
% 3 Sensors 5 pattern each of 400 data only.
% Display all data to 2000 samples
file=input('What file do you want to plot ?','s')

% clear
fid=fopen(file,'r');

for I=1:5,
  n=fscanf(fid,'%s',1);
  name1=fscanf(fid,'%s',1);
  name2=fscanf(fid,'%s',1);
  name3=fscanf(fid,'%s',1);
  for J=1:400
       x(J) = fscanf(fid,'%d',1);
       y1(J,I) = fscanf(fid,'%f',1);
       y2(J,I) = fscanf(fid,'%f',1);
       y3(J,I) = fscanf(fid,'%f',1);
  end
end

fclose(fid);

x=1:2000;
hold on;
plot(x(1:400),y1(:,1),'r');
plot(x(401:800),y1(:,2),'r');
plot(x(801:1200),y1(:,3),'r');
plot(x(1201:1600),y1(:,4),'r');
plot(x(1601:2000),y1(:,5),'r');

plot(x(1:400),y2(:,1),'g');
plot(x(401:800),y2(:,2),'g');
plot(x(801:1200),y2(:,3),'g');
plot(x(1201:1600),y2(:,4),'g');
plot(x(1601:2000),y2(:,5),'g');

plot(x(1:400),y3(:,1),'b');
plot(x(401:800),y3(:,2),'b');
plot(x(801:1200),y3(:,3),'b');
plot(x(1201:1600),y3(:,4),'b');
plot(x(1601:2000),y3(:,5),'b');
axis([0 2000 0 3]);

grid on
```

ไฟล์ genall.m

```
% Genall.m  m file for generate data of ammonia and alcohol
% Test at -3, -5, -7, -9

for j=3:2:9,
lzero=-j;
load am01-2                                              % ammonia 0.01%
am01_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
am01_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
am01_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
am01_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
am01_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[am01_1 am01_2 am01_4 am01_5];                        % training
valid=[am01_3];                                          % validation
test=[am01_5];                                           % test
temp=[log10(0.01) lzero]';
cc=[temp temp temp temp];                                % concentration
v=[temp];                                                % valid concen

load am05-2
am05_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
am05_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
am05_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
am05_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
am05_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr am05_1 am05_2 am05_4 am05_5];
valid=[valid am05_3];
test=[test am05_5];
temp=[log10(0.05) lzero]';                               % ammonia 0.05%
cc=[cc temp temp temp temp];
v=[v temp];

load am1-2
am1_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
am1_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
am1_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
am1_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
am1_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr am1_1 am1_2 am1_4 am1_5];
valid=[valid am1_3];
test=[test am1_5];
temp=[log10(0.1) lzero]';                                % ammonia 0.1%
cc=[cc temp temp temp temp];
v=[v temp];

load al01-2
al01_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
al01_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
al01_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
al01_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
al01_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr al01_1 al01_2 al01_4 al01_5];
valid=[valid al01_3];
test=[test al01_5];
temp=[lzero log10(0.01)]';                               % alcohol 0.01%
cc=[cc temp temp temp temp];
v=[v temp];

load al05-2
al05_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
al05_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
al05_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
al05_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
al05_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr al05_1 al05_2 al05_4 al05_5];
valid=[valid al05_3];
test=[test al05_5];
```

```
temp=[lzero log10(0.05)]';                          % alcohol 0.05%
cc=[cc temp temp temp];
v=[v temp];

load al1-2
al1_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
al1_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
al1_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
al1_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
al1_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr al1_1 al1_2 al1_4 al1_5];
valid=[valid al1_3];
test=[test al1_5];
temp=[lzero log10(0.1) ]';                          % alcohol 0.1%
cc=[cc temp temp temp temp];
v=[v temp];

load water-2
water_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
water_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
water_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
water_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
water_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr water_1 water_2 water_4 water_5];
valid=[valid water_3];
test=[test water_5];
temp=[lzero lzero]';
cc=[cc temp temp temp temp];
v=[v temp];

training=tr;
clear tr;
validation=valid;
clear valid;
c=cc;
clear cc;
clear temp;
cv=v;
clear v;

qcal=qmatrix(training,c);
[vc,vl]=pca(training);

for i=1:28,
   fcal=pcrcal(training,c,vc,i);
   result=fcal*validation;
   dif=(10.^result)-(10.^cv);
   dif2=dif.*dif;
   sse(j,i)=sum(sum(dif2));
end

end
```

ไฟล์ genall3.m

```
% Genall3.m  m file for generate data of ammonia and alcohol

lzero=-3;
load am01-2                                          % ammonia 0.01%
am01_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
am01_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
am01_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
am01_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
am01_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[am01_1 am01_2 am01_4 am01_5];                    % training
valid=[am01_3];                                      % validation
test=[am01_5];                                       % test
temp=[log10(0.01) lzero]';
```

```
cc=[temp temp temp temp];                                    % concentration
v=[temp];                                                    % valid concen

load am05-2
am05_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
am05_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
am05_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
am05_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
am05_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr am05_1 am05_2 am05_4 am05_5];
valid=[valid am05_3];
test=[test am05_5];
temp=[log10(0.05) lzero]';                                   % ammonia 0.05%
cc=[cc temp temp temp temp];
v=[v temp];

load am1-2
am1_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
am1_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
am1_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
am1_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
am1_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr am1_1 am1_2 am1_4 am1_5];
valid=[valid am1_3];
test=[test am1_5];
temp=[log10(0.1) lzero]';                                    % ammonia 0.1%
cc=[cc temp temp temp temp];
v=[v temp];

load al01-2
al01_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
al01_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
al01_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
al01_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
al01_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr al01_1 al01_2 al01_4 al01_5];
valid=[valid al01_3];
test=[test al01_5];
temp=[lzero log10(0.01)]';                                   % alcohol 0.01%
cc=[cc temp temp temp temp];
v=[v temp];

load al05-2
al05_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
al05_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
al05_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
al05_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
al05_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr al05_1 al05_2 al05_4 al05_5];
valid=[valid al05_3];
test=[test al05_5];
temp=[lzero log10(0.05)]';                                   % alcohol 0.05%
cc=[cc temp temp temp temp];
v=[v temp];

load al1-2
al1_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
al1_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
al1_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
al1_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
al1_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr al1_1 al1_2 al1_4 al1_5];
valid=[valid al1_3];
test=[test al1_5];
temp=[lzero log10(0.1) ]';                                   % alcohol 0.1%
cc=[cc temp temp temp temp];
v=[v temp];
```

```
load water-2
water_1=[y1(45:300,1); y2(45:300,1); y3(45:300,1);];
water_2=[y1(45:300,2); y2(45:300,2); y3(45:300,2);];
water_3=[y1(45:300,3); y2(45:300,3); y3(45:300,3);];
water_4=[y1(45:300,4); y2(45:300,4); y3(45:300,4);];
water_5=[y1(45:300,5); y2(45:300,5); y3(45:300,5);];
tr=[tr water_1 water_2 water_4 water_5];
valid=[valid water_3];
test=[test water_5];
temp=[lzero lzero]';
cc=[cc temp temp temp temp];
v=[v temp];

training=tr;
clear tr;
validation=valid;
clear valid;
c=cc;
clear cc;
clear temp;
cv=v;
clear v;

qcal=qmatrix(training,c);
[vc,vl]=pca(training);

for i=1:28,
   fcal=pcrcal(training,c,vc,i);
   result=fcal*validation;
   dif=(10.^result)-(10.^cv);
   dif2=dif.*dif;
   sse3(i)=sum(sum(dif2));
end
```

ภาคผนวก จ

โปรแกรมในส่วนของ remote module

ในภาคผนวกนี้จะแสดงรายละเอียดโปรแกรมในส่วนของ remote module ตัวโปรแกรมจะเขียนขึ้นโดยใช้ภาษาแอสเซมบลีของ TMS320C50

ไฟล์ rec9.asm

```
;************************************************************************
;       File:           REC9.ASM
;       Date:           Nov 14, 1996
;       Author:         Opas Siricunchittawon
;
;       Purpose:        Test routine transmit by used interrupt from A/D
;                       New version from sendnew in A/D interrupt at 19200bps
;                       ADD test routine for receive from DSK's RS232
;                       ADD circular buffer for TX buffer
;
;       COPYRIGHT (C) 1996 COMMUNICATION LAB.
;                          Chulalongkorn University Thailand
;       History         Nov 12, 1996 copy from rec1.asm
;                                    Modify function ATOI of dl6.asm
;         Nov 14, 1996 merge dl1.asm for sampling 5 sensor
;         Jan 21, 1997 Change I/O for new I/O (574)
;         Jan 23, 1997 Correct bug at circular buffer
;         Jan 24, 1997 Add check CAN,RESET
;         Jan 28, 1997 Save data to Data Memory
;         Jan 31, 1997 change lacc sacl -> SPLK, Add command T0-T7,A0-A7
;         Feb  4, 1997 change sendword to send high byte before low byte
;         Feb  6, 1997 Add command S11-S54, Optimize call reset_buf B main
;         Feb 10, 1997 Remove SF,SP,CS
;                      Add PL (pattern Length), SL (SAmpling length),SS
;------------------------------------------------------------------------
        .mmregs
*
;-------------------------------------------------------------;
; Bit definition of the control register in the AIC           ;
;-------------------------------------------------------------;

      ;+-------------------------+
      ;|LP xx G1 G0 | SY AX LB BP|              G1  G0  gain
      ;+------------+------------+             ---------------
      ; |     GAIN    |  |  |  +-- BP Filter     0   0  4
      ; |     Synch --+  |  +----- Loopback      1   1  4
      ; |     Auxin -----+                       1   0  2
      ; + (sinx)/x filter                       0   1  1

*
*****************************************************************
*  Register Used table
*  ar0  OutBuf_head
*  ar1  OutBuf_Tail
*  ar2  Address data of sensor1
*  ar3  Address data of sensor2
*  ar4  Address data of sensor3
*  ar5  Address data of sensor4
*  ar6  Address data of sensor5
```

```
*  ar7  Auxilary register
*

DigIO           .set    0e000h          ; Port A (8 bits)     Digital I/O
PortB           .set    0d000h          ; Port B (8 bits)     Mux
OutBuffer       .set    0f00h           ; Output Buffer start address
BufferLength    .set    0ffh            ; Buffer length= 15 word (or byte)

Start_add       .set    0980h           ; Start addess of linear buffer
ACK             .set    0006h           ; Acknowledge code
CAN             .set    0018h           ; Cancel code
RESET           .set    0012h           ; DC2 code used as reset DSK module

;  Start address of each sensor
st_ads1         .set    01000h          ; start addres of buffer of sensor1
st_ads2         .set    01100h          ; start addres of buffer of sensor2
st_ads3         .set    01200h          ; start addres of buffer of sensor3
st_ads4         .set    01300h          ; start addres of buffer of sensor4
st_ads5         .set    01400h          ; start addres of buffer of sensor5

StartT0         .set    0e00h           ; start address of T0
StartA0         .set    0e08h           ; start address of A0

Sen1Vec1        .set    01500h          ; Start address of reference vector1
Sen1Vec2        .set    01600h
Sen1Vec3        .set    01700h
Sen1Vec4        .set    01800h

Sen2Vec1        .set    01900h          ; Start address of reference vector1
Sen2Vec2        .set    01a00h
Sen2Vec3        .set    01b00h
Sen2Vec4        .set    01c00h

Sen3Vec1        .set    01d00h          ; Start address of reference vector1
Sen3Vec2        .set    01e00h
Sen3Vec3        .set    01f00h
Sen3Vec4        .set    02000h

Sen4Vec1        .set    02100h          ; Start address of reference vector1
Sen4Vec2        .set    02200h
Sen4Vec3        .set    02300h
Sen4Vec4        .set    02400h

Sen5Vec1        .set    02500h          ; Start address of reference vector1
Sen5Vec2        .set    02600h
Sen5Vec3        .set    02700h
Sen5Vec4        .set    02800h

                .ds     0e00h           ; Data variable
; GLOBAL Variable
T0              .word   1               ; time 0 at
T1              .word   2
T2              .word   3
T3              .word   4
T4              .word   5
T5              .word   6
T6              .word   7
T7              .word   8

A0              .word   000h        ; action of time0 is
A1              .word   0ffh
A2              .word   055h
A3              .word   0aah
A4              .word   000h
A5              .word   0ffh
A6              .word   055h
A7              .word   0aah
```

```
TA        .word    13              ; Fcut = 19230.76 , Ts = 52us
RA        .word    13              ; Fcut = 19230.76 , Ts = 52us
TAp       .word    31              ;
RAp       .word    31              ;
TB        .word    20              ; Fs = 2*Fcut
RB        .word    20              ; Fs = 2*Fcut
AIC_CTR   .word    18h             ; No use Band Pass Filter

Buffer          .word   0
Numbit          .word   0
Data            .word   0
Counter         .word   0
Rx232Flag       .word   0
RxBit           .word   0
RxData          .word   0
RxFlag          .word   0
TEMP            .word   0
Temp            .word   0
CancFlag        .word   0
count           .word   0
data            .word   0
OldVect         .word   0935h           ; Old Address of ISR2

Sample          .word   400             ; Number of sample old 400
Pattern         .word   1
Sensor          .word   3               ; Pattern of sensors (00111b)
Sam_pe          .word   11538           ; 30 for 10 Min., 60 for 20 Min.
PatternLength   .word   256
StartSampling   .word   45

data_ad         .word   0
SamNumber       .word   0               ; counter for sampling number
Temp_pat        .word   0

StartAdd        .word   0
Length          .word   255
AdvData         .word   8
LowWord         .word   0
HighWord        .word   0

*****************************************************************
*   Set up the ISR vector                                       *
*****************************************************************
        .ps     080ah
rint:   B       RECEIVE         ;0A; Serial prot receive interrupt RINT.
xint:   B       TRANSMIT        ;0C; Serial port transmit interrupt XINT.
        ;-------------------------------------------------------
        .ps     0804h
int2:   B       Receive232

*
*****************************************************************
*  Interrupt Marsk Register
*  8      7   6     5    4       3     2   1    0
* INT4  TXNT TRNT XINT RINT    TINT INT3 INT2 INT1
*****************************************************************

        .ps 0a00h
        .entry
START:  SETC    INTM            ; Disable interrupts
        LDP     #0              ; Set data page pointer
        OPL     #0834h,PMST
        LACC    #0
        SAMM    CWSR            ; Set software wait state to 0
        SAMM    PDWSR           ;

* Reset AIC by writing to PA2 (address >52) on EVM
```

```
        SPLK    #022h,IMR       ; Using XINT syn TX & RX
        CALL    AICINIT         ; initialize AIC and enable interrupts
*
*****************************************************************
* This routine enables serial port rx interrupts & configures   *
* TLC32046 due to the frame sync. When RINT generate read a dummy *
* data from the AIC then generate a sine wave to send out.      *
*****************************************************************
;
        CLRC    OVM             ; OVM = 0
        SPM     0               ; PM = 0
        SPLK    #012h,IMR
        CLRC    INTM            ; enable

*
*****************************************************************
* Initialize routine                                            *
* This routine for initialization all output Port               *
*****************************************************************

       ZAP                              ; clear ACC and P register
       sacl    ar7
       OUT     ar7,DigIO
       OUT     ar7,PortB

       SETC    xf                       ; Set RS232 Tx to high
       SETC    SXM

       LAR     ar0,#OutBuffer
       MAR     *,ar0
       LAR     ar1,#OutBuffer
       MAR     *,ar1

ResBufMain:
       LAR     ar7,#Start_add  ; start address of buffer at 0980h
       MAR     *,ar7
MAIN:
       NOP                     ; wait for a transmit or receive
       LDP     #RxFlag
       LACC    RxFlag
       BCND    MAIN,EQ         ; If RxFlag =0 goto MAIN

       SPLK    #0,RxFlag       ; clear RxFlag after detect

       LAR     ar7,#Start_add  ; reset start address of buffer to 0980h
       MAR     *,ar7

       LACC    *+              ; load data at buffer to ACC
       SACL    Temp            ; save to temp
       SUB     #67             ; check for "C"
       BCND    C,EQ            ; If buffer(1) = 'C' goto C
       LACC    Temp            ;
       SUB     #83             ; check for "S"
       BCND    S,EQ            ; If buffer(1) = 'S' goto S
       LACC    Temp            ;
       SUB     #84             ; check for "T"
       BCND    T,EQ            ; If buffer(1) = 'T' goto T
       LACC    Temp
       SUB     #65
       BCND    A,EQ            ; If buffer(1) = 'A' goto A
       LACC    Temp
       SUB     #80
       BCND    P,EQ            ; If buffer(1) = 'P' goto P

       B       ResBufMain


P:     LACC    *+
```

```
        SUB     #76
        BCND    PL,EQ           ; If buffer(2) ='L' goto PL
    ; Error Trap
        B       ResBufMain

PL:     CALL    ATOI
        SACL    PatternLength
        CALL    SendWord
        B       ResBufMain


A:      LACC    *+
        SACL    Temp
        SUB     #86
        BCND    AV,EQ           ; If buffer(2) = 'V' goto AV

Action:                         ; Else check A0-A7
        LACC    Temp
        SUB     #48             ; sub '0'
        sacb                    ; save offset from acc -> accb
        sub     #7
        bcnd    EndA,GT         ; If buffer(2) > 7 goto EndA
        CALL    ATOI
        SACL    Temp            ; save data to Temp
        lacc    #StartA0
        ADDB
        SAMM    ar2
        mar     *,ar2
        lacc    Temp
        sacl    *,0,ar7
        CALL    SendWord
EndA:
        B       ResBufMain

AV:     CALL    ATOI
        SACL    AdvData
        call    SendWord
        B       ResBufMain

C:      LACC    *+
        SACL    Temp
        SUB     #88             ; check for "CX"
        bcnd    CX,EQ           ; If buffer(2) = 'X' goto CX

        LACC    Temp
        SUB     #80             ; check for "CP"
        bcnd    CP,EQ

        LACC    Temp
        SUB     #84             ; check for "CT"
        bcnd    CT,EQ
    ;error trap
        B       ResBufMain
CX:     call    ATOI
        SACL    Sensor
        call    SendWord
        B       ResBufMain
CP:     call    ATOI
        SACL    Sam_pe
        call    SendWord
        B       ResBufMain
CT:     call    ATOI
        SACL    Pattern
        call    SendWord
        B       ResBufMain

S:      LACC    *+
        SACL    Temp
```

```
        SUB     #65             ; check for "A"
        BCND    SA,EQ           ; if "A" then start AtoD
        LACC    Temp
        SUB     #83             ; check for "S"
        BCND    SS,EQ
        LACC    Temp
        SUB     #76             ; check for "L"
        BCND    SL,EQ
        lacc    Temp
        SUB     #53             ; - '5'
        BCND    LoadMem,LEQ     ; if buffer(2) < 5 goto LoadMem
    ;error trap
        B       ResBufMain

LoadMem:                        ; command S11-S14,....S51-S54
        LACC    Temp
        SUB     #49
        SACL    Temp
        LT      Temp
        MPY     #400h
        LTP     Temp            ; dummy load
        ADD     #Sen1Vec1
        sacl    StartAdd
        LACC    *+
        SUB     #49             ; - '1'
        sacl    Temp
        LT      Temp
        MPY     #100h
        LTP     Temp
        add     StartAdd
        SACL    StartAdd

        lar     ar5,StartAdd
        SPLK    #0,RxData
        call    ATOI
        sacl    Length
        call    SendWord
        lar     ar6,Length
;         mar     ar5
        call    DLD
        B       ResBufMain


SA:
        SPLK    #0,Temp_pat             ; Temp_pat =0
        call    start_ad                ; start A/D
        B       ResBufMain
SL:     call    ATOI
        SACL    Sample
        call    SendWord
        B       ResBufMain
SS:     CALL    ATOI
        SACL    StartSampling
        call    SendWord
        B       ResBufMain

T:      LACC    *+
        SACL    Temp            ; save to temp
        SUB     #83
        BCND    TS,EQ           ; If buffer[2] = 'TS' goto TS

        lacc    Temp
        SUB     #48             ; sub '0'
        sacb                    ; save offset from acc -> accb
        sub     #7
        bcnd    EndA,GT         ; If buffer < 7 goto EndA
        CALL    ATOI
        SACL    Temp            ; save data to Temp
```

```
        lacc    #StartT0
        ADDB
        SAMM    ar2
        mar     *,ar2
        lacc    Temp
        sacl    *,0,ar7
        CALL    SendWord
        B       ResBufMain

TS:     LACC    #4f4bh          ; 4f="O" 4b="K"
        call    SendWord        ; Send "OK"
        B       ResBufMain

;------- end of main program ----------;
*********************************************************************
* Function : DLD  Download data memory from PC to DSK
* Input    : ar5 = Start Address, ar6 = length
* Output   : Data Memory at StartAdd
*********************************************************************
DLD:
        call    ReadWord
        MAR     *,ar5
        sacl    *+,0,ar6
        BANZ    DLD,*-
        SPLK    #0,RxFlag
;        CALL    reset_buf
        ret

*********************************************************************
* Function : ATOI translate ASCII to Integer
* Input    : Buffer at ar7
* Output   : ACC
*********************************************************************
        .include        "atoi.inc"

*********************************************************************
*  Function reset_buf : Rewind buffer to 1000h
*********************************************************************
;reset_buf:
;        LAR     ar7,#Start_add      ; start address of buffer at 0980h
;        MAR     *,ar7
;        RET

*********************************************************************
* Function : ReadByte read data from RS232 Port (Polling Mode)
*            Return value at ACC
*********************************************************************
ReadByte:
        LDP     #RxData
        BIT     RxData,0                ; Check MSB of RxData
        BCND    ReadByte,NTC            ; If NOT(MSB) of Rxdata goto
        LACC    RxData
        and     #7fffh
        sacl    RxData
;        splk    #0,RxData               ; Clear RxData after used
        RET

*********************************************************************
* Function : ReadWord read data at RS232 Port
*            Return value at ACC
*********************************************************************
ReadWord:
        CALL    ReadByte                ; Read high byte
        sacl    Temp
        CALL    SendByte
        CALL    ReadByte                ; Read low byte
        add     Temp,8
        sacl    Temp
```

```
        call    SendByte
        LACC    Temp
        ret

*************************************************************************
* Function SendWord : Send word of data off ACC (send high byte first)
*                     In : ACC          Out : None
*                     Used ACC and ACCB, Modify ar0
*************************************************************************
SendWord:                       ; SEND word of data of ACC
        SACB                    ; save acc in accb
        RPT     #7
        SFR
        call    SendByte
        LACB                    ; (accb) -> acc
*************************************************************************
* Function : SendByte send byte in ACC
*
*************************************************************************
SendByte:
        MAR     *,ar0
        SACL    *+
        LAMM    ar0
        AND     #BufferLength
        OR      #OutBuffer
        SAMM    ar0
        RET

*************************************************************************
*  Function Send_ACK : Send Acknowledge code (06h)
*************************************************************************
;Send_ACK:
;        LACC    #ACK
;        call    SendByte
;        RET

*************************************************************************
* RECIEVER INTERRUPT SERVICE ROUTINE
*************************************************************************
*
RECEIVE:
        LAMM    DRR             ; read data from DRR
;         SAMM    DXR           ; send data to aic output (by DXR)
        SACL    data_ad         ; save data to variable

        LACC    Counter
        ADD     #1
        SACL    Counter         ; Update counter

        LACC    RxBit
        BCND    CheckTx,LEQ

        SUB     #15
        BCND    DecRxbit,GT

        Bit     RxBit,15        ; Test LSB of RxBit
        BCND    DecRxbit,NTC    ; If RxBit != even goto DecRxbit
        LACC    RxData          ; Else check bio pin and receive data
        SFR
        SACL    RxData

        BCND    DecRxbit,bio    ; If bio low goto DecRxbit
        ADD     #80h
        SACL    RxData

DecRxbit:
        LACC    RxBit
        SUB     #1
```

```
        SACL    RxBit
        BCND    CheckTx,GT      ; If not receive all bit goto CheckTx

        mar     *,ar7
        lacc    RxData
        OR      #08000h         ; set MSB to 1 (flag for receive all bit)
        SACL    RxData
        and     #0ffh
        sacl    *               ; Save data to Buffer (address at ar7)
        LAMM    ar7
        AND     #0ffh
        OR      #Start_add
        SAMM    ar7

** Check Buffer (check RESET, CAN, ENTER,
        SUB     #RESET
        BCND    CheckEnter,NEQ  ; If receive byte != RESET goto CheckEnter
        LDP     #data
        LACC    #805h
        TBLW    OldVect
        LACC    #800h
        BACC                            ; Branch to 800h mean reset DSK board

CheckEnter:
        LACC    *
        SUB     #13                     ; end code
        BCND    CheckCancel,NEQ         ; IF receive byte != Enter goto CkCan
        SPLK    #1,RxFlag               ; set RxFlag = 1

CheckCancel
        LACC    *+                      ; read and increment
        SUB     #CAN                    ; cancel code
        BCND    CheckTx,NEQ             ; If not cancel code goto CheckTx
;         LDP     #CancFlag
        SPLK    #1,CancFlag             ; set CancFlag =1

CheckTx:
        Bit     Counter,15
        BCND    End_receive,NTC

        LACC    Numbit
        BCND    CheckBuffer,LEQ         ; If Numbit <=0 goto CheckBuffer

        SUB     #11
        BCND    stopbit,NEQ
        CLRC    xf
        B       DecNumbit

stopbit:
        LACC    Numbit
        SUB     #2
        BCND    Txbit,GT
        SETC    xf
        B       DecNumbit

Txbit:
        LDP     #Buffer
        LACC    Buffer
        ROR
        SACL    Buffer
        bcnd    setxf,C
        clrc    xf
        B       DecNumbit
setxf:
        setc    xf


DecNumbit:
```

```
        LACC    Numbit
        SUB     #1
        SACL    Numbit

End_receive:
        RETE
CheckBuffer:                            ; Check ouput circulat buffer
        LAMM    ar0
        SACL    TEMP
        LAMM    ar1
        SUB     TEMP
        BCND    End_receive,EQ          ; If Outhead==OutTail goto End_receive
        MAR     *,ar1
        LACC    *+
        SACL    Buffer
        SPLK    #11,Numbit              ; load Numbit = 11 mean start send
RS232
        LAMM    ar1                     ;
        AND     #BufferLength
        OR      #OutBuffer              ;
        SAMM    ar1
        RETE

*
*********************************************************************
* TRANSMIT INTERRUPT SERVICE ROUTINE
*********************************************************************
*
TRANSMIT:
        RETE

*
*********************************************************************
* Receive DSK's RS232 INTERRUPT SERVICE ROUTINE
*********************************************************************
*
Receive232:

        LDP     #RxBit
        LACC    RxBit
        BCND    endrec,GT
        rpt     #260
        nop
        SPLK    #17,RxBit               ; Save RxBit = 17 mean start
        SPLK    #0,RxData               ; Clear RxData
endrec:
;       LAMM    IFR
;       SAMM    IFR
        RETE

*
*********************************************************************
*  DESCRIPTION: This routine initializes the TLC320C46 for       *
*               a 8Khz sample rate with a gain setting of 1      *
*********************************************************************
* aic initialization data
*

        .include        "aicinit.inc"

*********************************************************************
*  Function start_ad : Start of A/D
*********************************************************************
*
start_ad:                       ; program for sampling Gas Sensor

        LAR     ar2,#st_ads1
        LAR     ar3,#st_ads2
```

```
        LAR     ar4,#st_ads3
        LAR     ar5,#st_ads4
        LAR     ar6,#st_ads5

        LDP     #Sample         ; Initialize all parameter
        SPLK    #0,SamNumber    ; Clear counter of sample
        LACC    Temp_pat
        ADD     #1
        SACL    Temp_pat

LOOP_MAIN:                      ; count number of sample
        SPLK    #0h,Counter

        LDP     #CancFlag       ; check Cancal flag
        LACC    CancFlag
        BCND    AtoDcont,EQ     ; goto main if CancFlag=0

        SPLK    #0,CancFlag     ; Clear CancFlag after detect
        OUT     CancFlag,DigIO  ; Clear Digital I/O port
        RET                     ; Return from start_ad for cancal operation

AtoDcont:
;        LDP     #Temp_n
        LACC    SamNumber
        ADD     #1              ; Increment SamNumber
        SACL    SamNumber
************************   Check T0,T1,...T7
        SUB     T0
        BCND    Time1,NEQ
        OUT     A0,DigIO
Time1:  LACC    SamNumber
        SUB     T1
        BCND    Time2,NEQ
        OUT     A1,DigIO
Time2:  LACC    SamNumber
        SUB     T2
        BCND    Time3,NEQ
        OUT     A2,DigIO
Time3:  LACC    SamNumber
        SUB     T3
        BCND    Time4,NEQ
        OUT     A3,DigIO
Time4:  LACC    SamNumber
        SUB     T4
        BCND    Time5,NEQ
        OUT     A4,DigIO
Time5:  LACC    SamNumber
        SUB     T5
        BCND    Time6,NEQ
        OUT     A5,DigIO
Time6:  LACC    SamNumber
        SUB     T6
        BCND    Time7,NEQ
        OUT     A6,DigIO
Time7:  LACC    SamNumber
        SUB     T7
        BCND    start_sam,NEQ
        OUT     A7,DigIO

start_sam:
        LACC    SamNumber
        SUB     StartSampling
        BCND    LoopCounter,LT

        LACC    SamNumber
        SUB     StartSampling
        SUB     Sample
        BCND    LoopCounter,GEQ
```

```
Begin_ad:
        SPLK    #0,data
        OUT     data,PortB
        IDLE                        ; Wait for 1st sample

    ;-------------------------------------------------------
    ;   Subroutine for sensor No. 1
    ;-------------------------------------------------------

sensor1:
        SPLK    #01h,data
        OUT     data,PortB
        IDLE                        ; Wait for 1st sample
        ldp     #Sensor
        bit     Sensor,15           ; check for LSB of Sensor
        bcnd    sensor2,NTC         ; if TC=0 goto sensor2
        mar     *,ar2
        lacc    data_ad
        sacl    *+
        LACC    AdvData
        BCND    sensor2,GT          ; If AdvData goto sensor2
        lacc    data_ad
        CALL    SendWord            ; Send 1st sample

    ;-------------------------------------------------------
    ;   Subroutine for sensor No. 2
    ;-------------------------------------------------------

sensor2:
        SPLK    #02h,data
        OUT     data,PortB
        IDLE                        ; Wait for 2nd sample
;         ldp     #Sensor
        bit     Sensor,14           ; check for 2 of Sensor
        bcnd    sensor3,NTC         ; if TC=0 goto sensor2
        mar     *,ar3
        lacc    data_ad
        sacl    *+
        LACC    AdvData
        BCND    sensor3,GT          ; If AdvData goto sensor3
        lacc    data_ad
        CALL    SendWord            ; Send 2nd sample

    ;-------------------------------------------------------
    ;   Subroutine for sensor No. 3
    ;-------------------------------------------------------

sensor3:
        SPLK    #03h,data
        OUT     data,PortB
        IDLE                        ; Wait for 3rd sample
;         ldp     #Sensor
        bit     Sensor,13           ; check for LSB of Sensor
        bcnd    sensor4,NTC         ; if TC=0 goto sensor2
        mar     *,ar4
        lacc    data_ad
        sacl    *+
        LACC    AdvData
        BCND    sensor4,GT          ; If AdvData goto sensor4
        lacc    data_ad
        CALL    SendWord            ; Send 3rd sample

    ;-------------------------------------------------------
    ;   Subroutine for sensor No. 4
    ;-------------------------------------------------------

sensor4:
```

```
                SPLK    #04h,data
                OUT     data,PortB
                IDLE                    ; Wait for 4th sample
;                ldp     #Sensor
                bit     Sensor,12       ; check for LSB of Sensor
                bcnd    sensor5,NTC     ; if TC=0 goto sensor2
                mar     *,ar5
                lacc    data_ad
                sacl    *+
                LACC    AdvData
                BCND    sensor5,GT      ; If AdvData goto sensor5
                lacc    data_ad
                CALL    SendWord        ; Send 4th sample

            ;-------------------------------------------------------
            ;   Subroutine for sensor No. 5
            ;-------------------------------------------------------

        sensor5:
                SPLK    #05h,data
                OUT     data,PortB
                IDLE                                ; Wait for 1st sample
;                ldp     #Sensor
                bit     Sensor,11                   ; check for LSB of Sensor
                bcnd    LoopCounter,NTC             ; if TC=0 goto sensor2
                mar     *,ar6
                lacc    data_ad
                sacl    *+
                LACC    AdvData
                BCND    LoopCounter,GT              ; If AdvData goto LoopCounter
                lacc    data_ad
                CALL    SendWord                    ; Send 1st sample

        LoopCounter:
                LDP     #Sam_pe
                lacc    Counter
                SUB     Sam_pe
                bcnd    LoopCounter,LEQ             ; If Counter < Sam_pe goto LoopCounter

                LACC    SamNumber
                SUB     PatternLength
                BCND    LOOP_MAIN,NEQ               ; If SamNumber != PatternLength goto
LOOP_MAIN

                LACC    AdvData
                BCND    Ck_pat,EQ                   ; If not AdvData goto Ck_pat

        ********** start Dot product of all data
                SPM     3               ; shift P register 6 bit (/64)
        Sen1:
                ldp     #Sensor
                bit     Sensor,15       ; check for LSB of Sensor
                bcnd    Sen2,NTC        ; if TC=0 goto sensor2

                LAR     ar3,#st_ads1
                MAR     *,ar3
                LACC    #Sen1Vec1       ; Sensor1 Vector1
                SAMM    BMAR
                call    StartMAC

                LAR     ar3,#st_ads1
                MAR     *,ar3
                LACC    #Sen1Vec2       ; Sensor1 Vector2
                SAMM    BMAR
                call    StartMAC

                LAR     ar3,#st_ads1
                MAR     *,AR3
```

```
        LACC    #Sen1Vec3       ; Sensor1 Vector3
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads1
        mar     *,ar3
        LACC    #Sen1Vec4       ; Sensor1 Vector4
        SAMM    BMAR
        call    StartMAC

Sen2:
        bit     Sensor,14       ; check for LSB of Sensor
        bcnd    Sen3,NTC        ; if TC=0 goto sensor2

        LAR     AR3,#st_ads2
        MAR     *,AR3
        LACC    #Sen2Vec1       ; Sensor2 Vector1
        SAMM    BMAR
        call    StartMAC

        LAR     AR3,#st_ads2
        MAR     *,AR3
        LACC    #Sen2Vec2       ; Sensor2 Vector2
        SAMM    BMAR
        call    StartMAC

        LAR     AR3,#st_ads2
        MAR     *,AR3
        LACC    #Sen2Vec3       ; Sensor2 Vector3
        SAMM    BMAR
        call    StartMAC

        LAR     AR3,#st_ads2
        MAR     *,AR3
        LACC    #Sen2Vec4       ; Sensor2 Vector4
        SAMM    BMAR
        call    StartMAC

Sen3:
        bit     Sensor,13       ; check for LSB of Sensor
        bcnd    Sen4,NTC        ; if TC=0 goto sensor2

        LAR     ar3,#st_ads3
        MAR     *,ar3
        LACC    #Sen3Vec1       ; Sensor3 Vector1
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads3
        MAR     *,ar3
        LACC    #Sen3Vec2       ; Sensor3 Vector2
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads3
        MAR     *,ar3
        LACC    #Sen3Vec3       ; Sensor3 Vector3
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads3
        MAR     *,ar3
        LACC    #Sen3Vec4       ; Sensor3 Vector4
        SAMM    BMAR
        call    StartMAC

Sen4:
        bit     Sensor,12       ; check for LSB of Sensor
```

```
        bcnd    Sen5,NTC        ; if TC=0 goto sensor2

        LAR     ar3,#st_ads4
        MAR     *,ar3
        LACC    #Sen4Vec1       ; Sensor4 Vector1
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads4
        MAR     *,AR3
        LACC    #Sen4Vec2       ; Sensor4 Vector2
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads4
        MAR     *,AR3
        LACC    #Sen4Vec3       ; Sensor4 Vector3
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads4
        MAR     *,AR3
        LACC    #Sen4Vec4       ; Sensor4 Vector4
        SAMM    BMAR
        call    StartMAC

Sen5:
        bit     Sensor,11       ; check for LSB of Sensor
        bcnd    Ck_pat,NTC      ; if TC=0 goto sensor2

        LAR     ar3,#st_ads5
        MAR     *,ar3
        LACC    #Sen5Vec1       ; Sensor5 Vector1
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads5
        MAR     *,AR3
        LACC    #Sen5Vec2       ; Sensor5 Vector2
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads5
        MAR     *,AR3
        LACC    #Sen5Vec3       ; Sensor5 Vector3
        SAMM    BMAR
        call    StartMAC

        LAR     ar3,#st_ads5
        MAR     *,AR3
        LACC    #Sen5Vec4       ; Sensor5 Vector4
        SAMM    BMAR
        call    StartMAC

Ck_pat:
        SPM     0
        LACC    Temp_pat
        SUB     Pattern
        BCND    start_ad,NEQ    ; If Temp_pat != Pattern goto start_ad

        RET

*****************************************************************************
* Function : StartMAC  Mutiply accumulate data at BMAR and (dma)
* Input    : BMAR=pma, ar5=start address of buffer, Length = length of buffer
* Output   : Send all 32 bit to RS232 send High order first
*****************************************************************************
StartMAC:
```

```
        ZAP
        RPT     Length
        MADS    *+
        APAC
        SACL    LowWord                 ; Save Low word
        SACH    HighWord                ; Save Upper word
        LACC    HighWord
        CALL    SendWord                ; Send Upper word
        LACC    LowWord
        CALL    SendWord                ; Send Lower word
        RET

        .end
```

ไฟล์ atoi.inc

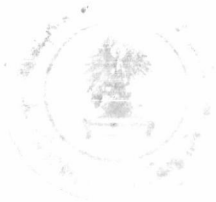
```
*********************************************************************
* Function : ATOI translate ASCII to Integer
* Input    : Buffer at ar7
* Output   : ACC
*********************************************************************

ATOI:
        lacc    count
        add     #1
        sacl    count           ; increment count

        LACC    *+              ; load data at buffer to ACC
        SUB     #13
        bcnd    ATOI,GT     ; branch if ACC > 0
end_count:

; Calcualte Value of data in buffer
        LACC    *-              ;
;        lacc    count
;        sub     #1
;        sacl    count

        LACC    *-              ; read 0Dh
        lacc    count
        sub     #1
        sacl    count

        LACC    *-              ; read last Digit
        SUB     #30h            ; minus 30h
        sacl    data
        lacc    count
        sub     #1
        sacl    count
        bcnd    end_atoi,LEQ     ; ACC <= 0

        LACC    *-              ; READ SECOND DIGIT (X10)
        SUB     #30h
        sacl    Temp
        LT      Temp
        MPY     #10
        LTP     count           ; dummy load count to TREG0 and P to ACC
        add     data
        sacl    data
        lacc    count           ; decrease count and check for end
        sub     #1
        sacl    count
        bcnd    end_atoi,LEQ     ; ACC <= 0

        LACC    *-              ; READ THIRD DIGIT (X 100)
        SUB     #30h
        sacl    Temp
        LT      Temp
```

```
        MPY     #100
        LTP     count           ; dummy load count to TREG0 and P to ACC
        add     data
        sacl    data
        lacc    count           ; decrease count and check for end
        sub     #1
        sacl    count
        bcnd    end_atoi,LEQ     ; ACC <= 0

        LACC    *-              ; READ FOURTH DIGIT (X1000)
        SUB     #30h
        sacl    Temp
        LT      Temp
        MPY     #1000
        LTP     count           ; dummy load count to TREG0 and P to ACC
        add     data
        sacl    data
        lacc    count           ; decrease count and check for end
        sub     #1
        sacl    count
        bcnd    end_atoi,LEQ    ; ACC <= 0

        LACC    *-              ; READ FIFTH DIGIT (X10000)
        SUB     #30h
        sacl    Temp
        LT      Temp
        MPY     #10000
        LTP     count           ; dummy load count to TREG0 and P to ACC
        add     data
        sacl    data
        lacc    count           ; decrease count and check for end
        sub     #1
        sacl    count
        bcnd    end_atoi,LEQ    ; ACC <= 0
; Error Trap
end_atoi:
        lacc    data
        RET
```

ไฟล์ aicinit.inc

```
*
*****************************************************************
*  DESCRIPTION: This routine initializes the TLC320C46 for      *
*               a 8Khz sample rate with a gain setting of 1     *
*****************************************************************
* aic initialization data
*
AICINIT: SPLK    #20h,TCR          ; To generate 10 MHz from Tout
         SPLK    #1,PRD          ; for AIC master clock
         MAR     *,AR0
         LACC    #0008h            ; Non continuous mode
         SACL    SPC               ; FSX as input
         LACC    #00c8h            ; 16 bit words
         SACL    SPC
         LACC    #080h           ; Pulse AIC reset by setting it low
         SACH    DXR
         SACL    GREG
         LAR     AR0,#0FFFFh
         RPT     #10000            ; and taking it high after 10000 cycles
         LACC    *,0,AR0           ; (.5ms at 50ns)
         SACH    GREG
         LDP     #TA             ;
         SETC    SXM             ;
         LACC    TA,9            ; Initialized TA and RA register
         ADD     RA,2            ;
         CALL    AIC_2ND         ;
         ;------------------------
```

```
        LDP     #TB
        LACC    TB,9            ; Initialized TB and RB register
        ADD     RB,2            ;
        ADD     #02h            ;
        CALL    AIC_2ND         ;
        ;-------------------------
        LDP     #AIC_CTR
        LACC    AIC_CTR,2       ; Initialized control register
        ADD     #03h             ;
        CALL    AIC_2ND         ;
        RET                     ;

AIC_2ND:
        LDP     #0
        SACH    DXR             ;
        CLRC    INTM
        IDLE
        ADD     #6h,15          ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH    DXR             ;
        IDLE
        SACL    DXR             ;
        IDLE
        ZAP
;       LACL    #0              ;
        SACL    DXR             ; make sure the word got sent
        IDLE
        SETC    INTM
        RET                     ;
```



ประวัติผู้เขียน

นายโอภาส ศิริครรชิตถาวร เกิดวันที่ 5 ตุลาคม พ.ศ. 2513 ที่เขตคลองเตย จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตร์ สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ ในปีการศึกษา 2535 จากนั้นได้ทำงานที่บริษัท เพอร์ซันแนล คอมมูนิเคชั่น เน็ตเวอร์ค จำกัด เป็นเวลา 1 ปี และได้เข้าศึกษาต่อระดับปริญญามหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2537