Thermochimica Acta, 52 (1982) 305–315
Elsevier Scientific Publishing Company, Amsterdam—Printed in The Netherlands

A MICROPROCESSOR-BASED EVOLVED GAS ANALYSIS SYSTEM

WALTER R. JONES

Bell Laboratories, Murray Hill, NJ 07974 (U.S.A.)

(Received 22 July 1981)

ABSTRACT

A microprocessor based evolved gas analysis system in which measurements could be performed in an external magnetic field was designed and constructed. The system consisted of a microprocessor-based furnace controller which could be programmed to provide various controlled temperature cycles and a microcomputer which performed data acquisition. The details (i.e. configuration, hardware, software) of the system are discussed.

INTRODUCTION

It has been reported that the reaction rate of some materials is greatly affected when the material is subjected to an external magnetic field [1–3]. Using thermogravimetry (TG) to study these magnetic materials in an external magnetic field poses obvious difficulties, both in performing the measurement and interpreting the results. These difficulties arise from apparent weight changes produced by the presence of a field gradient and changes in the magnetic moment of the sample. This problem, however, does not arise with evolved gas analysis (EGA). Recently, it has been shown that in the absence of a magnetic field with its accompanying difficulties, results obtained from TG and EGA measurements are the same [4]. Therefore, an EGA system was constructed to study the reaction rates of these materials in the presence of a magnetic field.

Since the controlled temperature cycles of the samples can range from a very fast heating rate to isothermal heating, it would be ideal to construct a system which would be under computer control. The computer would be in charge of controlling the temperature of the furnace and data acquisition. Such a system has been constructed. This automated EGA system allows us to perform measurements in an external magnetic field while the entire measurement process is under the control of dual microprocessors. One microprocessor controls the temperature cycle while the other performs the data acquisition. The purpose of this paper is to describe in detail the hardware and software used in an automated evolved gas analysis system.

0040-6031/82/0000–0000/$02.75

DESCRIPTION AND TESTING OF THE SYSTEM

This particular EGA system was designed to measure the amount of water vapor given off by a sample undergoing a controlled temperature cycle. A schematic of the complete EGA system is shown in Fig 1. The system consists of an aluminum sample chamber connected to an inert or an active gas source, a hydrometer which measures the amount of water vapor given off by the sample, a programmable microprocessor-based furnace controller which controls the furnace temperature, and an LSI-11 microcomputer which performs the data acquisition. The design of the sample chamber was chosen so as to minimize the volume between the sample and sensor. It consisted of two arms joined by a middle section and the entire sample chamber was mounted between the pole faces of a 4 in. magnet. The sample is put in a quartz bowl which hangs in a Perkin-Elmer Model 319-0252 furnace. The furnace is then inserted into the middle section of the sample chamber. The input for the gas stream is provided through arm A. At the base of this arm is a sensor probe which is connected to a Panametrics model 3000 hydrometer. Similarly, arm B provides the output for the gas flow and also contains a sensor probe connected to the model 3000 at its base. The gas flows into the input of arm A through the middle section over the sample and out of the output of arm B Sensor 1 can be used to monitor the water vapor pressure of the input gas stream while sensor 2 is used to measure the water vapor pressure of the output gas. The Panametrics model 3000 hydrometer measures water vapor pressure and produces a voltage output from 0–1 V proportional to a dew/frost point temperature range of 20–80°C.

The furnace temperature is controlled by a Perkin-Elmer System 4 microprocessor -based controller. It can be programmed via a front panel keyboard to provide a wide range of temperature cycles ranging from a simple heat and cool cycle to a

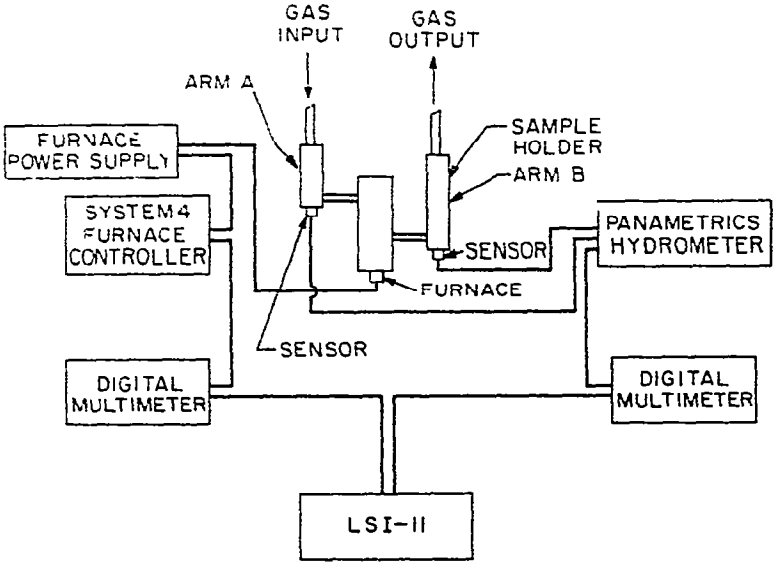

Fig 1 Schematic of the complete evolved gas analysis system

multi-step heat and hold cycle. This microprocessor-based furnace controller was used since conventional furnace controllers require constructing temperature programs on cards (e.g. Ransco Industries Model 5300 Data Trak card programmer). The System 4 can display the programmed temperature or the actual temperature, which is measured by a thermocouple placed in the furnace near the sample. The System 4 also provides an output voltage proportional to the actual temperature. The voltages provided by the Model 3000 hydrometer and the System 4 are read by two Hewlett Packard 3455A digital multimeters. These multimeters are connected to an LSI-11 microcomputer via an IEEE-488 bus. The LSI-11 acts as a satellite processor of a Digital Equipment Corp. PDP 11/45 computer. The details of this configuration can be found elsewhere [5]. This microcomputer performs the data acquisition Its memory capacity allows us to store 512 floating point data points but, for temperature cycles lasting on the order of days, this capacity is not sufficient. Therefore, one has to either send the data to the 11/45 when the memory is filled or use external auxiliary data storage. Caution, however, must be used if the data is to be sent to the 11/45. A system crash on the 11/45 would result in the loss of all the accumulated data, even a momentary crash once every three days would, on average, make it impossible to run a three-day temperature cycle. Since a system crash cannot be predicted and to avoid the loss of data, a digital cassette recorder was used for external auxiliary data storage. It was less costly and much easier to interface to our system than a floppy disk. A copy of the program which reads the multimeter and transfers the data to the cassette recorder is shown in Appendix A. The program is written in C language. Since the heating rate can be different for each sample, the program prompts the user for a sampling rate. This version of the program assumes the System 4 is programmed for a simple linear heating rate, but it can be modified if the System 4 is programmed for a non-linear heating rate. The program also asks the user for the number of data points to average and the length of the temperature cycle in minutes. Since file space on the 11/45 is expensive, the data is averaged over some time interval which is determined by the number of points to be averaged and the sampling rate. This allows us to record the maximum number of data points possible without generating large expensive data files. The time parameter controls how long the LSI is required to record data. This time is usually set from the time it takes the System 4 to complete its programmed temperature cycle. The program is also capable of error detection. Sometimes erroneous data points are generated by external noise or if the LSI-11 misreads the multimeters. Even though these events rarely occur, they sometimes do happen. The program tests to see whether each data point is zero and it also compares each data point to the previous data point taken If the difference between them is greater than some arbitrary limit set by the user, then the data point is a "bad" one. Otherwise, it is considered a valid data point. Zero data points are also considered "bad" data points. The limit, which is set by the user, is determined from the maximum heating rate and the slowest sampling rate to be used in a temperature cycle. When "bad" data points are detected, they are discarded and a count is kept of the number of errors detected during a complete temperature cycle. Since this system has been operational, the number of errors

detected during a temperature cycle has been less than 0.1%. The subroutine SEND TAPE is responsible for the transfer of the data from the LSI's memory to the cassette tape. Using the microcomputer in this mode allows us to perform real-time data acquisition while not being inhibited by the limited memory capacity of the microcomputer and system crashes. However, once the memory is filled, some of the data is lost during the data transfer to the cassette since the LSI-11 cannot record and transfer data simultaneously. Usually, the LSI-11 takes about 11 sec to transfer 512 data points to cassette tape. The fastest heating rate for the System 4 is 200°C min^{-1}. At a sampling rate of one data pair per sec, there is about a 5% data loss in a temperature cycle from room temperature to 700°C. This loss can be tolerated in most cases and is an extreme case. Obviously, for slower rates the data loss will be less. Once the temperature cycle is completed, the data on the cassette is transferred via the LSI-11 (see Appendix B) to the PDP 11/45 where data analysis can be performed.

Once assembled, the thermal decomposition of $Ba(OH)_2 \cdot H_2O$ was investigated. This provided a means to test the apparatus since similar work was done on this compound using simultaneous thermogravimetry–evolved gas analysis [4]. The temperature program consisted of heating from room temperature at 1°C min^{-1} to

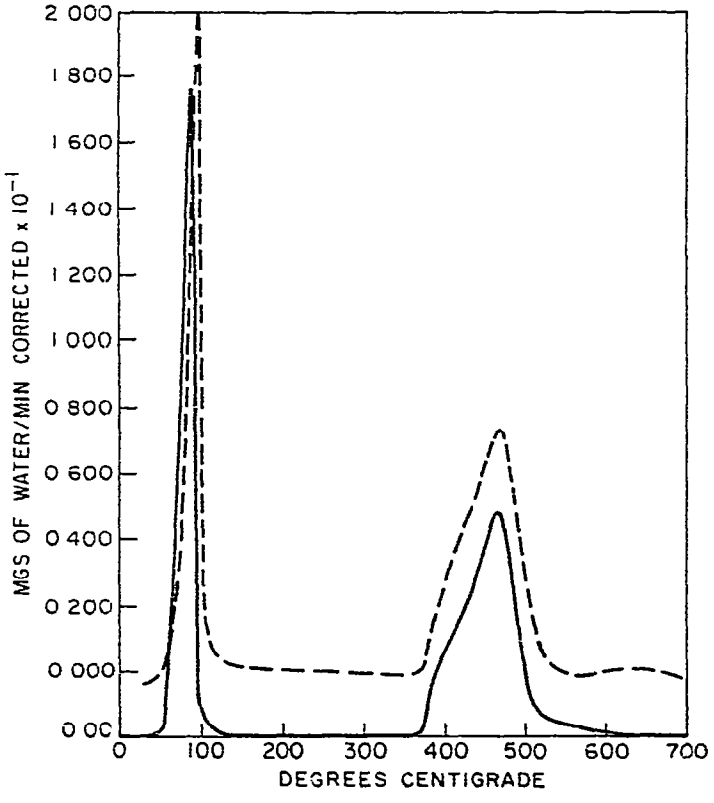

Fig 2 Comparison of the data obtained from ref 4 (solid curve) with that from the microprocessor-controlled EGA system (broken curve)

700°C, holding for 2 min and then cooling H_2 gas was flowed through the system at 300 cm^3 min^{-1}. The total time for this temperature cycle was 700 min. The data were recorded at a sampling rate of once every two seconds while averaging 10 data points. The results are shown in Fig. 2 along with data obtained from ref. 4. Note the excellent agreement. Both sets of data are the corrected curves in which the background moisture in the gas flow was subtracted out [4]. The data obtained from ref. 4 is displaced from zero for clarity.

CONCLUSION

An automated evolved gas analysis system was constructed to study the reaction rates of materials in an external magnetic field. The system utilizes a microprocessor-based furnace controller and data acquisition system. The results from measurements on $Ba(OH)_2 \cdot H_2O$ obtained from this system showed excellent agreement with the previous results of ref. 4. This system has the flexibility to offer a wide variety of temperature programs and data acquisition rates. Since this system is a complete controlled temperature and data acquisition package, it can be easily adapted for other types of thermal analysis measurements. Future plans are to modify this system to be able to measure the magnetic susceptibility as a function of temperature using the Faraday method. The System 4/LSI-11 configuration will remain the same. The hydrometer, however, will be replaced by a Cahn balance. The sample will be suspended from the Cahn balance between the pole faces of the 4 in. magnet. A magnetic field gradient will be produced by coils mounted to the magnets pole faces. The Cahn balance will then be used to detect the apparent weight of the sample produced by the magnetic field gradient. As the sample undergoes a temperature cycle, the apparent weight change will be measured.

ACKNOWLEDGEMENT

The author would like to thank E.M. Gyorgy and P.K. Gallagher for their encouragement and many helpful criticisms and suggestions.

REFERENCES

1 G S Krinchik, R M Shavartsman and A Y. Kepnis, JETP Lett , 19 (1974) 231
2 M W Rowe, S M Lake and R Faneck, Nature (London), 266 (1977) 612
3 M W. Rowe, D.A. Edgerley, M Hyman and S M. Lake, J. Mater Sci , 14 (1979) 999
4 P K Gallagher and E.M Gyorgy, in Vol. I, Thermal Analysis, H G. Wiedemann (Ed), Birkhauser Verlag, Boston, 1980, pp 113–118
5 S Murrel and T Kowalski, Behav Res Methods Instrum , 12(2) (1980) 126

APPENDIX A

EGA.C

```
# include <math.h>
# include <mini.h>
# include <stdio.h>
# define asize 25/*array size*/
# define bell 007
# define wsize 4/*bytes per data pt*/
run()
{
        double fabs(),reed();
        double d2t[asize],pdata;
        FILE *fopen(),*fp;
        float sendtape(),sumd1,sumd2t,d1[asize];
        int cnt,tau,i,l,k,pave;
        int rl,nblk;
        static long int tnpts;
        long int npts,nerror;
start:
        nerror = tnpts = 0;
        bfree(0);
        printf("IS GAS FLOW ON???\n");
        tau = number("sample rate?"); /*prompts user for sample rate*/
        setsync((int)(tau*SEC));
repeat:
        pave = number("number of pts to average? 25 max");
        if(pave = = 0)/*prompts for number of points to average*/
        {
                printf("Has to be integer greater than zero\n");
                goto repeat;
        }
        printf("number of pts to ave = %d\n",pave);
        rl = number("length of exp run in mins"); /*prompts for length*/
        nblk = (rl*15)/(64*tau*pave); /*of temperature cycle*/
        if(nblk = = 0)
        {
                printf("INVALID BLOCK COUNT!!!\n");
                printf("Runlength to short for sample rate");
                printf("and number of\ n points to average");
                printf("Reevaluate and reenter\n");
                goto start;
        }
```

```
printf("%d blocks needed\n",nblk);
write(30,cmd("?1"),2); /*initializes multimeters*/
write(30,cmd("2"),1);
write(30,"F1R7T1D0H0A0",12);
write(30,cmd("?"),1);

/*take data*/

rsvp("hit return to start");
cnt = 1;
write(30,cmd("?2"),2);
write(30,"T3",2);
write(30,"T3",2);
write(30,cmd("?R>"),3);
pdata = reed();
write(30,cmd("?1"),2);
write(30,"T3",2);
while(cnt <= nblk)
{
      i = 0;
      for(k = 0;k < pave;k++)
      {
            sync();
            write(30,cmd("?1"),2);
            write(30,"T3",2);
            write(30,cmd("?Q>"),3);
            d1[i] = reed();
            write(30,cmd("?2"),2);
            write(30,"T3",2);
            write(30,cmd("?R>"),3);
            d2t[i] = reed();
            if(fabs(pdata-d2t[i])>.004|d2t[i]==0 0)/*error detection*/
                  ++nerror;
            else
            {
                  pdata = d2t[i];
                  ++i;
            }
      }/* end of for loop*/

      /*average the data*/

      sumd1 = 0.0;
      sumd2t = 0.0;
```

```
        if(i==0)
                goto skip;
        for(l=0;l<i;l++)
        {
                sumd1 = sumd1 + d1[1];
                sumd2t = sumd2t + d2t[1];
        }
        if(bput(0,sumd2t/i)==-1)/*store in memory if memory*/
        {               /*full store on tape*/
                npts = sendtape();
                tnpts = tnpts + npts;
                setsync((int)(tau*SEC));
                printf("pts written %ld total pts written %ld\n",npts,tnpts);
                ++cnt;
                bfree(0);
                bput(0,sumd2t/i);
        }
        if(bput(0,sumd1/i)==-1)
        {
                npts = sendtape();
                tnpts = tnpts + npts;
                setsync((int)(tau*SEC));
                printf("pts written %ld total pts written %ld\n",npts,tnpts);
                ++cnt,
                bfree(0);
                bput(0,sumd1/i);
        }
skip:
        if(bput(0)%100==0)
        {
                putchar(bell);
                printf("100 pts put in buffer\n");
        }
        if(getvolt(10)/VOLTS>2.0)
        {
                npts = sendtape();
                tnpts = tnpts + npts;
                printf("pts written %ld total pts written %ld\n",npts,tnpts);
                fp = fopen("tdpw","w");
                fprintf(fp,"%ld\n",tnpts);
                fclose(fp);
                printf("errors: %ld\n",nerror);
                bfree(0);
                goto start;
```

```
            }
            write(30,cmd("?2"),2);
            write(30,"T3",2);
            write(30,cmd("?R>"),3);
            pdata = reed();
        }

        /*end of while loop*/

        fp = fopen("tdpw","w");
        fprint(fp,"%ld\n",tnpts);
        fclose(fp);
        printf("errors: %ld\n",nerror);
        printf("data acquisition finished\n");
}

/*subroutine sendtape*/

float sendtape()
{
        int npt,1,cnt = 0,temp;
        float data[4];
        npt = bput(0);
        setsync((int)(1*SEC));
        startape();
        delay(4);
        for(1 = 1;1 < npt;1 + = 4)
        {
                data[0] = bgetf(0,1);
                data[1] = bgetf(0,1 + 1);
                data[2] = bgetf(0,1 + 2);
                data[3] = bgetf(0,1 + 3);
                temp = wtape(data,16);
                cnt = cnt + temp;
        }
        offtape();
        return(cnt/wsize);
}
```

APPENDIX B

GETAPE.C

```
#include "wtape.h"
#include <mini.h>
#include <stdio.h>
#define wsize 4/*bytes per number*/
#define nb 16/*bytes transferred*/
run()
{
        register dlv *tape = addr;
        FILE *fopen(),*fp;
        long int ndp,npt = 0;
        float data[4];
        int cnt,z,temp,trash,l;
        fp = fopen("tdpw","r");
        fscanf(fp,"%ld",&ndp);
        printf("name of datafile\n");/*prompts user for name of data file*/
        bname();
        bname(0,"-tape");
        rsvp("hit return to rewind tape");/*allows user to rewind and */
        startape();          /*position tape to read data */
        rsvp("hit return to stop tape");
        offtape(),
        rsvp("hit return to read tape");
        for(z = 1;z <= ndp/512;++z)
        {
                cnt = 0;
                startape();
                trash = tape-> rbuf&0377;
                for(l = 1;l < 512;l += 4)
                {
                        temp = rdtape(data,nb);
                        bput(0,data[0]);
                        bput(0,data[1]);
                        bput(0,data[2]);
                        bput(0,data[3]);
                        cnt = cnt + temp;
                }
                offtape();
                printf("%d pts read\n",cnt/wsize);
                bsend(0);
                printf("data sent to 11/45\n");
```

```
        bfree(0);
        npt = npt + cnt/wsize;
}
printf("total pts read = %ld\n",npt);
```