รายการอ้างอิง

ภาษาไทย

มานะ ศรียุทธศักดิ์, มนตรี สวัสดิ์ศฤงฆาร และธารา ชลปราณี, "รายงานการวิจัยและพัฒนาโครงการพัฒนาเทคโนโลยีวัสดุและชิ้นส่วนอิเล็กทรอนิกส์ เรื่อง การพัฒนาหัววัดน้ำตาลกลูโคสและระบบใช้งานเชิงอุตสาหกรรม", ศูนย์เทคโนโลยีอีเล็กทรอนิกส์ และคอมพิวเตอร์แห่งชาติ กระทรวงวิทยาศาสตร์ เทคโนโลยี และสิ่งแวดล้อม, 2536.

มานะ ศรียุทธศักดิ์,"การประดิษฐ์ไบโอเซนเซอร์สำหรับตรวจวัดน้ำตาลกลูโคส และการประยุกต์ใช้งานทางชีวภาพ", การประชุมวิชาการทางไฟฟ้า ประจำปี 2534, หน้า 254-263.

นารเมธ นานานุกูล, มานะ ศรียุทธศักดิ์, ธารา ชลปราณี และมนตรี สวัสดิ์ศฤงฆาร, "ระบบวัดน้ำตาลกลูโคสแบบโฟลว์อินเจ็กชัน," การประชุมวิชาการวิทยาศาสตร์ และเทคโนโลยีแห่งประเทศไทยครั้งที่ 19, สมาคมวิทยาศาสตร์แห่งประเทศไทยในพระบรมราชูปถัมภ์,โรงแรมดุสิต เจบี หาดใหญ่, 27-29 ตุลาคม 2536.

นารเมธ นานานุกูล, มานะ ศรียุทธศักดิ์, ธารา ชลปราณี และมนตรี สวัสดิ์ศฤงฆาร, "ระบบวัดน้ำตาลกลูโคสสำหรับการใช้งานเชิงอุตสาหกรรม,"การประชุมใหญ่ทางวิชาการประจำปี 2536,วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์, กรุงเทพมหานคร, 27-30 พฤษจิกายน 2536.

ภาษาอังกฤษ

Terje A. Skotheim(Ed),"Handbook of Conducting Polymers",Vol.1, Chapter 3, 1986.

Elizabeth A.H. Hall, "Biosensors", Chapter 1, 1990.

George S. Wilson and Daniel R.Thevenot, "Biosensors :a practical

approach", chapter1,2, 1990.

Elizabeth A.H. Hall, "Overview of Biosensors", Biosensor and Chemical sensors, pp.1-13, 1990.

Wolfgang Schuhmann and Ruth Kittsteiner-Eberle, "Evaluation of poly pyrrole/glucose oxidase electrodes in flow-injection systems for sucrose determination," Biosensors & Bioelectronics, pp. 263-273, 1991.

M.Trojanowicz et al., "Enzyme Entrapped Polypyrrole Modified Electrode for Flow-Injection Determination of Glucose," Biosensors & Bio electronics, pp. 149-156, 1991.

ภาคผนวก

โปรแกรมที่ใช้เก็บสัญญาณ ssys.c

```
/* collect data by using 8254 timer trig operation */
/* this program must be linked with diskio.c, get.c
,diskio.c and rp2.c when compiled                    */
/*-------------------------------------*/

/* header file */
#include "sys1.h"
#include "sys2.h"
#include <dos.h>
#include "disk.h"
#include "get.h"
#include "filter.h"


#define   INTR     13       /* IRQ = 5 plus 8 */
#define   b13      0x02ed
#define   b14      0x02ee
#define   b15      0x02ef

#ifdef __cplusplus
#define __CPPARGS ...
#else
#define __CPPARGS
#endif

extern int    maxx, maxy; /*  external variable  */
extern int    Xmax,Ymax;
extern long   ct,ot;
extern int    bcol,tmp_key;
extern float  maxd,scx,scy;

void   initialize(void); /*  function  declaration  */
void   mcplot(float far *y,int n);

int    j=1;
int    pg,lp,pt;
int    v1,v2,v3,v4,num,value[100];
float  valuef[100],tm1[4000],tm2[4000];

main()
{
void interrupt ( *oldhandler)(__CPPARGS);
void interrupt handler(__CPPARGS);
int  tmp1,tmp2,tmp3;
int  i,s;
union tm {
char ch[2];
int  key;
} u;
DSP_FILE  *in,*out1,*out2;
char *cp;

/*---------------set frame---------------*/
bcol = 1;          /*  blue background  */
```

```
clrscr();
printf("Enter number of point:");
num = 5000;
scanf("%d",&num);

initialize();
setbkcolor(bcol);

/* set initial value */
for(lp = 1;lp<=num+1;lp++)
{
valuef[lp] = 0.;
}

for(lp = 1;lp<=4000;lp++)
{
tm1[lp] = 0;
tm2[lp] = 0;
}

/* plot scale and axis */
mcplot(valuef,num);
pg = 1;
pt = 1;

do    /*  wait for esc  */
tmp_key = getch();
while (tmp_key != 27);
/*----------------------------end--------------------*/


/*--------------change interrupt vector------------*/
/* save the old interrupt vector */
oldhandler = getvect(INTR);

/* install the new interrupt handler */
setvect(INTR, handler);
/*-----------------end-------------------------*/

/*--------------atod SETUP  2 hz----------------*/
outportb(0x2e3,0x25);
outportb(0x2e2,0x0A);
outportb(b15,0xb4);
outportb(b14,40);
outportb(b14,0);
outportb(b15,0x74);
outportb(b13,80);
outportb(b13,195);
/*---------------end---------------------*/


/*---------------enable int--------------*/
i    = inportb(0x21);
```

```
tmp1 = inportb(0x21);
tmp2 = 223;
tmp3 = tmp1  & tmp2;
outportb(0x21,tmp3);
/*---------------end---------------------*/


for(;;) {
if( bioskey(1) != 0 ){
u.key = bioskey(0);

for(;;){
if( bioskey(1) != 0 ){
u.key = bioskey(0);
switch(u.ch[1]){

/* increase sampling frequency  */
if( u.ch[0] == 'i'){
outportb(b15,0xb4);
outportb(b14,40);
outportb(b14,0);
outportb(b15,0x74);
outportb(b13,32);
outportb(b13,78);
}

/* decrease sampling frequency */
if( u.ch[0] == 'd'){
outportb(b15,0xb4);
outportb(b14,40);
outportb(b14,0);
outportb(b15,0x74);
outportb(b13,80);
outportb(b13,195);
}

/* quit program */
if( u.ch[0] == 'q'){
outportb(0x21,i);
setvect(INTR, oldhandler);

closegraph();
exit(1);
}


/* write data to file */
if( u.ch[0] == 'f'){
outportb(0x21,i);
cp = get_string("ENTER FILENAME :");

strcat(cp,".dat");
out2 = open_write(cp,FLOAT,1,num+1);
valuef[0] = num;
write_record((char *)valuef,out2);
```

```
setviewport(0,0,maxx,maxy,1);
clearviewport();
mcplot(valuef,num);
setviewport(maxx*leftmargin,maxy*topmargin,maxx*rightmargin
,maxy*bottommargin,1);
moveto(0,(maxd-valuef[1])*scy);
setcolor(13);
for(lp = 2;lp<=num+1;lp++)
{
lineto(lp*scx,(maxd-valuef[lp])*scy);
}
getch();
outportb(0x21,tmp3);
}

}

}

}


/*-----------------------int routine --------------*/
void interrupt handler(__CPPARGS)
{

v1 = inportb(b1);
v2 = inportb(b0);
v3 = v1<<4;
v4 = v2>>4;
value[0] = v3+ v4 ;
valuef[1] = (float)value[0];
valuef[1] = (valuef[1] - 2048)*20.0/4096.0;
tm1[pt++]    = valuef[1];

setactivepage(pg);
clearviewport();
setviewport(0,0,maxx,maxy,1);
mcplot(valuef+1,num);

setviewport(maxx*leftmargin,maxy*topmargin,maxx*rightmargin
,maxy*bottommargin,1);
moveto(0,(maxd-valuef[1])*scy);
setcolor(13);
for(lp = 2;lp<=num+1;lp++)
{
lineto(lp*scx,(maxd-valuef[lp])*scy);

}
setvisualpage(pg);
if(pg == 0)
{
pg=1;
}
```

```
else
{
pg=0;
}
movmem( &valuef[1],&valuef[2],4*num);

outportb(0x20,0x20);            /* end of int signal */
}
/*----------------------end int-----------------------*/
```

โปรแกรมที่ใช้วิเคราะห์สัญญาณ rf.c

```
/*  read file and evaluate peak value */
/*  this program must be link with diskio.c
rp .c and get.c when compile      */
/*-----------------------------------*/

/* header file */
#include "sys1.h"
#include "sys2.h"
#include "disk.h"
#include "get.h"

/* variable declaration */
extern int     maxx, maxy;
extern int     Xmax,Ymax;
extern long    ct,ot;
extern int     i,z,bcol,tmp_key;
extern float   maxd,scx,scy;
extern float   mul;

/* function declaration */
void   initialize(void);
void   mcplot(float far *y,int n);
void   fmax(float *fit,float *lat,int ft);
void   plot(float max,int point);

main()
{
float      *valuef;
DSP_FILE   *in1,*in2;
char       *np,*cp,*fp,c;
int        lp,*num;
float      *ptr,max,min,pk[20];
int        n=0,k=0,j=0,i,ft,lt;


/* read filename */
cp =  get_string("enter filename to read\n");
strcat(cp,".dat");

initialize();  /* initialize graphic mode */
setbkcolor(1);

in2 = open_read(cp);
valuef = calloc(2565,sizeof(float));
valuef = read_float_record(in2);
*num = valuef[0];
mcplot(valuef,*num); /* plot frame and axis */
setviewport(maxx*leftmargin,maxy*topmargin
,maxx*rightmargin,maxy*bottommargin,1);
setcolor(13);
moveto(0,(maxd-valuef[1])*scy);
for(i=2;i<=(*num);i++) /* plot data on screen */
{
lineto(i*scx,(maxd-valuef[i])*scy);
}
```

```
do
tmp_key = getch();
while(tmp_key != 27);

/* peak detection */
for( i=0;i <= *num;i++){
if(k < 20){
if(valuef[i] > 0.6)
k++;
if(valuef[i] < 0.6)
k=0;
}
else{
if(k == 20){
ft = i;
k = 52;
}
if(valuef[i] < 0.6){
lt = i;
k  = 0;
n++;
fmax(&valuef[ft],&valuef[lt],ft);
}
}
}

getch();
closegraph();
}

/* find maxvalue of peak */
void fmax(float *fit,float *lat,int ft)
{
float *i;
float max,*tt;
int point;

tt  = fit;
max = *fit;
for( i=fit;i <= lat;i++,fit++){
if(*fit >= max)
max = *fit;
}
for( i=tt;i <= lat;i++,tt++,ft++){
if(*tt == max){
point = ft;
i = lat;
}
}
plot(max,point);

}

/* plot max value on screen */
```

```
void  plot(float max,int point)
{
char buf[5];
setviewport(maxx*leftmargin,maxy*topmargin
,maxx*rightmargin,maxy*bottommargin,1);
setcolor(14);
settextstyle(DEFAULT_FONT,1,0);
settextjustify(0,0);
gcvt(max,3,buf);
outtextxy(point*scx+3,(maxd-max)*scy-8,buf);
}
```

rp.c

```
/*  Function definition  */
/*-----------------------------------*/

#include "sys1.h"                   /*  header file  */
#include "sys2.h"

/*  variable declaration  */
int     graphdriver;                /*  The Graphics device driver  */
int     graphmode;                  /*  The Graphics mode value  */
int     maxx, maxy;                 /*  The maximum resolution of the screen
int     errorcode;                  /*  Reports any graphics errors  */
int     numheight,numwidth;
float   aspectratio;                /*  Aspect ratio of a pixel on the screen
char    xname[][20]={"Time(Minute)"};
char    yname[][20]={"I(10E-7 Ampere)"};
int     Xmax,Ymax,bcol,tmp_key;
long    ct,ot;
float   maxd,scx,scy,mind;
float   mul;


/*  function declaration  */
void    initialize(void);
void    findblock(float *maxdata,float *mindata,int *countblock);
void    mcplot(float far *y,int n);


void initialize(void)  /* initialize graphic mode */
{
int xasp, yasp;           /*  Used to read the aspect ratio */
int g_driver,g_mode;

detectgraph(&g_driver,&g_mode);
g_mode = VGAMED;
initgraph( &g_driver, &g_mode, "c:\tc" );
errorcode = graphresult();       /* Read result of initialization*/

if( errorcode != grOk ){         /* Error occured during init */
printf(" Graphics System Error: %s\n", grapherrormsg( errorcode ) );
exit(1);
}

maxx = getmaxx();
maxy = getmaxy();                /*  Read size of screen */
Xmax = maxx+1;
Ymax = maxy+1;

getaspectratio( &xasp, &yasp );  /* read the hardware aspect */
aspectratio = (double)xasp / (double)yasp; /* Get correction factor */

numheight=textheight("O");
numwidth=textwidth("O");

}
```

```
/* calculate scale on x and y axis */
void findblock(float *maxdata,float *mindata,int *countblock)
{
float rangeblock,factor,tmp;
int   i,sign;

if ((*maxdata-*mindata)<1.0e-10) {
*maxdata+=fabs(*maxdata);
*mindata-=fabs(*mindata);
}
i=0;
rangeblock=(*maxdata-*mindata)/10.0;
factor=(rangeblock>=10.0) ? 0.1 : 10.0;
while (rangeblock<1.0 || rangeblock>=10.0) {
rangeblock*=factor;
i++;
}
rangeblock=ceil(rangeblock)*pow(factor,-i);
if (((*maxdata)*(*mindata))<=0.0) {
tmp=0.0;
*countblock=0;
while (tmp<*maxdata) {
tmp+=rangeblock;
(*countblock)++;
}
*maxdata=tmp;
tmp=0.0;
while (tmp>*mindata) {
tmp-=rangeblock;
(*countblock)++;
}
*mindata=tmp;
}
else {
*countblock=(int)ceil((*maxdata-*mindata)/rangeblock);
sign=(*mindata<0) ? -1 : 1;
*mindata=fabs(*mindata);
factor=(*mindata>=10.0) ? 0.1 : 10.0;
i=0;
while (*mindata<1.0 || *mindata>=10.0) {
*mindata*=factor;
i++;
}
*mindata=((sign>0)?floor(*mindata):-1.0*ceil(*mindata))*pow(factor,-i);
*maxdata=rangeblock*(*countblock)+*mindata;
}
}


/* plot axis and data on screen */
void mcplot(float far *y,int n)
{
int maxviewx,maxviewy,countblock,linex,liney,xblock,xblock2;
int i,j,grid=1,line_dot=0,grtype=0;
```

```
float maxdata,mindata,sclx,scly,rangeblock,tmp,tmp2;
char buf[15];
float value,x0=0,dx=0.1,db=30.0;
struct linesettingstype linetype;

setcolor(15);
rectangle(0,0,maxx,maxy);
maxviewx=widthviewx*maxx;
maxviewy=widthviewy*maxy;
rectangle(maxx*leftmargin-offset,maxy*topmargin-offset*aspectratio
,maxx*rightmargin+offset,maxy*bottommargin+offset*aspectratio);
setviewport(maxx*leftmargin,maxy*topmargin,maxx*rightmargin
,maxy*bottommargin,1);

/* Find maxdata,mindata */
switch (grtype) {
case 0 :
case 1 :
maxdata=5.0;
mindata=0.0;
if (maxdata==0 && mindata==0) { /* Solve maxdata=0 and mindata=0 */
mindata=-10.0;
maxdata=10.0;
}
break;
}
findblock(&maxdata,&mindata,&countblock);
maxd = maxdata;
mind = mindata;
scx=sclx=(float)maxviewx/((float)(grtype==0)?(n):(n-1));
scy=scly=maxviewy/(maxdata-mindata);
setviewport(0,0,maxx,maxy,1);
rangeblock=(maxdata-mindata)/countblock;
value=maxdata;
linex=maxx*leftmargin-offset;

for (i=0;i<=countblock;i++) {       /* Draw scale y-axis */
liney=maxy*topmargin+rangeblock*scly*i;
line(linex,liney,linex-longlength,liney);
if (grid) {
getlinesettings(&linetype);
setlinestyle(DOTTED_LINE,linetype.upattern,linetype.thickness);
line(linex,liney,linex+maxviewx+2*offset,liney);
setlinestyle(linetype.linestyle,linetype.upattern,linetype.thickness);
}
gcvt(value,3,buf);
j=strlen(buf)+1;
outtextxy(linex-longlength-j*numwidth,liney-numheight/2,buf);
value-=rangeblock;
if (fabs(value)<rangeblock/10.0)
value=0.0;
for (j=1;i<countblock&&j<=4;j++) {
tmp=liney+(rangeblock*scly*j)/5;
line(linex,tmp,linex-shortlength,tmp);
}
```

```
}
xblock=(grtype==0&&((int)pow(2,logN(n,2))==n)) ? 8 : 5;
xblock2=(grtype==0&&((int)pow(2,logN(n,2))==n)) ? 8 : 10;
value=x0;
liney=maxy*bottommargin+offset*aspectratio;
tmp2=liney+longlength*aspectratio+numheight/2;
for (i=0;i<=xblock;i++) {          /* Draw scale x-axis */
linex=maxx*leftmargin+(i*maxviewx)/(float)xblock;
line(linex,liney,linex,liney+longlength*aspectratio);
if (grid) {
getlinesettings(&linetype);
setlinestyle(DOTTED_LINE,linetype.upattern,linetype.thickness);
line(linex,liney,linex,liney-maxviewy-2*aspectratio*offset);
setlinestyle(linetype.linestyle,linetype.upattern,linetype.thickness);
}
gcvt(value*mul,3,buf);
j=strlen(buf);
outtextxy(linex-j*numwidth/2-2,tmp2,buf);
value+=dx*((grtype==0)?(n):(n-1))/(float)xblock;
}
outtextxy(15,maxy*topmargin/2.0,yname[grtype]);  /*  write axis name */
outtextxy(maxx*0.8,tmp2+numheight*2,xname[grtype]);


}
```

get.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

/*****************************************************************************

GET.C - Source code for user input functions

get_string      get string from user with prompt
get_int         get integer from user with prompt and range
get_float       get float from user with prompt and range

*****************************************************************************/

/*****************************************************************************

get_string - get string from user with prompt

Return pointer to string of input text, prompts user with string
passed by caller.  Indicates error if string space could not be
allocated.  Limited to 80 char input.

char *get_string(char *prompt_string)

prompt_string  string to prompt user for input

*****************************************************************************/

char *get_string(title_string)
char *title_string;
{
char *alpha;                                /* result string pointer */

alpha = (char *) malloc(80);
if(!alpha) {
printf("\nString allocation error in get_string\n");
exit(1);
}
printf("\nEnter %s : ",title_string);
scanf("%s",alpha);

return(alpha);
}
/*****************************************************************************

get_int - get integer from user with prompt and range

Return integer of input text, prompts user with prompt string
and range of values (upper and lower limits) passed by caller.

int get_int(char *title_string,int low_limit,int up_limit)

title_string  string to prompt user for input
low_limit     lower limit of acceptable input (int)
```

```
up_limit      upper limit of acceptable input (int)

*****************************************************************************

int get_int(title_string,low_limit,up_limit)
char *title_string;
int low_limit,up_limit;
{
int i,error_flag;
char *get_string();                    /* get string routine */
char *cp,*endcp;                       /* char pointer */
char *stemp;                           /* temp string */

/* check for limit error, low may equal high but not greater */
if(low_limit > up_limit) {
printf("\nLimit error, lower > upper\n");
exit(1);
}

/* make prompt string */
stemp = (char *) malloc(strlen(title_string) + 60);
if(!stemp) {
printf("\nString allocation error in get_int\n");
exit(1);
}
sprintf(stemp,"%s [%d...%d]",title_string,low_limit,up_limit);

/* get the string and make sure i is in range and valid */
do {
cp = get_string(stemp);
i = (int) strtol(cp,&endcp,10);
error_flag = (cp == endcp) || (*endcp != '\0'); /* detect errors */
free(cp);                                       /* free string space */
} while(i < low_limit || i > up_limit || error_flag);

/* free temp string and return result */
free(stemp);
return(i);
}
/*****************************************************************************

get_float - get float from user with prompt and range

Return double of input text, prompts user with prompt string
and range of values (upper and lower limits) passed by caller.

double get_float(char *title_string,double low_limit,double up_limit)

title_string  string to prompt user for input
low_limit     lower limit of acceptable input (double)
up_limit      upper limit of acceptable input (double)

*****************************************************************************

double get_float(title_string,low_limit,up_limit)
```

```
char *title_string;
double low_limit,up_limit;
{
double x;
int error_flag;
char *get_string();                    /* get string routine */
char *cp,*endcp;                       /* char pointer */
char *stemp;                           /* temp string */

/* check for limit error, low may equal high but not greater */
if(low_limit > up_limit) {
printf("\nLimit error, lower > upper\n");
exit(1);
}

/* make prompt string */
stemp = (char *) malloc(strlen(title_string) + 80);
if(!stemp) {
printf("\nString allocation error in get_float\n");
exit(1);
}

sprintf(stemp,"%s [%1.2g...%1.2g]",title_string,low_limit,up_limit);

/* get the string and make sure x is in range */
do {
cp = get_string(stemp);
x = strtod(cp,&endcp);
error_flag = (cp == endcp) || (*endcp != '\0'); /* detect errors */
free(cp);                                       /* free string space */
} while(x < low_limit || x > up_limit || error_flag);

/* free temp string and return result */
free(stemp);
return(x);
}
```

diskio.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/*************************************************************************

DISKIO.C - Source code for DSP data format read and write functions

open_read           open DSP data file to be read
open_write          create header and open DSP data file for write
read_record         read one record
write_record        write one record
seek_record         seek to beginning of specified record
read_float_record   read one record and convert to float array
read_trailer        read the trailer text
write_trailer       write the trailer text
append_trailer      read a trailer and add to it

*************************************************************************/

/* DSP INFORMATION STRUCTURE FOR MANIPULATING DSP DATA FILES */
typedef struct {
unsigned char type;              /* data type 0-7 as defined below */
unsigned char element_size;      /* size of each element */
unsigned short int records;      /* number of records */
unsigned short int rec_len;      /* number of elements in each record */
char *name;                      /* pointer to file name */
FILE *fp;                        /* pointer to FILE structure */
} DSP_FILE;

/* FILE HEADER STRUCTURE FOR DSP DATA FILES */
typedef struct {
unsigned char type;              /* data type 0-7 as defined below */
unsigned char element_size;      /* size of each element */
unsigned short int records;      /* number of records */
unsigned short int rec_len;      /* number of elements in each record */
} HEADER;

/* defines for data type used ind DSP data file header and structure */

#define UNSIGNED_CHAR    0
#define UNSIGNED_INT     1
#define UNSIGNED_LONG    2
#define FLOAT            3
#define SIGNED_CHAR      4
#define SIGNED_INT       5
#define SIGNED_LONG      6
#define DOUBLE           7

/*************************************************************************

open_read - open a DSP data file for read

Returns a pointer to a DSP_FILE structure allocated by the
function and opens file_name.
```

```
Allocation errors or improper type causes a call to exit(1).
A bad file_name returns a NULL pointer.

DSP_FILE *open_read(char *file_name)

*****************************************************************************

DSP_FILE *open_read(file_name)
char *file_name;            /* file name string */
{
DSP_FILE *dsp_info;
int status;

/* allocate the DSP data file structure */

dsp_info = (DSP_FILE *) malloc(sizeof(DSP_FILE));
if(!dsp_info) {
printf("\nError in open_read: structure allocation, file %s\n",
file_name);
exit(1);
}

/* open file for binary read and update */
dsp_info->fp = fopen(file_name,"r+b");
if(!dsp_info->fp) {
printf("\nError opening %s in open_read\n",file_name);
return(NULL);
}

/* copy and allocate file name string for the DSP_FILE structure */
dsp_info->name = malloc(strlen(file_name) + 1);
if(!dsp_info->name) {
printf("\nUnable to allocate file_name string in open_read\n");
exit(1);
}
strcpy(dsp_info->name,file_name);

/* read in header from file */
status = fread((char *)dsp_info,sizeof(HEADER),1,dsp_info->fp);
if(status != 1) {
printf("\nError reading header of file %s\n",file_name);
exit(1);
}

/* return pointer to DSP_FILE structure */
return(dsp_info);
}

/*****************************************************************************

open_write - open a DSP data file for write

Returns a pointer to a DSP_FILE structure allocated by the function.
Allocation errors or improper type causes a call to exit(1).
```

```
A bad file name returns a NULL pointer.

DSP_FILE *open_write(char *file_name,int type,int records,int rec_len)

file_name       pointer to file name string
type            type of DSP data (0-7 specified in defines)
records         number of records of data to be written
rec_len         number of elements in each record

*****************************************************************************

DSP_FILE *open_write(file_name,type,records,rec_len)
char *file_name;              /* file name string */
int type;                     /* data type 0-7 */
unsigned short int records;   /* number of records to be written */
unsigned short int rec_len;   /* elements in each record */
{
DSP_FILE *dsp_info;
int status;

/* allocate the DSP data file structure */
dsp_info = (DSP_FILE *) malloc(sizeof(DSP_FILE));
if(!dsp_info) {
printf("\nError in open_write: structure allocation, file %s\n",
file_name);
exit(1);
}

/* set the basics */
dsp_info->type = (unsigned char)type;
dsp_info->records = records;
dsp_info->rec_len = rec_len;

/* set element size from data type */
switch(type) {
case 0:
case 4:
dsp_info->element_size = sizeof(char);
break;
case 1:
case 5:
dsp_info->element_size = sizeof(short int);
break;
case 2:
case 6:
dsp_info->element_size = sizeof(long int);
break;
case 3:
dsp_info->element_size = sizeof(float);
break;
case 7:
dsp_info->element_size = sizeof(double);
break;
default:
printf("\nUnsupported data type, file %s\n",file_name);
```

```
exit(1);
}

/* open file for binary write */
dsp_info->fp = fopen(file_name,"wb");
if(!dsp_info->fp) {
printf("\nError opening %s in open_write\n",file_name);
return(NULL);
}

/* copy and allocate file name string for the DSP_FILE structure */
dsp_info->name = malloc(strlen(file_name) + 1);
if(!dsp_info->name) {
printf("\nUnable to allocate file_name string in open_write\n");
exit(1);
}
strcpy(dsp_info->name,file_name);

/* write header to file */
status = fwrite((char *)dsp_info,sizeof(HEADER),1,dsp_info->fp);
if(status != 1) {
printf("\nError writing header of file %s\n",file_name);
exit(1);
}

/* return pointer to DSP_FILE structure */
return(dsp_info);
}

/*****************************************************************************

read_record - read one record of DSP data file

Exits if a read error occurs or if the DSP_FILE structure is invalid.

void read_record(char *ptr,DSP_FILE *dsp_info)

ptr       pointer to previously allocated memory to put data
dsp_info  pointer to DSP data file structure

*****************************************************************************/

void read_record(ptr,dsp_info)
char *ptr;                  /* pointer to some type of data */
DSP_FILE *dsp_info;
{
int status;

if(!dsp_info) {
printf("\nError in DSP_FILE structure passed to read_record\n");
exit(1);
}

status = fread(ptr,dsp_info->element_size,
dsp_info->rec_len,dsp_info->fp);
```

```
fclose(dsp_info->fp);
if(status != dsp_info->rec_len) {
printf("\nError in read_record, file %s\n",dsp_info->name);
exit(1);
}
}

/*************************************************************************

write_record - write one record of DSP_FILE data

Exits if write error occurs or if the DSP_FILE structure is invalid.

void write_record(char *ptr,DSP_FILE *dsp_info)

ptr         pointer to data to write to disk (type in dsp_info)
dsp_info    pointer to DSP data file structure

*************************************************************************/

void write_record(ptr,dsp_info)
char *ptr;                    /* pointer to some type of data */
DSP_FILE *dsp_info;
{
int status;

if(!dsp_info) {
printf("\nError in DSP_FILE structure passed to write_record\n");
exit(1);
}

status = fwrite(ptr,dsp_info->element_size,
dsp_info->rec_len,dsp_info->fp);
fclose(dsp_info->fp);
if(status != dsp_info->rec_len) {
printf("\nError write_record, file %s\n",dsp_info->name);
exit(1);
}
}

/*************************************************************************

seek_record - seek to the beginning of a record of DSP data file

If improper seek is requested, the function calls exit(1).

void seek_record(int record_num,DSP_FILE *dsp_info)

record_num     integer record number to seek in DSP data file
dsp_info       pointer to DSP data file structure

*************************************************************************/

void seek_record(record_num,dsp_info)
int record_num;                   /* record number to seek */
```

```
DSP_FILE *dsp_info;
{
long int position,bytecount;

if(!dsp_info) {
printf("\nError in DSP_FILE structure passed to seek_record\n");
exit(1);
}

/* get the number of bytes to the beginning of the record of data */
bytecount = (long)dsp_info->element_size * (long)dsp_info->rec_len
* (long)record_num;
bytecount += sizeof(HEADER);   /* add on the header length */

/* find the end of file location */
fseek(dsp_info->fp,0L,2);       /* seek to end of file */
position = ftell(dsp_info->fp);

/* check for errors in position */
if(position <= 0L || position < bytecount) {
printf("\nError in locating record in file %s\n",dsp_info->name);
exit(1);
}
fseek(dsp_info->fp,bytecount,0);         /* move to record */
}

/*************************************************************************

read_float_record - read one record of DSP data file and convert
to float array of values.

Returns a pointer to the beginning of the allocated float array
of values representing the record read from the DSP_FILE.

Exits if a read or allocation error occurs.

float *read_float_record(DSP_FILE *dsp_info)

*************************************************************************/

float *read_float_record(dsp_info)
DSP_FILE *dsp_info;
{
void read_record();
static long int prev_size = 0; /* previous size in bytes */
static double *buf;            /* input buffer to read data in */

float *out;                    /* return output pointer */
float *out_ptr;

long int byte_size;            /* current size in bytes */
int i,length;

length = dsp_info->rec_len;
```

```
byte_size = (long)length*dsp_info->element_size;

/* check to see if we have to allocate the input buffer */
if(byte_size != prev_size) {

if(prev_size != 0) free(buf);   /* free old buffer */

/* allocate input buffer area cast to double for worst case alignment
buf = (double *) calloc(length,dsp_info->element_size);

if(!buf) {
printf("\nAllocation error in input buffer\n");
exit(1);
}

prev_size = byte_size;      /* lattest size */
}

/* allocate the output pointer only if conversion required */
if(dsp_info->type != FLOAT) {
out = (float *) calloc(length,sizeof(float));
if(!out) {
printf("\nAllocation error in read_float_record\n");
exit(1);
}
}

/* read the record into buf */
read_record((char *)buf,dsp_info);

/* perform conversion to floating point */

out_ptr = out;

switch(dsp_info->type) {
case UNSIGNED_CHAR: {
unsigned char *uc_ptr;
uc_ptr = (unsigned char *)buf;
for(i = 0 ; i < length ; i++)
*out_ptr++ = (float)(*uc_ptr++);
}
break;
case SIGNED_CHAR: {
char *sc_ptr;
sc_ptr = (char *)buf;
for(i = 0 ; i < length ; i++)
*out_ptr++ = (float)(*sc_ptr++);
}
break;
case SIGNED_INT: {
int *si_ptr;
si_ptr = (int *)buf;
for(i = 0 ; i < length ; i++)
*out_ptr++ = (float)(*si_ptr++);
}
```

```
break;
case UNSIGNED_INT: {
unsigned int *ui_ptr;
ui_ptr = (unsigned int *)buf;
for(i = 0 ; i < length ; i++)
*out_ptr++ = (float)(*ui_ptr++);
}
break;
case UNSIGNED_LONG: {
unsigned long *ul_ptr;
ul_ptr = (unsigned long *)buf;
for(i = 0 ; i < length ; i++)
*out_ptr++ = (float)(*ul_ptr++);
}
break;
case SIGNED_LONG: {
long *sl_ptr;
sl_ptr = (long *)buf;
for(i = 0 ; i < length ; i++)
*out_ptr++ = (float)(*sl_ptr++);
}
break;
case FLOAT:
out = (float *) buf;          /* no conversion */
prev_size = 0;                /* force next allocation */
break;
case DOUBLE: {
double *d_ptr;
d_ptr = buf;
for(i = 0 ; i < length ; i++)
*out_ptr++ = (float)(*d_ptr++);
}
break;
}

return(out);                  /* return converted pointer */
}

/*****************************************************************************

read_trailer - read trailer from existing DSP data file

Returns pointer to allocated string.
Returns NULL if read error or file error.
Exits if allocation error.

char *read_trailer(DSP_FILE *dsp_info)

*****************************************************************************/

char *read_trailer(dsp_info)
DSP_FILE *dsp_info;
{
int status,len;
long int position,bytecount,old_pos;
```

```
char *text;

/* get the number of bytes in the data */
bytecount = (long)dsp_info->element_size * (long)dsp_info->records
* (long)dsp_info->rec_len;
bytecount += sizeof(HEADER);   /* add on the header length */

/* save current position */
old_pos = ftell(dsp_info->fp);

/* find the end of file location */
fseek(dsp_info->fp,0L,2);       /* seek to end of file */
position = ftell(dsp_info->fp);

/* check for errors in position */
if(position <= 0L || position < bytecount) {
printf("\nError in trailer, file %s\n",dsp_info->name);
return(NULL);
}

/* try to allocate space for the trailer text */
len = (int)(position - bytecount);
text = malloc(len+1);       /* +1 for NULL termination */
if(!text) {
printf("\nError in allocating trailer space for file %s\n",
dsp_info->name);
exit(1);
}

/* read trailer */
fseek(dsp_info->fp,bytecount,0);    /* seek to beginning of trailer */
status = fread(text,sizeof(char),len,dsp_info->fp);
if(status != len) {
printf("\nError reading trailer of file %s\n",dsp_info->name);
return(NULL);
}

/* restore file position */
fseek(dsp_info->fp,old_pos,0);

/* make sure the trailer is NULL terminated */
text[len] = NULL;

return(text);
}

/*****************************************************************************

write_trailer - write trailer to DSP data file

Writes *text to the trailer of the DSP_FILE.
Exits with error message if write or positioning error.
Returns the number of characters in the trailer string.

int write_trailer(char *text,DSP_FILE *dsp_info)
```

```
**************************************************************************/

int write_trailer(text,dsp_info)
char *text;             /* pointer to trailer text */
DSP_FILE *dsp_info;
{
int status,len;
long int position,bytecount,old_pos;

/* get the number of bytes in the data */
bytecount = (long)dsp_info->element_size * (long)dsp_info->records
* (long)dsp_info->rec_len;
bytecount += sizeof(HEADER);   /* add on the header length */

/* save current position */
old_pos = ftell(dsp_info->fp);

fseek(dsp_info->fp,bytecount,0);    /* seek to beginning of trailer */

/* determine current file position and check if all records written */
position = ftell(dsp_info->fp);

if(bytecount != position || position <= 0L) {
printf("\nError in write_trailer: all records not written, file %s\n",
dsp_info->name);
exit(1);
}

/* write out the trailer */
len = strlen(text);

status = fwrite(text,sizeof(char),len,dsp_info->fp);
if(status != len) {
printf("\nError in writing trailer to file %s\n",dsp_info->name);
exit(1);
}

/* restore file position */
fseek(dsp_info->fp,old_pos,0);

return(len);        /* return length of trailer */
}

/**************************************************************************

append_trailer - read trailer from existing DSP data file
and add on a new string

Returns pointer to new trailer which is the old trailer + *string.
Exits if read error, file error, or allocation error.

char *append_trailer(char *string, DSP_FILE *dsp_info)

**************************************************************************/
```

```
char *append_trailer(string,dsp_info)
char *string;              /* string to add */
DSP_FILE *dsp_info;        /* input trailer file */
{
char *read_trailer();
char *trail;               /* trailer pointer */

trail = read_trailer(dsp_info);

/* re-allocate the output for the total size */

trail = realloc(trail,strlen(trail) + strlen(string) + 1);
if(!trail) {
printf("\nError in re-allocating trailer space for file %s\n",
dsp_info->name);
exit(1);
}

/*  add on string to trail */

strcat(trail,string);

return(trail);
}
```

ภาคผนวก ข

การทำงานของระบบ EB

บทนำ

ระบบนี้ออกแบบมาใช้สำหรับเคลือบสารกึ่งตัวนำหรือโลหะลงบนแผ่นฐาน ซึ่งchamber มีขนาดเส้นผ่านศูนย์กลาง 450 มม. และสูง 500 มม. bell jar ทำมาจากสเตนเลส ระบบนี้มีไว้สำหรับใช้ในการทดลอง และประดิษฐ์ผลิตภัณฑ์ที่มีขนาดเล็ก

ระบบปั๊มใช้ oil diffusion pump ขนาด 6 นิ้วเป็นปั๊มหลักและใช้ oil rotary pump เป็น fore pump เพื่อให้ได้สูญญากาศสูงๆในระยะเวลาอันสั้น

ในการระเหยไอของสารกึ่งตัวนำหรือโลหะใช้แหล่งจ่าย electron beam evaporation (EB) เป็นตัว evaporate การทำงานของระบบปั๊ม และระบบ evaporation จะกระทำด้วย manual

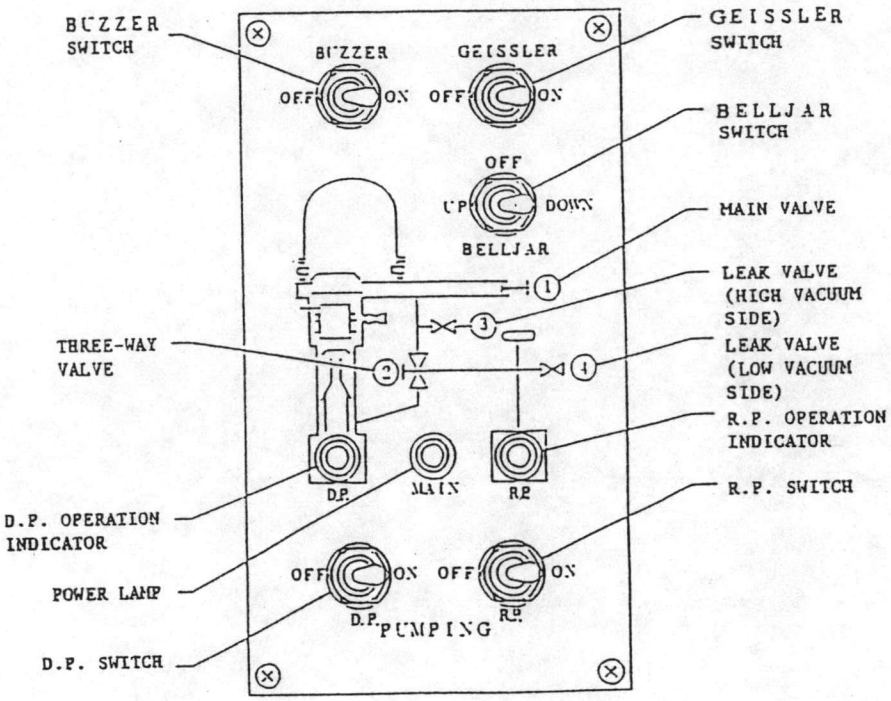
การทำงานของระบบปั๊ม (Pumping Operation)

การทำงานของปั๊มในระบบ EBV-6DH จะกระทำด้วย manual การควบคุมการทำงานทำได้โดยใช้วาล์วบนหน้าบัทม์ด้านหน้า และ Toggle switch บนแผงควบคุมปั๊ม op–1 บนหน้าบัทม์การทำงาน รูปที่ ข.1 และ ข.2 แสดงตำแหน่งของวาล์ว และ Toggle switch ที่ใช้ควบคุมการทำงานของปั๊ม

1. การเตรียมระบบปั๊ม (Preparation for Pumping System)

(สมมุติว่าสวิทช์ทุกตัวอยู่ในตำแหน่ง OFF, วาล์วทุกตัวเปิดอยู่และภายใน diffusion pump ความดันยังมีค่าต่ำพอ)

1.1 เปิดวาล์วน้ำ(เปิดสวิทช์ระบบทำน้ำเย็น, ปั๊มลม และเสียบมอเตอร์สูบน้ำ) เพื่อให้น้ำเย็นไหลเข้าสู่ bell jar)



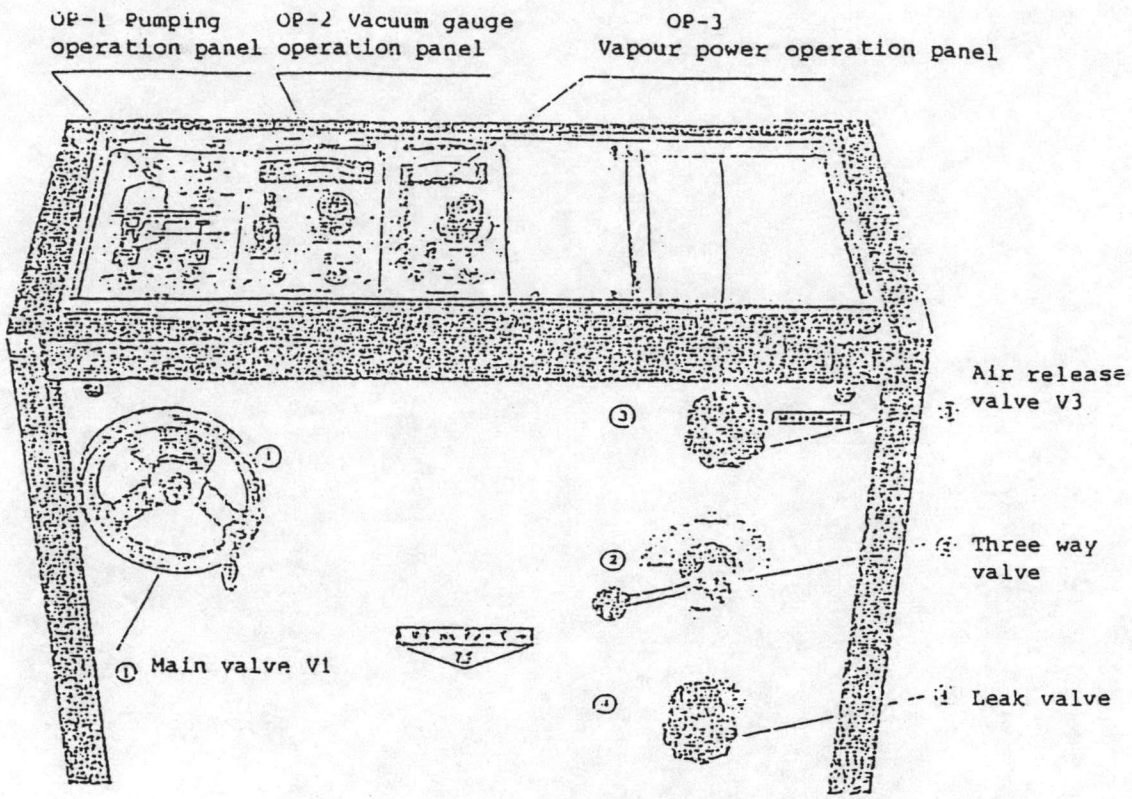
รูปที่ ข.1 แผงควบคุมระบบปั๊ม

1.2 เปิด breaker power switch (ที่อยู่ข้างฝาผนัง) แล้วเปิด power switch ที่อยู่บน switch board หลังจากนั้นเปิด main switch NFB ของระบบ เพื่อให้เครื่องทำงาน

1.3 ตรวจดูว่าหลอดไฟ PL1 ติดหรือไม่

1.4 เปิด toggle switch SSI บน OP-1 ปรับ three way valve 2

ไปที่ตำแหน่ง fore เพื่อดูดอากาศใน diffusion pump และเปิดสวิทช์ SS2 หลังจากนั้นประมาณ 20 นาที เมื่อน้ำมันของ diffusion pump เริ่มร้อนจึงเริ่มทำงาน



รูปที่ ข.2 ตำแหน่งของวาล์วที่ใช้ควบคุมการทำงานของปั๊ม

1.5 ตรวจดู back pressure ของ diffusion pump ด้วยหลอด Geissler ถ้าสังเกตุเห็นหลอด Geissler เรืองแสง ให้เปิด filament ของ vacuum gauge บน OP-2 และอ่านค่าความดัน ความดันที่อ่านได้จะอยู่ในช่วง 10^{-5} torr.

1.6 เติม liquid nitrogen ผ่านกรวย หลังจากนั้นความดันที่อ่านได้จะอยู่ในช่วง 10^{-7} torr.

2. การทำงานของระบบปั๊ม

สมมุติว่าได้ทำการเตรียมระบบปั๊มตามหัวข้อที่ 1 แล้ว

2.1 ปรับสวิทช์ SS5 ให้อยู่ในตำแหน่ง down bell jar จะลง และหยุดที่ feedthrough

2.2 ปิด three way valve 2

2.3 ปิดสวิทช์ filament บนแผงหน้าบัทม์ OP-2

2.4 ปรับ three way valve 2 ไปที่ตำแหน่ง rough และดูดอากาศออกจาก bell jar เมื่อความดันถึง 10^{-1} torr. ให้ปรับ three way valve 2 ไปที่ตำแหน่ง fore แล้วเปิด main valve 1 เพื่อเริ่มให้ diffusion pump ดูดอากาศภายใน belljar เมื่อความดันลดลงถึง 10^{-6} torr. จึงเริ่มทำการ evaporate ได้

2.5 หลังจากปลูกฟิล์มเสร็จเรียบร้อยแล้ว ให้ปิด main valve 1 แล้วเปิด valve3 เพื่อปล่อยอากาศให้เข้าไปใน bell jar จากนั้นรอให้อุณหภูมิภายใน chamber มีค่าใกล้เคียงกับอุณหภูมิภายนอกจึงสามารถเปิด chamber ได้

2.6 ปรับสวิทช์ SS5 ให้อยู่ในตำแหน่ง UP เพื่อเปิด bell jar

3. การหยุดระบบปั๊ม

3.1 ปิด main valve 1

3.2 ตรวจดู three way valve 2 ให้อยู่ที่ตำแหน่ง fore แล้วปิด diffusionpump ด้วยสวิทช์ SS2

3.3 รอประมาณ 30 นาที เมื่อน้ำมัน diffusion pump เริ่มเย็นลงจึงปิด threeway valve 2

3.4 ปิด rotary pump ด้วยสวิทช์ SS1 และปิดvalve 4 เพื่อปล่อยให้อากาศภายนอกเข้าไปใน rotary pump

4. ข้อควรระวังในการใช้งาน

4.1 เมื่อทำการเปลี่ยนหลอด GI-T หรือระบายอากาศเข้าไปในdiffusion

pumpให้เป็นความดันบรรยากาศ ในขณะที่ทำการบำรุงรักษา อย่าให้อากาศเข้าจากด้าน back pressure ของ pump ถ้าอากาศเข้าไปจะทำให้ pump jet เสีย หรือประสิทธิภาพของปั๊มเสื่อม

4.2 หลังจากที่ rotary pump หยุดทำงานจะต้องเปิดvalve 4 เพื่อทำการลดความดันภายในปั๊มให้เป็นความดันบรรยากาศ มิฉะนั้นน้ำมันใน rotary pump จะไหลกลับไปยังbell jar หรือ diffusion pump ซึ่งจะทำให้ประสิทธิภาพของปั๊มต่ำลง

5. การปลูกฟิล์ม (Deposition Operation)

5.1 ปล่อยอากาศให้เข้าไปใน chamber เพื่อให้อากาศภายใน chamber เป็นความดันบรรยากาศ

5.2 เปิด bell jar และติดตั้งแผ่นฐานบนที่ยึดแผ่นฐาน (substrate holder)

5.3 เปิด shutter และใส่วัตถุที่เป็นตัวระเหย แล้วจึงทำการปิด shutter

5.4 ทำความสะอาดภายใน chamber

5.5 ปรับ drive jig และตรวจดูว่าไม่มีสิ่งใดผิดปกติ

5.6 ปิด chamber และทำการดูดอากาศภายใน chamber ตามขั้นตอนที่แสดงไว้ในหัวข้อที่ 2

5.7 เมื่อความดันภายในลดลงถึง 10^{-6} torr. ให้เปิดกุญแจที่เป็นสวิทช์ของแหล่งจ่ายไฟ E/B จากตำแหน่ง offไปยังตำแหน่ง on ถ้าหากระบบ interlock ทั้งหมดถูกต้อง HV ที่ตำแหน่ง off จะมีไฟสีเขียวติด

5.8 ตรวจดูว่าปุ่มที่ใช้ควบคุมลำอิเล็กตรอนปรับอยู่ที่ 0 แล้วทำการกดสวิทช์HV ให้อยู่ในตำแหน่ง on หลอดไฟสีแดงจะติด และ E/B gun filament จะเริ่มทำงาน

5.9 ค่อยๆ ปรับ potentiometer ที่อยู่บน remote controller ให้มีค่าเพิ่มขึ้นทีละน้อย และจะต้องปรับให้ลำอิเล็กตรอนตกอยู่ที่ศูนย์กลางของตัวอย่าง(ตัวระเหย)

5.10 ปิด shutter หลังจากที่ทำการ deposit เสร็จแล้ว

5.11 ปรับปุ่มควบคุมลำอิเล็กตรอนไปที่ 0 และกดสวิทช์ HV ไปที่ตำแหน่งoff

5.12 รอให้แผ่นฐานเย็นตัวลง

5.13 เมื่ออุณหภูมิภายใน chamber เย็นตัวลง ปิดสวิทช์ filament ของ vacuum guage และปล่อยอากาศให้เข้าไปใน chamber

ภาคผนวก ค

การใส่ค่าพารามิเตอร์ของ CRTM FILM THICKNESS MONITOR

บทนำ

CRTM film thickness monitor ออกแบบเพื่อใช้ในการวัด และแสดงความหนาของฟิล์ม ที่ deposit ด้วยระบบสุญญากาศ

ความหนาของฟิล์มคำนวณได้จาก ความหนาแน่นของวัสดุที่ใช้เป็น evaporant และอัตราส่วนระหว่าง ค่า acoustic impedance ของ crystal และ evaporant ของวัสดุนั้นๆ (Z-ratio) ค่าที่แสดงมีความถูกต้องสูงกว่าความถี่ย่านกว้าง

มี setpoint ที่ตั้งค่าได้อยู่ 3 ค่า เพื่อใช้ควบคุมความหนาของฟิล์ม และมี relay contact output ของ setpoint แต่ละตัวแยกจากกัน

การใส่ค่าพารามิเตอร์ (input parameter)

CRTM มีค่าพารามิเตอร์ที่จะต้องใส่เข้าไปมีอยู่ 3 ค่าคือ ความหนาแน่น (density), Z-Ratio และ Tooling เพื่อใช้ในการคำนวณหาความหนาของฟิล์ม ส่วนพารามิเตอร์ที่ตั้งค่าได้มีอยู่ 3 ค่า คือ THK1, THK2 และ TIME LIMIT เพื่อใช้ในการควบคุมความหนาของฟิล์ม

1. ความหนาแน่น (density: g/cm^3)

ค่าพารามิเตอร์ความหนาแน่นนี้ จะเป็นตัวบ่งบอกค่าความหนาแน่นของวัสดุที่ใช้เป็นตัวระเหย (evaporation) และ ค่าความหนาแน่นจะมีหน่วยเป็น g/cm^3 ตารางที่ ค.1 แสดงค่าความหนาแน่นของสารที่ใช้

2. z-ratio

ค่า acoustic impedance เป็นอัตราส่วนระหว่างผลึก (crystal) ของ sensor และวัสดุที่ใช้เป็นตัวระเหย ตารางที่ ค.1 แสดงค่า z-ratio ของสารที่ใช้

Substance	Density (g/cm³)	z-ratio	Substance	Density (g/cm³)	z-ratio
Al	2.70	1.08	Mo	10.2	0.257
Sb	6.62	0.768	Ni	8.91	0.331
As	5.73	0.966	Nb	8.57	0.493
Be	1.85	0.543	Pd	12.0	0.357
B	2.54	0.389	Pt	21.4	0.245
Cd	8.64	0.682	KeI	1.98	2.05
CdS	4.83	1.02	Se	4.82	0.864
CdTe	5.85	0.980	Si	2.32	0.712
CaF_2	3.18	0.775	SiO_2	2.20	1.07
C	2.25	3.26	Ag	10.5	0.529
Cr	7.20	0.305	AgBr	6.47	1.18
Co	8.71	0.343	AgCl	5.56	1.32
Cu	8.93	0.437	NaCl	2.17	1.57
Ga	5.93	0.593	Ta	16.6	0.262
GaAs	5.31	1.59	Te	6.25	0.900
Ge	5.35	0.516	Sn	7.30	0.724
Au	19.3	0.381	Ti	4.50	0.628
In	7.30	0.841	W	19.3	0.163
InSb	5.76	0.769	WC	15.6	0.151
Ir	22.4	0.129	U	18.7	0.238
Fe	7.86	0.349	V	5.96	0.530
Pb	11.3	1.13	Y	4.34	0.835
PbS	7.50	0.566	Zn	7.04	0.514
LiF	2.64	0.774	ZnO	5.61	0.556
Mg	1.74	1.61	ZnSe	5.26	0.722
MgO	3.58	0.441	ZnS	4.09	0.775
Mn	7.20	0.377			

ตารางที่ ค.1 ค่า density และ acoustic impedance ของวัสดุชนิดต่างๆ

3. tooling %

แฟคเตอร์นี้ใช้เพื่อเป็นตัวแก้ความถูกต้องที่แตกต่างกัน ระหว่างความหนาของฟิลม์ที่ตำแหน่งแผ่นฐาน และฟิล์มที่ปลูกบนผิวของผลึก (crystal) ของ sensor ค่า tooling

factor คือ

$$\text{Tooling} = \frac{\text{ความหนาของฟิล์มที่ตำแหน่งแผ่นฐาน} * 100}{\text{ความหนาของฟิล์มที่ตำแหน่ง sensor}}$$

ถ้าจำนวนของวัสดุที่ปลูกบนแผ่นฐาน (substrate) และเซนเซอร์ (sensor) มีจำนวนเท่ากันค่า tooling factor จะมีค่า 100 % ค่าความถูกต้องของค่า tooling factor จะหาได้จากการทดลอง โดยวาง monitor substrate ไว้ที่ตำแหน่งแผ่นฐาน(substrate) แล้วทำการวัดค่าความหนาของฟิล์มที่ปลูกบน monitor substrate และนำมาเปรียบเทียบกับความหนาที่แสดงบน CRTM ค่าความถูกต้องของ tooling factor จะหาได้จากสูตรต่อไปนี้

$$\text{Tooling} = \frac{\text{ค่าที่วัดได้จริงบนแผ่นฐาน} * 100}{\text{ค่าที่แสดงบน CRTM}}$$

4. thk1, thk2

พารามิเตอร์ตัวนี้เป็นการตั้งค่าความหนาของฟิล์มที่ต้องการ เพื่อใช้ในการควบคุมความหนาของฟิล์ม ค่าพารามิเตอร์ทั้งสองจะมีหน่วยเป็น K°A.

5. time limit

การตั้งค่าพารามิเตอร์ เพื่อใช้ในการควบคุมความหนาของฟิล์ม โดยการตั้งค่าเวลานี้จะใส่ค่าอยู่ในหน่วยของนาที และวินาที

ประวัติผู้เขียน

นายนารเมธ นานานุกูล เกิดวันที่ 3 พฤษภาคม พ.ศ.2513 ที่อำเภอเมือง จังหวัดนนทบุรี สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมควบคุม จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2533 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2534