《研究論文》

Journal of the Human Engineering Society of Korea
Vol.9 No. 1, June 1990

IBM PC를 이용한 심장 박동 간격의 측정

(Heart Beat Interval Measurement using an IBM PC)

李東夏† 朴景洙††

Abstract

This article develops a cost-effective and accurate measurement system for heart best intervals. The system is composed of an analog to digital (A/D) converter, an IBM personal computer (an 8088 microprocessor, an 8253-5 timer, an 8259A interrupt controller, and memories) and assembler programs for controlling these hardware components. An exponential smoothing algorithm effectively reduced noise effects from A/D converted electrocardiogram (ECG) signals influenced by 60 Hz alternating current (AC). The system can collect 15000 heart beat intervals with an 1/5400 second unit.

1. 서론

육체부하(physical load)나 정신부하(mental load)를 평가하려는 연구에 있어서 많이 쓰이는 생리 지수(physiological index)인 심박수 (heart rate)와 부정맥(sinus arrhythmia) 점수는 심장 박동 간격(heart beat interval)으로부터 구해진다. 심장 박동 간격은 심장 박동에 기인하는 생체 전기 신호인 심전도(electrocardiogram;ECG)의 R파와 R파 사이의 시간 간격(줄여서, R-R 구간이라고 함)으로 대표된다[Fig.1]

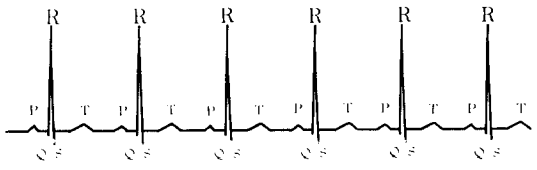

Fig 1. Electrocardiogran (ECG).

Apple Ⅱ PC에 비하여 값은 다소 비싸지만 현재 학계에서 가장 많이 사용되는 IBM PC를 이용한 R-R 구간 측정 체계는 부정맥 연구를 널리 보급하는 데 기여할 수 있다. 또한 확장된 memory 용량과 빠른 연산 속도로 인하여 Applle II PC를 이용한 체계[4]에 비하여 더 많은 양의 R-R구간 자료를 더 높은 정밀도로 수집할 수 있는 장점이 있다.

2. 체계의 구성

측정 체계는 (1) 생체 신호 증폭기 (bioelectric amplifier)를 통해 여파(filtering)되고 증폭된 analog 심전도 신호를, 컴퓨터가 읽을 수 잇는 형태의 digital 신호로 변환하는 A/D 변환기(Analog/Digital converter), (2) A/D 변환을 지시하고, 60Hz 주파수를 갖는 교류의 영향을 제거하기 위해 digtal 신호에 대한 평활(smoothing)과정을 수행하고, R파의 첨두(peak)를 탐지하고, 각 R파 첨두 사이의 시간을 측

† 수원대학교 산업공학과
†† 한국과학기술원 산업공학과

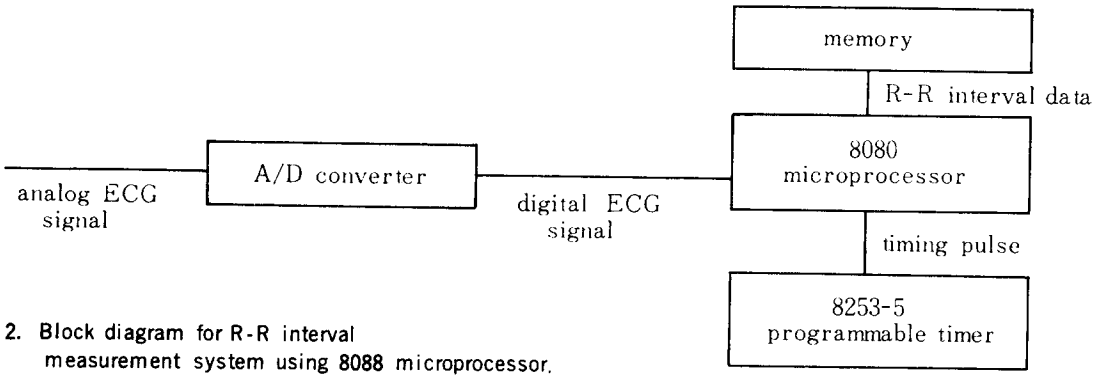

Fig 2. Block diagram for R-R interval measurement system using 8088 microprocessor.

정하는 intel 8088(또는 80286) microprocessor, (3) 일정한 주파수로 발진되는 timing 신호를 이용하여 microprocessor의 작동을 조정함으로써 측정 시간의 정밀도를 18.206 Hz-5400 Hz 범위에서 유지할 수 있게 하는 intel 8253-5 programmable timer, 및 (4) 측정된 R-R 구간 자료를 저장하는 memory로 구성된다. memory에 기록된 자료는 바로 분석에 이용되거나, disktte에 file로 저장될 수 있다. 본 연구에서는 KSE-8627 A/D 변환기, IBM PC에 내장된 8088 microprocessor, intel 8253-5 programmble timer, 및 memory로 측정 체계를 구성하였다.

3. R 첨두 탐지 원리

KSE-8227 A/D 변환기는 -5 volt에서 5volt 사이의 analog 신호를 0에서 4095 사이의 digital 신호로 선형변환(linear transformation)한다. 이러한 A/D 변환기의 특성에 맞추기 위하여 심전도 신호의 최대치와 최소치를 -5에서 5volt 사이에 오도록 생체 신호 증폭기의 위치 조정 스위치(position control switch)와 증폭 스위치를 이용하여 조정한다.

변환된 신호에는 교류에 의한 60Hz 주파수의 잡음 신호(noise signal)가 섞여 있으므로 이동 평균(moving average)이나 지수 평활(exponential smoothing) 방법을 사용하여 신호를 평활하는 것이 바람직하다. 각 방법에 의한 연산 과정을 평가한 결과 지수 평활법이 이동평균법에 비하여 평활 효과도 더 좋고 평활에 소요되는 시간도 덜 걸리는 것으로 나타났기 때문에 본 연구에서는 지수 평활법을 사용하였다.

심전도 신호 [Fig.1]의 각 R파 첨두 높이는 서로 약간씩 다르기 때문에 일률적인 기준에 의하여 정확한 R파 첨투 발생 시점을 탐지하기 어렵다. 가장 단순하면서도 비교적 정확한 방법은 평균적인 R첨두의 높이를 5volt 미만(대략 4volt 정도)이 되도록 증폭한 후 P파나 T파의 첨두가 도달하지 못하고 단지 R파의 첨두 만이 도달할 수 있는 적당한 수준(대략 3.6volt, digital로 변환된 값은 2457정도)을 R파 첨두 탐지를 위한 기준으로 정하고 digtal로 변환된 심전도 신호가 이 값보다 커지는 순간을 R파가 첨두에 도달하는 시점으로 대신하는 것이다.

4. 8088 microprocessor를 이용한 R-R 구간 계측 원리

컴퓨터 프로그램밍 언어 중에서 assembler는 microprocessor 통계에 편리하고 고속 처리되는 장점이 있다. 따라서 microprocessor에 의해 고속 처리해야하는, 심전도 신호에 대한 A/D 변환, 지수 평활에 의한 잡음 신호(noise signal) 제거, R 첨두 탐지, R-R 구간 측정, 및 8253-5 timer 조종 프로그램 작성에 알맞다. 그러나 assembler는 일반 사용자에게 익숙한 언어가 아니므로 사용자가 직접 처리해야하는 사항들(예를 들면, 측정할 R-R 구간 갯수, 측정 정밀도, 지수 평활 계수(exponential smoothing coefficient), 및 자료를 저장할 file 이름의 지정, 또는 측정된 R-R 구간 자료의 분석, 등)은 사용자에게 익숙한 BASIC 프로그래밍 언어로 작성하여, assembler로 작성된 R-R 구간 측정 routine과 결합하여 사용하도록 하였다.

Fig.3은 BASIC으로 작성된 전산 프로그램을 보여준다. 사용자로부터 측정하고자 하는 R-R 구간 자료의 갯수를 변수 NO로 받아들인다(20번 문장). 연산 시간에 대한 평가 결과 BASIC과 결합된 assembler routine의 A/D 변환속도 범위는 19-3300Hz 였다. 따라서 R-R구간 자료의 정밀도를 결정하는 초(sec) 당 A/D변환 횟수는 [19, 3300] 범위에서 선택할 수 있고, 이 값은 변수 HZ에 저장된다.(30번 문장). 지수 평활 과정에는 새로운 자료의 반영 정도를 나타내는 [0, 1] 사이의 지수 평활 계수의 지정이 요구된다. microprocessor는 지수 평활 계수 중 특히 $1/(2^N)$ (N=0,1,2,.....)을 빨리 처리할 수 있는 기능을

```
10 DEFINT A-Z
20 INPUT "Number of R-R intervals to collect (1 - 15000) : ";NO
30 INPUT "Frequency of A/D conversion (19 - 3300Hz) : ";HZ
40 INPUT "For a exponential smoothing coefficient 1/(2^N), type N : ";N
50 INPUT "File name : ";NAM$
60 DIM X(NO) 'data buffer of BASIC
70 DEF SEG=&H9620 'segment address of assembly routine
80 BLOAD "rrm.bin" 'loading of assembly routine
90 RRM=&HO 'offset address of assembly routine
100 CLS
110 LOCATE 10,30:PRINT "now a/d conversion"
120 CALL RRM (NR,NO,HZ,N,X(0)) 'R-R interval measurement by assembly routine
130 'NR = Number of R-R intervals measured by assembly routine
140 CLS
150 LOCATE 10,30:PRINT "end of a/d conversion"
160 OPEN "O",#1,NAM$
170 FOR I=0 TO NR-1
180 WRITE #1, INT(X(I)*1000/HZ+.5) 'saving R-R intervals in msec
190 NEXT I
200 CLOSE #1
210 END
```

Fig 3. BASIC Program part for R-R interval measurement.

가지고 있으므로, 본 프로그램에서는 정수 N에 의하여 평할 계수 $1/(2^N)$을 선택할 수 있도록 하였다(40번 문장). 예를 들면 N=5로써 지수 평할 계수 $1/(2^5)=.031$을 선택할 수 있다. $1/(2^N)$의 계수를 사용한 지수 평할은 $2^{N+1}-1$ 개의 자료에 대한 이동 평균을 사용한 경우와 같은 평할 효과를 가지며 [3], 이동 평균의 경우 N의 증가에 따라 연산에 소요되는 시간이 기하 급수적으로 증가하지만 지수 평할은 N에 관계 없이 연산 시간이 거의 일정하다는 장점이 있다. 심전도 신호처리에 있어서는 N=7 즉 $1/2^7=.0078$의 지수평활 계수에 의하여 교류에 의한 60hz 주파수의 잡음신호를 제거할 수 있다.

R-R 구간자료를 저장할 BASIC data buffer를 60번 문장 [Fig.3]으로 확보한다. BASIC program과 결합시키고자 하는 assembler routine을 memory 상에 load 한다(80번 문장)를 지정한다. "RRM"이라는 이름의 assembler routine을 BASIC에서 call 할 때(120번 문장) 측정할 R-R 구간 자료의 갯수, A/D변화 주파수, 지수 평할 계수를 지정하는 값, 첫번째 R-R 구간 자료를 저장할 BASIC data buffer의 번지수를 assembler routine에 보내고, assembler routine에 의하여 측정된 R-R 구간 자료의 갯수 및 R-R 구간 자료를 BASIC data buffer로 받아들인다. BASIC data buffer에 기록된 R-R 구간 자료는 바로 분석에 이용하거나 160-200번 문장으로 diskette에 기록하여 보관한다.

Fig.4는 R-R 구간을 측정하는 assembler program을 보여준다. main routine(13-146번 문장)에서 microprocessor는 먼저 BASIC program을 통해 입력된, 측정할 R-R 구간의 갯수, A/D변환 주파수, 지수 평할 계수를 지정하는 값을 assembler program에서 사용할 변수 no_data , freg, s_c에 각각 옮겨 놓는다(13-28번 문장). R-R 간격은 보통 0.5초 이상이므로 R 첨두가 탐지 될 때마다 이전의 R 첨두와의 시간 간격이 0.5초 이상이 되는지 확인할 필요가 있다. 만일 0.5초 이내이면 탐지된 R 첨두를 잡음신호로 간주하는 것이 바람직하다. 본 프로그램에서는 0.5초의 시간 간격에 해당하는, A/D변환 주파수의 절반을 latency 라는 변수에 저장하여(23-24번 문장) A/D 변환시마다 하나씩 증가되는 R-R구간 counter의 숫자와 비교함으로써 R-R간격이 0.5초 이상이 되는가를 확인하였다. 8253-5 timer로 원하는 주파수의 timing 신호를 발생 시키기 위하여 8253-5 timer chip의 기본 주파수인 1,193,180Hz를 원하는 주파수로 나누어 그 몫을 timer의 입력 port에 입력 시켜야한다 [1]. 이와같은 timer의 특성에 맞추기 위하여 1,193,180을 변수 freq에 저장된 주파수로 나누고 몫을 다시 freq에 저장하여 (37-45번 문장) timer의 입

```
*
title        R-R Interval Measurement
;
code segment     para
assume cs:code,ds:code
rrm        proc far   ;define assembly routine procedure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;MAIN ROUTINE PERFORMED BY BASIC CALL;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;;;;;;;;Transfer BASIC data to assembly routine;;;;;;;;;;;
;
;transfer number of R-R intervals to no_data
     push    bp
     mov     bp,sp
     mov     bx,[bp+12]
     mov     ax,[bx]
     mov     no_data,ax
;transfer frequency of A/D conversion to freq
     mov     bx,[bp+10]
     mov     ax,[bx]
     mov     freq,ax
;establish a criterion (minimum R-R interval=latency) to detect R peak
     shr     ax,1
     mov     latency,ax
;transfer a exponential smoothing coeff. to s_c
     mov     bx,[bp+8]
     mov     ax,[bx]
     mov     s_c,ax
;transform number of R-R intervals into a suitable form for
;assembly routine
     mov     ax,no_data
     shl     ax,1
     sub     ax,2
     mov     no_data,ax
;transform A/D conversion frequency into a suitable form for
;8253-5 timer chip
     mov     dx,12h
     mov     ax,34dch
     div     freq
     shr     freq,1
     cmp     dx,freq
     jl      roundoff
     inc     ax
roundoff:
     mov     freq,ax
;
;;;;;;;;Assign segment and offset addresses of timing pulse routine
;;;;;;;;in which 8088 microprocessor should branch out at each
;;;;;;;;timing pulse from a 8253-5 timer chip
;
     push    ds
     mov     ax,0
     mov     ds,ax
     mov     si,112  ;memory address for location of timing pulse
                     ;routine
     mov     cx,word ptr [si]    ;offset of old timing pulse routine
     mov     dx,word ptr [si+2]  ;segment of old timing pulse routine
     mov     ax,offset adcon     ;offset of new timing pulse routine
     mov     word ptr[si],ax     ;set offset of new
```

Fig 4. Assembler program part for R-R interval measurment.

```
61:       mov     ax,seg adcon       ;segment of new timing pulse routine
62:       mov     word ptr[si+2],ax  ;set segment of new
63: ;
64:       pop     ds
65:       mov     mm1,cx  ;save offset of old timing pulse routine
66:       mov     mm2,dx  ;save segment of old timing pulse routine
67: ;
68: ;;;;;;;;Prepare data for timing pulse routine by changing data segment
69: ;;;;;;;;into a suitable form for timing pulse routine
70: ;
71:       mov     ax,latency
72:       mov     bx,s_c
73:       push    ds
74:       push    cs
75:       pop     ds
76:       mov     latency,ax  ;minimum R-R interval
77:       mov     s_c,bx      ;exponential smoothing coeff.
78:       mov     si,0    ;initialize R-R interval counter
79:       mov     bx,0    ;initialize memory address indicator
80:       mov     m1,500  ;initialize forecasted value
81:       pop     ds
82: ;
83: ;;;;;;;;Set 8253-5 timer chip by input frequency of A/D conversion
84: ;
85:       mov     al,00110110b    ;timer setting data
86:       out     43h,al      ;port address for timer setting data
87:       mov     ax,freq     ;input timing pulse frequency
88:       out     40h,al      ;set lower 8 bit of timing pulse frequency
89:       mov     al,ah
90:       out     40h,al      ;set higher 8 bit of timing pulse frequency
91: ;
92: ;;;;;;;;Checking loop for number of R-R intervals;;;;;;;;;;;;;;
93: ;
94: ;if number of R-R intervals <= no_data then loop, o/w pass to the next
95: cont:
96:       cmp     bx,no_data
97:       jle     cont
98: ;save memory size for R-R intervals
99:       mov     di,bx
100: ;
101: ;;;;;;;;Retrieve original default timing pulse frequency;;;;;;;;;;;;
102: ;
103:      mov     al,00110110b    ;timer setting data
104:      out     43h,al      ;port address for timer setting
105:      mov     ax,0ffffh  ;data for default timer frequency
106:      out     40h,al      ;set lower 8 bit of timer frequency
107:      mov     al,ah
108:      out     40h,al      ;set higher 8 bit of timer frequency
109: ;
110: ;;;;;;;;Retrieve address of default timing pulse routine;;;;;;;;;;;
111: ;
112:      mov     cx,mm1  ;offset of default timing pulse routine
113:      mov     dx,mm2  ;segment of default timing pulse routine
114:      push    ds
115:      mov     ax,0
116:      mov     ds,ax
117:      mov     si,112  ;memory address for location of timing pulse
118:                      ;routine
119:      mov     word ptr [si+2],dx   ;set segment of default routine
120:      mov     word ptr [si],cx     ;set offset of default routine
```

Fig 4. Continued

```
121:      pop     ds
122: ;
123: ;;;;;;;;Transfer measured R-R intervals to BASIC data buffer;;;;;;;;;;;
124: ;
125: ;transfer number of measured R-R intervals to a BASIC variable NR
126:      shr     di,1
127:      mov     bx,[bp+14]
128:      mov     [bx],di
129: ;transfer R-R intervals in assembly routine memory to BASIC data buffer
130:      mov     cx,di
131:      mov     si,0            ;initialize address indicator
132: a4:
133:      push    ds
134:      push    cs
135:      pop     ds
136:      mov     dx,[ary_x+si]   ;address of assembly routine memory
137:      pop     ds
138:      mov     bx,[bp+6]       ;address of BASIC data buffer
139:      mov     [bx+si],dx      ;R-R intervals to BASIC data buffer
140:      inc     si              ;increment address indicator
141:      inc     si              ;by 16 bit
142:      loop    a4
143: ;
144: ;;;;;;;;End of main routine---Return to BASIC program;;;;;;;;;;;;;;;
145: ;
146:         pop     bp
147:         ret     10
148: ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
149: ;;;;;;;;TIMING PULSE ROUTINE PERFORMED BY EACH TIMING PULSE;;;;;;;;;;;;
150: ;;;;;;;;GENERATED BY A 8253-5 TIMER CHIP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
151: ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
152: adcon  proc far       ;define timing pulse routine procedure
153:      push    ds
154:      push    ax
155:      push    cx
156:      push    dx
157:      push    cs
158:      pop     ds
159: ;analog to digital conversion of ECG signal using
160: ;a KSE-8627 A/D converter
161:      mov     dx,380h  ;port address for channel selection = 380h
162:      mov     ax,7     ;current channel number = 7
163:      out     dx,ax    ;channel selection
164:      mov     dx,385h  ;port address for conversion start = 385h
165:      mov     ax,0     ;start signal = 0
166:      out     dx,ax    ;conversion start
167: waitt:
168:      mov     dx,380h  ;port address for reading lower 8 bit data
169:      in      al,dx    ;read lower 8 bit
170:      mov     lsb,al   ;temporal storage of lower 8 bit data
171:      mov     dx,381h  ;port address for reading higher 8 bit data
172:      in      al,dx    ;read higher 8 bit
173:      mov     msb,al   ;temporal storage of higher 8 bit data
174:      and     al,00010000b   ;check for completion of reading
175:      jnz     waitt     ;if not yet read then read again
176:                        ;otherwise pass to the next
177:      mov     al,lsb
178:      mov     ah,msb
179:      and     ah,0fh    ;obtain A/D converted 12 bit data
180: ;exponential smoothing for A/D converted data using forecasted
```

Fig 4. Continued

```
;value and given smoothing coeff.
      mov     cl,byte ptr s_c
      shr     ax,cl
      add     ax,m1
      shr     m1,cl
      sub     ax,m1
      mov     m1,ax  ;save smoothed data for next use
;compare smoothed data with R peak detection criteria
      cmp     ax,2457  ;compare with minimum R peak height
      jl      branch
      cmp     si,latency  ;compare R-R interval with minimum
      jl      branch
;if R peak then save R-R interval counter into assembly routine memory
;otherwise pass to branch
      mov     [ary_x + bx],si   ;save R-R interval counter
      inc     bx                ;increment address indicator
      inc     bx                ;by 16 bits
      mov     si,0              ;reset R-R interval counter
branch:
      inc     si         ;increment R-R interval counter by one
;return to main routine
      pop     dx
      pop     cx
      pop     ax
      pop     ds
      iret
adcon   endp       ;end of timing pulse routine procedure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;ESTABLISH AREA FOR MEMORIES USED IN MAIN ROUTINE;;;;;;;;;;;;;
;;;;;;;;OR IN TIMING PULSE ROUTINE;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      m1      dw      ?   ;memory for forecasted value
      lsb     db      ?   ;memory for A/D converted lower 8 bit data
      msb     db      ?   ;memory for A/D converted higher 8 bit data
      no_data dw      ?   ;memory for number of R-R intervals
      freq    dw      ?   ;memory for A/D conversion frequency
      latency dw      ?   ;memory for minimum R-R intervals
      s_c     dw      ?   ;memory for code of smoothing coeff.
      ary_x   dw      20000 dup (?)  ;memory for R-R intervals
      mm1     dw      ?   ;memory for offset of timing pulse routine
      mm2     dw      ?   ;memory for segment of timing pulse routine
rrm     endp      ;end of assembly routine
code      ends
          end
```

력 자료로 사용하도록 하였다.

8253-5 timer에서 timing 신호가 발생될 때마다 microprocessor는 수행하던 연산을 멈추고: (1) system clock의 시간을 갱신하고, (2) diskette drive의 motor구동 상태를 확인하고, (3) PC system이 사용하는 112번부터 115번까지의 memory에 기록된 번지(112, 113번에는 offset 번지수, 114,115 번에는 segment 번지수가 각각 기록되어 있음)로 분기하여 그 번지로부터 시작되는 routine(특별히 지정하지 않으면 PC system timing pulse routine)을 수행한다. 이 timing pulse routine은 사용자가 임의로 변경할 수도 있다. 만일 112, 113번 memory에 사용자가 작성할 timing pulse routine의 offset 번지를, 114, 115번 memory에 사용자가 작성할 timing pluse routine의 segment번지를 새로 기록하면 timing 신호가 발생될 때마다 사용자가 작성한 routine이 시행된다[1].

R-R구간 측정이 끝난 후 PC system을 원상 복귀시키기 위하여 112번에서 115번에 기록되어 있는 기존의 PC system timing pulse routine의 offset 및 segment 번지를 각각 mm1, mm2의 변수에 임시 보

관한다(64-65번 문장).

main routine에서 작성된 자료(변수 latency와 s..c에 저장된 값, 및 변수 m1에 저장된 지수 평할 과정에서 사용될 최초의 추정치)를 timing pulse routine에서도 사용할 수 있도록 main routine과 timing pulse routine의 자료(data) segment address를 일치시킨다(70-80번 문장).

8253-5 timer의 (16진수로 표현된) 43h 번 입력 port를 통해 주파수 변경을 위한 신호를 보내고 (84-85번 문장), freq에 담긴(원하는 주파수를 발진시키기 위한) 값을 하위 8bit와 상위 8bit로 나누어 40h번 port를 통해 보낸다(86-89번) [1]. microprocessor가 89번 문장까지의 프로그램을 수행하면 8253-5 timer는 18.206Hz의 PC system(default) 주파수에서 사용자가 원하는 주파수로 변경된 timing 신호를 발진시킨다.

timing pulse routine에 의해 측정된 R-R 구간의 갯수가 원하는 갯수 만큼 측정되었는지 확인하고(95-95번 문장) 측정이 완료되면 8253-5 timer에서 발생되는 timing 신호 주파수를 원래 상태의 18.206Hz로 복귀시킨다(102-107번 문장). 또한 11번에서 115번 memory에 기록된, R-R 구간 측정을 위한 timing pulse routine의 offset 및 segment address도 mm1, mm2에 임시 보관했던, 원래의 PC system timing pulse routine의 offset 및 segment address로 원상 복귀시킨다(111-120번 문장).

측정된 R-R 구간의 갯수와 R-R 구간 자료들을 assembler routine의 memory 영역으로부터 BASIC data buffer로 옮겨놓고 (125-141번 문장) BASIC program의 call 문장 다음으로 return 한다(145-146번 문장).

8253-5 timer로부터 발진되는 timing 신호에 따라 시행되는 timing pulse routine은 KSE-8627 A/D변환기로 하여금 A/D변환을 하도록 지시하는 것으로부터 시작된다.

컴퓨터와 외부 회로 사이의 자료 입출력에 사용하는 380h번 port에 사용할 A/D 변환기의 channel 번호(본 체계에서는 7번)를 보내고 (161-163번 문장) 38h번 port에 변환 시작을 지시하는 신호를 보낸다 (164-165번 문장).

A/D변환된 12bit 자료 중 하위 8bit를 380h 번 port로부터 받아서 변수 lsb에 임시 저장하고, 318h 번지 port로부터 상위 4bit를 받아서 변수 msb에 임시 저장한다(168-173번 문장). 8bit memory인 변수 msb에 저장된 다섯번째 bit의 2진수는 A/D 변환 완료 신호를 포함하고 있는데 이 숫자를 점검하여 A/D변환이 완료되었는지 확인한다. 완료되지 않으면 계속해서 380h번 port로부터 하위 8bit, 381h번 port로부터 상위 4nit를 받는 과정을 되풀이한다(167-175번 문장). A/D변환이 완료되면 하위 8bit 자료와 상위 4bit 자료를 register(AX)에 임시 저장하여 지수 평할 과정에 사용한다(177-178번 문장).

T-1 단계(T=1,2,3...)에서 지수 평할되어 변수 m1에 저장된 값을 S_{T-1}이라 하고, 입력된 지수 평할 계수를 α라 하고, T단계에서 A/D 변환되어 register에 저장된 값을 Xt라 하면. T단계에서 지수 평활된 값 St는

$$S_T = \alpha X_T + (1-\alpha) S_{T-1}$$

로서 구해진다(181-186번 문장).

제1단계의 지수 평할치 S_1을 구하기 위해 필요한 S_0의 값은 임의로 결정되어 지수 평할 과정이 시작되기 전에 변수 m1에 저장해둔다(79번 문장).

R-R구간의 경과된 시간을 세는 register(SI)를 1 증가시키고 (200번 문장)지수 평활될 자료를 사용하여 R 첨두 탐지 기준과 비교한다.(189-192번 문장). 기준을 만족시켰을 경우에는 그동안 register가 count 했던 R-R구간 자료를 assembler routine의 memory에 저장하고(195번 문장), register를 0으로 되돌리고 (195-198번 문장), timing pulse routine을 시행하기 이전의 main routine으로 돌아간다. R 첨두 탐지 기준을 만족시키지 않은 경우에는 바로 main routine으로 돌아간다.

Fig.4의 assembler 프로그램은 EDLIN, WORDSTAT, 또는 그외의 word processor에 의해 작성할 수 있으며, 본 체계에서는 "RRM. ASM"의 이름으로 diskette에 저장하였다. assembler compiler MASM과 link 프로그램 LINK를 사용하여 "RRM. ASM" file을 PC의 디스크 운용체계(DOS)에서 사용 가능한 실행 file(execution file), "RRM.EXE"로 만든다.

"RRM.EXE" file을 BASIC mode에서 load 할 수 있도록 하기 위해서는 GWBASIC을 run한 후

```
DEF SEG=&H9620
BSAVE "RRM.BIN", 0, &H9D8C
```

의 문장을 입력시켜 diskette에 "RRM.BIN"의 이름으로 save 해두어야 한다. 여기서 &H9620은 RRM.EXE file이 load되는 위치의 segment address를 나타내며 BSAVE 문장의 첫번째 다음에 나오는 0은 offset address를 나타낸다. &H9D8C는 RRM.EXE file의 크기를 byte로 표현한 수이다. RRM.EXE file이 load 되는 memory의 segment address는 debug 프로그램을 이

```
 1:
 2: title        R-R interval measurement
 3: stack        segment stack
 4:      dummy   dw      100 dup(?)
 5: stack        ends
 6: data segment
 7:      in1     db      0dh,0ah,'# of R-R intervals to collect : ','$'
 8:      in2     db      0dh,0ah,'accuracy in HZ (19 - 5400) : ','$'
 9:      in3     db      0dh,0ah,'threshold level of R waves : ','$'
10:*     in4     db      0dh,0ah,'code # for smoothing coeff. : ','$'
11:      msg1    db      0dh,0ah,0dh,0ah,'Start : press ESC key','$'
12:      msg2    db      0dh,0ah,'now collecting of data','$'
13:      msg3    db      0dh,0ah,'data size : ','$'
14:      m1      dw      ?
15:      latency dw      ?
16:      handle  dw      ?
17:      lsb     db      ?
18:      msb     db      ?
19:      no1     dw      ?
20:      no2     dw      ?
21:      no3     dw      ?
22:      no4     dw      ?
23:      count   dw      ?
24:      datsiz  dw      ?
25:      nambuff         db      50
26:                      db      ?
27:                      db      50 dup(?)
28:      datbuff         dw      20000 dup(?)
29:      filnam          db      0dh,0ah,'file name : ','$'
30:              crlf            db      0dh,0ah,'$'
31: data ends
32: code segment public
33:      assume  cs:code,ds:data,ss:stack
34: main:
35:      mov     ax,stack
36:      mov     ss,ax
37:      mov     sp,0fffeh
38:      mov     ax,data
39:      mov     ds,ax
40:      call    clear
41:      call    cursor
42: ;
43: inpt macro   inpt1,inpt2
44:      mov     dx,offset inpt1
45:      mov     ah,9
46:      int     21h
47:      call    decibin
48:      mov     inpt2,bx
49:      call    crlfp
50:      endm
51: ;
52:      inpt    in1,no1
53:      mov     ax,no1
54:      shl     ax,1
55:      sub     ax,2
56:      mov     no1,ax
57:      inpt    in2,no2
58:      mov     dx,12h
59:      mov     ax,34dch
60:      div     no2
```

Fig 5. Assembler program for R-R interval measurement.

```
        shr     no2,1
        mov     cx,no2
        mov     latency,cx
        cmp     dx,no2
        jl      roundoff
        inc     ax
roundoff:
        mov     no2,ax
        mov     bx,no2
        inpt    in3,no3
        inpt    in4,no4
        mov     dx,offset filnam
        mov     ah,9
        int     21h
        mov     dx,offset nambuff
        mov     ah,0ah
        int     21h
        mov     bl,[nambuff+1]
        mov     bh,0
        mov     [nambuff+bx+2],0
;
        mov     dx,offset nambuff+2
        mov     cx,0
        mov     ah,3ch
        int     21h
        mov     handle,ax
;
        mov     dx,offset msg1
        mov     ah,9
        int     21h
cont1:
        mov     ah,1
        int     21h
        cmp     al,27
        jne     cont1
        call    clear
        call    cursor
        mov     dx,offset msg2
        mov     ah,9
        int     21h
;
        mov     ax,0
        mov     ds,ax
        mov     si,112
        push    word ptr[si]
        push    word ptr[si+2]
        mov     ax,offset adcon
        mov     word ptr[si],ax
        mov     ax,seg adcon
        mov     word ptr[si+2],ax
;
        mov     ax,data
        mov     ds,ax
        mov     si,0
        mov     di,0
        mov     ml,500
;
        mov     al,00110110b
        out     43h,al
        mov     ax,no2
```

Fig 5. Continued

```
        out     40h,al
        mov     al,ah
        out     40h,al
;
cont2:
        cmp     di,no1
        jle     cont2
;
        mov     al,00110110b
        out     43h,al
        mov     ax,0ffffh
        out     40h,al
        mov     al,ah
        out     40h,al
;
        mov     ax,0
        mov     ds,ax
        mov     si,112
        pop     word ptr[si+2]
        pop     word ptr[si]
        mov     ax,data
        mov     ds,ax
        mov     count,di
;
        shr     di,1
        mov     datsiz,di
        mov     dx,offset msg3
        mov     ah,9
        int     21h
        mov     bx,datsiz
        call    putdec
;
        mov     bx,handle
        mov     dx, offset datbuff
        mov     cx,count
        mov     ah,40h
        int     21h
;
        mov     bx,handle
        mov     ah,3eh
        int     21h
        mov     ax,4c00h
        int     21h
;
adcon  proc far
        push    ds
        push    ax
        mov     ax,data
        mov     ds,ax
        mov     dx,380h
        mov     ax,7
        out     dx,ax
        mov     dx,385h
        mov     ax,0
        out     dx,ax
waitt:
        mov     dx,380h
        in      al,dx
        mov     lsb,al
        mov     dx,381h
```

Fig 5. Continued

```
181:        in      al,dx
182:        mov     msb,al
183:        and     al,00010000b
184:        jnz     waitt
185:        mov     al,lsb
186:        mov     ah,msb
187:        and     ah,0fh
188:        mov     cl,byte ptr no4
189:        shr     ax,cl
190:        add     ax,m1
191:        shr     m1,cl
192:        sub     ax,m1
193:        mov     m1,ax
194:        cmp     ax,no3
195:        jl      branch
196:        cmp     si,latency
197:        jl      branch
198:        mov     [datbuff+di],si
199:        inc     di
200:        inc     di
201:        mov     si,0
202: branch:
203:        inc     si
204:        pop     ax
205:        pop     ds
206:        iret
207: adcon   endp
208: ;
209: crlfp       proc    near
210:        push    dx
211:        push    ax
212:        mov     dx,offset crlf
213:        mov     ah,9
214:        int     21h
215:        pop     ax
216:        pop     dx
217:        ret
218: crlfp       endp
219: ;
220: clear       proc    near
221:        mov     ax,0600h
222:        mov     bh,7
223:        mov     cx,0
224:        mov     dx,184fh
225:        int     10h
226:        ret
227: clear       endp
228: ;
229: cursor      proc    near
230:        mov     ah,2
231:        mov     bh,0
232:        mov     dx,0
233:        int     10h
234:        ret
235: cursor      endp
236: ;
237: putdec      proc    near
238:        push    bx
239:        push    dx
240:        push    ax
```

Fig 5. Continued

```
241:        mov     dx,0
242:        mov     ax,bx
243:        mov     bx,10000
244:        div     bx
245:        push    dx
246:        mov     dl,al
247:        call    prin
248:        pop     dx
249:        mov     ax,dx
250:        mov     dx,0
251:        mov     bx,1000
252:        div     bx
253:        push    dx
254:        mov     dl,al
255:        call    prin
256:        pop     dx
257:        mov     ax,dx
258:        mov     bl,100
259:        div     bl
260:        push    ax
261:        mov     dl,al
262:        call    prin
263:        pop     ax
264:        mov     al,ah
265:        mov     ah,0
266:        mov     bl,10
267:        div     bl
268:        push    ax
269:        mov     dl,al
270:        call    prin
271:        pop     ax
272:        mov     dl,ah
273:        call    prin
274:        pop     ax
275:        pop     dx
276:        pop     bx
277:        ret
278: putdec      endp
279: ;
280: prin proc    near
281:        push    dx
282:        push    ax
283:        add     dl,30h
284:        mov     ah,2
285:        int     21h
286:        pop     ax
287:        pop     dx
288:        ret
289: prin endp
290: ;
291: decibin      proc    near
292: ;
293:        mov     bx,0
294: newchar:
295:        mov     ah,1
296:        int     21h
297:        sub     al,30h
298:        jl      exit1
299:        cmp     al,9
300:        jg      exit1
```

Fig 5. Continued

```
301:        cbw
302: ;
303:        xchg    ax,bx
304:        mov     cx,10d
305:        mul     cx
306:        xchg    ax,bx
307: ;
308:        add     bx,ax
309:        jmp     newchar
310: exit1:
311:        ret
312: ;
313: decibin        endp
314: ;
315: code ends
316:        end     main
```

용하여 확인할 수 있고 offset address는 보통 0으로 한다. program size는 MASM으로 compile하는 과정 중 생성되는 list file, RRM.LST에 나타나있다[2].

GWBSIC을 run 한 후 Fig.3의 BASIC program을 작성하고, 나중의 사용을 위하여 "RRM.BAS"라는 이름으로 diskette에 save 해두고, run시키면 측정할 R-R 구간 자료의 수, 원하는 A/D 변환 주파수, 지수평할 계수 및 R-R 구간 자료를 저장할 file 이름의 입력을 요구하는 문장이 화면에 나타난다. 이들을 차례로 입력시키면 R-R 구간의 측정이 시작된다. R-R 구간의 측정이 끝나면 R-R구간 자료는 입력된 file 이름으로 diskette에 저장되어 추후 분석에 사용될 수 있다.

BASIC mode에서 input 명령으로 입력된 자료는 BASIC data buffer에 저장되도록 되어 있다. 이 입력 자료를 assembly routine에서 사용하기 위해서는 BASIC data buffer의 위치를 지정하는 별도의 절차(Fig. 4의 156, 157번 문장)가 반드시 요구된다. assembler routine과 결합된 BASIC program은 사용자에게는 편리하지만 이와 같은 절차로 인하여 microprocessor의 연산 시간을 지연시키는 단점이 있다. 만일 입력자료를 BASIC program으로부터 받아들이지 않고 바로 assembler routine에서 받아들이도록 하면 위와 같은 절차는 불필요하게 되고 microprocessor의 연산 시간이 더욱 단축되어 단위 시간당 A/D 변환의 수를 증가시킬 수 있고 결과적으로 R-R구간 자료의 측정 정밀도를 높일 수 있다.

Fig.5는 자료 입력 과정을 assembler로 처리한 R-R 구간 측정 프로그램을 소개한다. Fig.3의 BASIC program과 동일한 순서로 자료의 입력을 요구하며 (43-86번 문장), R-R 구간을 측정하는 나머지 프로그램도 Fig.4의 assembler routine과 거의 동일하다.

Fig.5에 소개한 assembler program을 이용하면 R-R 구간의 측정 정밀도를 5400Hz까지 증가시킬 수 있다.

5. 토의

Intel 8253-5 timer chip에서 발진되어 microprocessor의 작동을 조종하는 timing 신호의 최대 주파수는 1,193,180Hz이다. 따라서 microprocessor 연산 속도의 제한이 없다면 이론적으로는 1/1193180=0.0008381 msec까지의 정밀도로 R-R구간을 측정할 수 있다. 결국, 본 연구에서 개발한 R-R구간 측정 체계의 정밀도는 사용하는 microprocessor의 연산 속도에 의해 좌우된다고 할 수 있다. 연산 속도가 더 빠른 microprocessor(예를 들면 intel 80286 또는 80386)를 사용하면 정밀도를 더 높일 수 있다. microprocessor의 연산 속도가 빨라지면 측정된 R-R 구간 자료를 memory에 임시 저장 했다가 후 처리(off-line processing)하는 현재의 방식으로부터 측정 즉시 처리(on-line processsting)하는 방식으로 변경할 수도 있다. 이 경우, R-R 구간 자료의 즉시 처리 방식에 수반되는 algorithm 개발이 연구 과제가 될 것이다.

일정한 주파수를 가지는 timing 신호에 맞추어 자료 분석하는 방식은 R-R 구간 자료의 측정 뿐만 아니고 시계열(time series) 성격을 지닌 뇌파(electroencephalogram;EEG)나 근전도(electromyogram;EMG)와 같은 다른 생체 신호 분석에도 적용할 수 있다. 각 신호에 적합한 분석 프로그램 개발도 추후의 연구 과제가 될 것이다.

참고문헌

[1] Eggebrecht, L.C., Interfacing to the IBM Personal Computer, Howard W. Same & Co., Indianapolis, 1983.

[2] Lafore. R., Assembly Language Primer for the IBM PC & XT, Waite Group, New York. 1984.

[3] Montgomery. J., Forecasting and Time Series Analysis, McGraw-Hill, New York, 1976

[4] Park, K.S. and Lee, D.H., "Measurement of R-R Intervals with a microprocessor", Journal of the Human Engineering Society of Korea, Vol. 4 No.2, pp.3-10, 1985.