《研究論文》

Journal of the Human Engineering Society of Korea
Vol. 4 No. 2, Dec 1985

마이크로프로세서를 이용한 R-R 구간 측정

(Measurement of R-R Intervals with a Microprocessor)

朴 景 洙† 李 東 夏††

Abstract

This article developes a cost-effective on-line measurement system of R - R intervals in ECG. The system is composed of a R peak detector, a timer and an Apple II computer (a 6502 microprocessor and memories). The system measures the R - R intervals in msec and stores them in a disk, for off-line analysis. A circuit diagram of the R peak detector and programs for controlling the microprocessor are presented.

Ⅰ. 서 론

육체 부하(physical load)나 정신 부하(mental load)를 측정하려는 연구에 있어서 많이 쓰이는 생리 지수(physiological index)로서 심박수(heart rate)나 부정맥(sinus arrhythmia)이 있다. 심박수와 부정맥은 심장 박동 간격(heart beat interval)을 측정하여 구하는데,심장 박동 간격은 심전도(Electrocardiogram, ECG)의 R-R 구간으로 대표된다(그림 1).

대부분의 연구에 있어서 R-R구간은 msec 단위의 정밀도로 측정된다.

이 논문에서는 조작과 통제가 간편한, 경제적인 R-R 구간 측정 체계를 소개한다. 체계는 크게 R 첨두(peak) 탐지기, timer 그리고 AppleⅡ 컴퓨터(부속 6502 microprocessor 및 memory)로 구성되어 있다. 이 중 R 첨두 탐지기는 회로도에 의하여 제작하였고, timer는 Thunderclock Plus timer board를 이용하였다.

R 첨두 탐지기는 심전도로부터 R波의 첨두

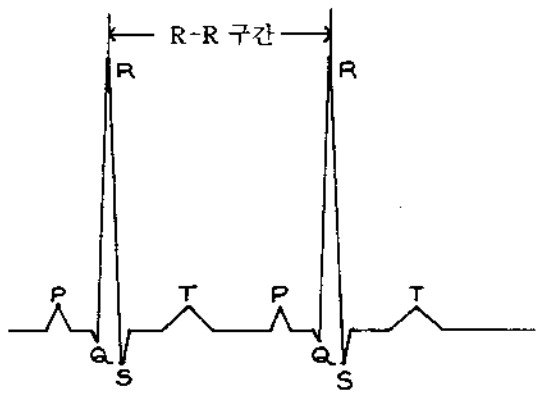


그림 1. 심전도 ; 각 波의 이름 및 R-R구간

†韓國科學技術院 産業工學科
††同 人間-機械/生産 体系 研究室

를 찾는 역할을 하며 microprocessor와 timer는 각 R波의 첨두 사이의 시간 간격을 msec 단위로 측정하는 역할을 한다. 측정된 R-R구간 자료는 AppleⅡ 컴퓨터의 memory에 일단 저장된 후, 분석에 사용되거나 disk에 file로서 기록된다. 체계의 전반적인 구성은 그림2와 같다.

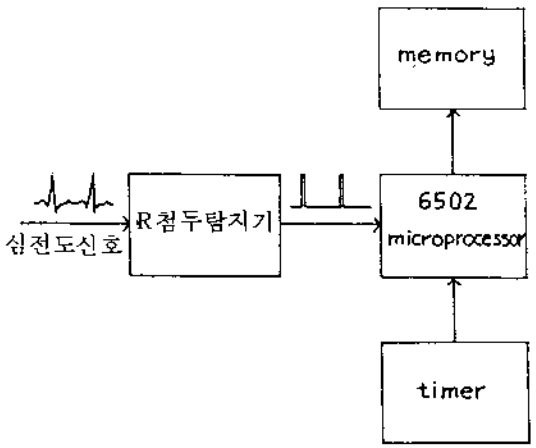
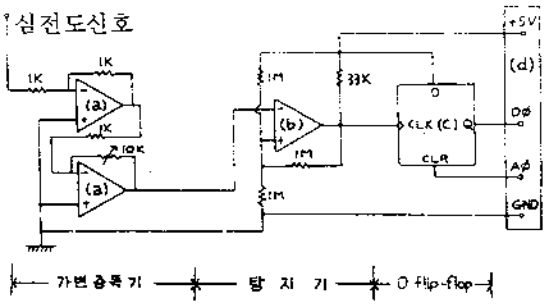


그림 2. 체계구성의 block diagram

Ⅱ. R 첨두 탐지

R-R 구간을 측정하기 위해서는 먼저 심전도상에서 R波를 찾아야 한다. 그림1에서 보는 바와 같이 R波의 첨두는 다른 첨두 波(peak wave), 즉 P波나 T波에 비하여 두드러진 높이를 가지므로, P波나 T波의 첨두가 도달하지 못하는 높이에 일정한 閾値(threshold)를 설정해 놓으면, 그 이상을 도달하는 신호는 R波로 간주될 수 있다. R波가 첨두에 이르는 時點을 찾기 위한 이상적인 방법은, 관측되고 있는 심전도의 R波 첨두 높이에 閾値를 설정해 놓고 심전도 신호(ECG signal)가 이 閾値에 접촉하는 순간을 포착하는 것이다. 그러나 각 R波의 첨두마다 약간씩의 높이 차이가 있으므로, 일정하게 R波의 첨두 높이를 지정하기가 어렵다. 그러므로 실제로는, 평균적인 R波 첨두높이의 90% 수준에 閾値를 설정하고, 심전도 신호가 이 閾値를 접촉하는 순간을, R波가 첨두에 이르는 時點으로 대신한다. 이와 같은 원리에 바탕을 둔 R첨두 첨지기의 회로가 그림3에 나와 있다.



a : operational amplifier (μA 1458)
b : precision voltage comparator (UA 311)
c : D flip-flop (DM 74LS74N)
d : peripheral connector

그림 3. R첨두 탐지기 회로

회로는 가변증폭기, 탐지기(detector), D flip-flop으로 구성되어 있다. 가변증폭기에 입력되는 신호는 생체증폭기(예를 들면 HP 8811A 생체증폭기)를 통하여 적절한 수준으로 증폭되고 濾波(filter)된 심전도이다.

가변증폭기는 탐지기의 특성에 맞게, 입력된 심전도를 적절한 수준으로 증폭하는 역할을 한다.

가변증폭기를 통과한 심전도는 탐지기의 입력 신호(input signal)로서 입력된다. 탐지기는 그림4와 같은 履歷 현상(hysteresis)을 특성으로서 가지고 있다. 즉 탐지기의 입력 신호가 3V 이상이 되는 순간, 출력 신호(output signal)는 0V에서 5V로 바뀐다. 입력 신호가 상단(그림4의 a點)에 도달한 후 0V 이하로 떨어질 때까지 출력 신호는 계속 5V를 유지한다. 입력 신호가 0V 이하로 떨어지는 순간, 출력신호는 0V로 떨어지고 입력 신호가 하단(그림4의 b點)을 지나 다시 3V가 될 때까지, 출력 신호는 0V를 유지한다.

이와 같은 탐지기의 특성에 맞추기 위하여, 그림5(a)와 같이 탐지기에 입력되는 심전도 신호는 기준선(baseline)이 0V가 되도록 생체증폭기로써 위치(position)를 조정해야 하며, 생

체증폭기나 가변증폭기를 사용하여, 평균적인 R波의 첨두높이가 3.3V가 되도록 조정해야 한다. 이 때, R波의 첨두를 찾기 위한 閾値는 3V이다. 가변증폭기를 통하여 나온 심전도와 탐지기의 출력 신호가 그림 5의 (a),(b)에 각각 나와 있다.

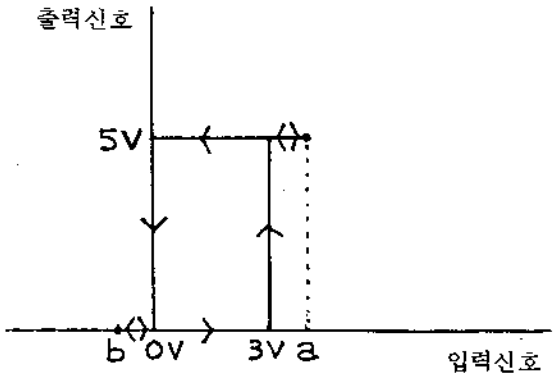


그림 4. 탐지기의 履歷현상

D flip-flop은 탐지기의 출력 신호로 발생되는 톱니 모양의 신호(그림 5의 (b))를 입력으로 하여 그림 5의 (c)와 같이 5V의 impulse를 출력으로 발생한다. 이 impulse로써, 심전도상에서, R波가 첨두에 이르는 時點을 microprocessor에 전달한다. D flip-flop의 두 단자(CLR, Q)는, Apple Ⅱ 컴퓨터의 peripheral connector중 사용할 수 있는 임의의 slot을 선정하여, 그 slot의 제 2번핀(A0)과 제 49번 핀(D0)에 각각 연결한다.

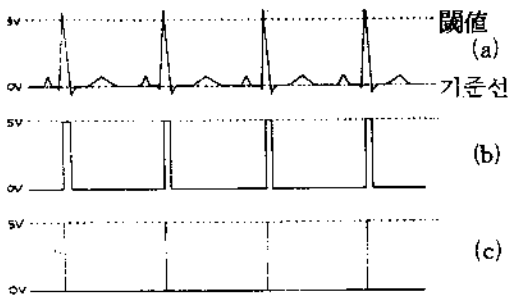


그림 5. 출력신호 -
(a) 가변증폭기
(b) 탐지기
(c) D flip-flop

본 체계에서 D flip-flop은 peripheral connector의 제 3번 slot에 연결되어 있다. Apple Ⅱ컴퓨터가 제3번 slot을 사용하여 외부와의 communication을 할 때, 사용하는 input/output 장소의 base address는 16진법으로 표시하여 C0B0~C0BF인데 microprocessor가 D flip-flop으로부터의 신호를 받아들이는데 사용하는 memory는 이 중에서 address의 LSB (least significant bit)가 0인 것 및 1인것 각 하나씩이다.

본체계에서는 address가 C0B2와 C0B1인 memory를 사용하고 있다. Microprocessor가, 예를 들어 C0B2번지의 내용을 accumulator에 load할 경우에는, peripheral connector의 A15~A0 핀에는 번지 수 "C0B2"가 2진법으로 "1100,0000,1011,0010"과 같이 배치되고 이와 동시에 peripheral connector의 D7~D0핀에 2진법으로 배치된 값이, C0B2 번지의 memory에 입력되고, 그 값이 microprocessor의 accumulator에 load된다.

D flip-flop이 그림 5(b)에 나와있는 톱니모양의 신호로부터 그림 5(c)와 같은 impulse 신호를 만드는 원리는 다음과 같다. 톱니 모양의 신호가 0V에서 5V로 될 때, 그림 3에 나와 있는 D flip-flop의 D에 걸려있던 5V가 Q에 전달되고, 따라서 Q와 연결된 D0 핀에 5V가 걸리게 된다. 이 때, microprocessor가 C0B2번지의 내용을 microprocessor의 accumulator에 load시키면 A0의 값은 0이 된다. 동시에 D0에 걸린 5V는 1로 code되므로, D7~D0에 2진법으로 걸린 값은, LSB(least significant bit)가 1인 상태로 accumulator에 load된다. 이것으로써, microprocessor는 심전도 신호 상에서 R 첨두가 발생한 것으로 인식한다. 바로 그 다음 단계에서, microprocessor는 C0B1번지의 내용을 accumulator에 load하도록 되어 있는데, 그러면 peripheral connector의 A15-A0 핀에 "1100,0000,1011,0001"(C0B1을 2진법으로 표기)이 배치되고 A0의 값은 1이 된다. 이것으로써 A0와

연결된 D flip-flop의 CLR을 clear하게 되고 동시에 Q의 값은 5V에서 0V로 떨어진다. 이와 같은 과정은 불과 8 μsec사이에 발생하므로, D flip-flop의 Q단자에는, 지속 시간 8 μsec의 impulse가 만들어진다.

Ⅲ. R-R 구간 측정

Microprocessor가 수행하는 역할은 다음과같다. 우선 impulse확인 loop를 수행하는데, 여기서는 peripheral connector의 제 49번 핀(D 0)를 통하여 impulse가 들어왔는가의 여부를 每 16μsec 주기로 계속 확인한다. 한편, timer는 peripheral connector의 제 30번핀(IRQ)을 통하여 2048 Hz의 timing pulse 신호를 microprocessor에 보낸다. Microprocessor는 각 timing pulse 신호를 받을 때마다, impulse확인 loop의 수행을 중단하고 timing pulse routine을 수행하게 되는데, timing pulse routine에서는 counter memory內의 숫자를 1씩 증가시킨다. Timing pulse routine의 수행이 끝나면 microprocessor는 본래 수행하던 impulse확인 loop를 계속 수행한다. impulse가 들어왔을 경우, microprocessor는 impulse 확인 loop를 벗어나 impulse routine을 수행하게 되는데, 여기서는 그 동안 counter memory에 기록된 숫자를, R-R구간 자료가 보관될 지정된 번지의 memory에 기록하고, counter memory內의 숫자를 0으로 clear한다. Impulse routine을 수행하고 나면 다시 impulse 확인 loop로 되돌아간다. 이와 같은 과정을 되풀이함으로써, impulse 사이의 시간 간격을 지정된 번지의 memory에 연속적으로 기록할 수 있다. 이 시간 간격은 약0.49 msec의 정밀도를 가진다.

그림 6에 주어진 전산 프로그램은 microprocessor로 하여금 위와 같은 impulse 확인 loop, timing pulse routine, impulse routine을 수행하게 한다.

```
###ASSIGNING NAME TO ADDRESS###
LOW     EQU  $A0        ;1  LOW DIGIT COUNTER
HIGH    EQU  $A1        ;2  HIGH DIGIT COUNTER :TOTAL=HIGH*100+LOW
INDCT   EQU  $A2        ;3  INDICATOR OF ADDRESS, WHERE TIMING PULSES
                            ARE WRITTEN
###INITIALIZING PROGRAM####
        ORG  $0C00      ;4  STARTING ADDRESSS OF RRCOUNT.OBJO
        LDA  #$00       ;5  \
        STA  LOW        ;6  :CLEAR LOW COUNTER MEMORY
        STA  HIGH       ;7  /CLEAR HIGH COUNTER MEMORY
        STA  INDCT      ;8  \
        LDA  #$0E       ;9  :ASSIGN STARTING ADDRESS (0E00)
        STA  INDCT+1    ;10 /TO ADDRESS INDICATOR (INDCT)
        LDA  #$00       ;11 \
        STA  $03FE      ;12 :INDICATE LOCATION TO GO (0D00)
        LDA  #$0D       ;13 :WHEN TIMING PULSE OCCURS
        STA  $03FF      ;14 /
        LDY  #$70       ;15 INDICATE THE NUMBER OF SLOT WHERE  TIMER
                            IS INSERTED (ROUTINE)
        LDA  #$40       ;16 \
        STA  $C080,Y    ;17 :ALLOW MICROPROCESSOR TO RECOGNIZE
        CLI             ;18 /TIMING PULSE (ROUTINE)
        LDY  #$00       ;19 CLEAR Y REGISTER
###IMPULSE IDENTIFICATION LOOP###
LOOP    LDA  $C0B2      ;20 IDENTIFY WHETHER IMPULSE OCCURS
        AND  #$01       ;21 MAKE  CONTENT OF  ACCUMULATOR  (1),  IF
                            IMPULSE OCCURS, OTHERWISE, MAKE IT (0)
        BNE  BR1        ;22 IF IMPULSE OCCURS, THEN BRANCH TO BR1
        JMP  LOOP       ;23 OTHERWISE, JUMP TO LOOP
###IMPULSE ROUTINE###
BR1     LDA  $C0B1      ;24  CLEAR  D  FLIP-FLOP  :CONTENT  IS
                             IMMATERIAL
        SEI             ;25 SET TIMING PULSE TO BE MASKED
```

```
        LDA  HIGH       ;26 LOAD TIMING PULSES FROM HIGH COUNTER
        STA  (INDCT),Y  ;27 STORE TIMING PULSES TO MEMORY (0E00+Y)
        INY             ;28 INCREMENT Y REGISTER BY 1
        LDA  LOW        ;29 LOAD TIMING PULSES FROM LOW COUNTER
        LDA  (INDCT),Y  ;30 STORE TIMING PULSES TO MEMORY (0E00+Y)
        LDA  #$00       ;31 \
        STA  HIGH       ;32 :CLEAR LOW AND HIGH COUNTERS
        STA  LOW        ;33 /
        INY             ;34 INCREMENT Y REGISTER BY 1
        BNE  BR2        ;35 IF Y<256,  THEN BRANCH TO BR2
        LDA  #$00       ;36 \
        STA  $C0F0      ;37 /TURN OFF THE TIMER INSERTED IN THE
                             7 TH SLOT
        RTS             ;38 RETURN TO SUBROUTINE
BR2     CLI             ;39 ALLOW MICROPROCESSOR TO RECOGNIZE  TIMING
                            PULSES
        JMP  LOOP       ;40 JUMP TO LOOP
###TIMING PULSE ROUTINE###
        ORG  $0D00      ;41 STARTING ADDRESS OF TIMING PULSE ROUTINE
        TXA             ;42 \
        PHA             ;43 :
        TYA             ;44 :SAVE X AND Y REGISTER TO STACKS
        PHA             ;45 /
        LDY  #$70       ;46 LOAD  SLOT  INDEX  (7)  TO
                            Y REGISTER (ROUTINE)
        LDA  $C088,Y    ;47 \
        LDA  $C080,Y    ;48 /CLEAR TIMING PULSE (ROUTINE)
        LDA  #$01       ;49 \
        CLC             ;50 :
        ADC  LOW        ;51 :INCREMENT LOW COUNTER BY 1
        STA  LOW        ;52 /
        CMP  #$64       ;53 IF TIMING PULSES<100, THEN BRANCH TO BR3
        LDA  #$00       ;54 \
        STA  LOW        ;55 /CLEAR LOW COUNTER
        LDA  #$01       ;56 \
        CLC             ;57 :
        ADC  HIGH       ;58 :INCREMENT HIGH COUNTER BY 1
        STA  HIGH       ;59 /
BR3     PLA             ;60 \
        TAY             ;61 :
        PLA             ;62 :RESTORE X AND Y REGISTER
        TAX             ;63 /
        LDA  $45        ;64 RESTORE SAVED CONTENT IN $45 (ROUTINE)
        RTI             ;65 RETURN FROM TIMING PULSE ROUTINE
```

그림 6 . microprocessor 통제를 위한 전산 프로그램

그림 6에 주어진 assembler 프로그램을 작성하는 요령은 다음과 같다. 먼저, "EDASM"이라는 assembler compiler를 run 시키면 스크린에 ":"이 나타난다. ":"에 이어 그림 6에 주어진 프로그램을 한 줄씩 타자한 후 return 鍵을 눌러 입력시킨다. 프로그램을 모두 입력시킨 후 RRCOUNT라는 이름으로 save 시킨다. 그 다음 단계로, ASMRRCOUNT를 타자한 후 RETURN鍵을 누르면 프로그램 중 "INITIALIZING PROGRAM", "IMPULSE IDENTIFICATION LOOP" 그리고 "IMPULSE ROUTINE"이 RRCOUNT.OBJ 0라는 이름으로 compile되고, "TIMING PULSE ROUTINE"이 RRCOUNT.OBJ1이라는 이름으로 compile 된다. 프로그램 중 ";" 뒤에 언급된 내용은 註釋(remark)으로서 프로그램 내용에 영향을 미치지 않는다. 註釋 중, 표기된 숫자는 문장 번호로서 프로그램 설명을 위하여 편의상 나타내었다. 프로그램의 각 부분을 설명하면 다음과 같다.

프로그램 중 "ASSIGNING NAME TO ADDRESS" 부분에서는 2개의 timing pulse counter의 address에 "LOW"와 "HIGH"라는 이름을 할당한다(1, 2번 문장). 이 중 "LOW"라는 이름의 counter에는 1단위의 timing pulse의 갯수가 count되고 "HIGH"라는 이름의 counter에는 100단위의 timing pulse의 갯수가 count된다.

timing pulse의 갯수를 기록할 memory 의 번지를 지정하는 indicator의 address에는 "INDCT"라는 이름이 할당된다(3번 문장).

"INITIALIZING PROGRAM"에서 하는 일은 다음과 같다. 먼저 RRCOUNT.OBJ0가 시작되는 memory 상의 번지를 지정한다(4번 문장).

LOW 및 HIGH counter를 clear한다(5~7번 문장).

"INDCT"에게 LOW 및 HIGH counter가 count한 timing pulse를 0E00번지부터 기록하기 시작한다고 통보한다(8~10번 문장).

Timing pulse routine이 시작되는 번지(0D00)를 지정한다(11~14번 문장).

Microprocessor에 timer가 제 7번 slot에 꽂혀 있다고 지시한다(15번 문장).

Microprocessor로 하여금 timer로부터 들어오는 2048 Hz의 timing pulse를 인식하도록 한다(16~18번 문장).

Y register를 clear한다(19번 문장).

"IMPULSE IDENTIFICATION LOOP"를 설명하면 다음과 같다.

Microprocessor가 C0B2번지의 내용을 load함으로써 peripheral connector의 D0 핀을 통해 impulse가 들어왔는지의 여부를 확인한다(20번 문장).

D7~D0 핀에 걸려 있는 값과 "1"을 AND 연산(operation)함으로써 D0에 1이 걸릴 경우(impulse가 들어온 경우)에만 1의 값을 갖고, 그 외에는 0의 값을 갖도록 한다(21번 문장).

Impulse가 들어온 경우에는 impulse routine으로 分枝하고 impulse가 들어오지 않은 경우에는 계속 impulse 확인 loop를 수행하도록 한다(22, 23번 문장).

"IMPULSE ROUTINE"을 설명하면 다음과 같다.

우선, C0B1번지의 내용을 load함으로써 peripheral connector의 A0 핀에 1이 걸리도록 한다. A0 핀에 1이 걸리면 A0와 연결된 D flip-flop의 CLR 단자가 clear된다(24번 문장).

Impulse routine을 수행하는 중 timing pulse routine을 수행하지 못하도록 timing pulse routine 수행을 보류시킨다(25번 문장).

HIGH counter의 내용을 0E00번지부터 시작하는 짝수 번지에, LOW counter의 내용을 0E01번지부터 시작하는 홀수 번지에 각각 기록한다(26~30번 문장).

LOW 및 HIGH counter를 clear한다(31~33번 문장).

Y register의 내용을 1증가시킨다(34번문장).

128개의 R-R 구간 자료가 memory 상에 기록될 때까지 프로그램은 계속 진행되는데, 프로그램을 계속 진행시키기 위하여 BR2로 分枝한다(35번 문장).

R-R 구간 자료가 도합 128개가 기록되면, timer의 작동을 중지시키고, R-R 구간측정을 종료한다(36~38번 문장).

BR2로 分枝한 경우 timing pulse routine 수행을 재개시키며 impulse 확인 loop로 되돌아간다(39,40번 문장).

"TIMING PULSE ROUTINE"을 설명하면 다음과 같다. 우선, timing pulse routine이 시작되는 memory 상의 번지(0D00)를 확인한다(41번 문장).

X와 Y register의 내용을 stack에 쌓아둔다(42~45번 문장).

Routine으로서, timing pulse를 clear한다(46~48번 문장).

LOW counter의 내용을 1증가시킨다(49~52번 문장).

```
10 D$ =  CHR$ (4)
20  PRINT D$;"BLOAD RRCOUNT.OBJ0"
30  PRINT D$;"BLOAD RRCOUNT.OBJ1"
40  CALL 3072
50  PRINT D$;"OPEN INTERVALS"
60  PRINT D$;"WRITE INTERVALS"
70  FOR I = 0 TO 255 STEP 2
80 A =  PEEK (3584 + I)
90 B =  PEEK (3585 + I)
100 C =  INT ((A * 100 + B) * .48828125 + .5)
110  PRINT C
120  NEXT I
130  PRINT D$;"CLOSE INTERVALS"
140  END
```

그림 7. R-R 구간 측정을 위한 전산 프로그램

LOW counter에 기록된 숫자가 100 미만일 경우에, 그대로 timing pulse routine을 벗어나기 위하여 分枝한다(53번 문장).

LOW counter에 기록된 숫자가 100이 되었을 경우에는 LOW counter의 내용을 clear시키고 HIGH counter의 내용을 1증가시킨다(54~59번 문장).

X와 Y register의 내용을 회복시키고, timing pulse routine을 수행하기 이전의 프로그램으로 되돌아간다(60~65번 문장).

R-R 구간을 측정하는 절차는 다음과 같다. 그림 2와 같이 R-R구간 측정을 위한 체계를 구성하여 놓고, 그림 6에 주어진 전산 프로그램을 disk에 save한 다음, 그림 7에 주어진 전산 프로그램을 Apple Ⅱ 컴퓨터에 입력시키면 스크린에는 명멸하는 點이 나타난다. run을 타자한 후 return 鍵을 누르면 R-R 구간에 대한 측정이 on-line으로 진행된다. 일정한갯수의 R-R 구간 자료(그림 6에 주어진 프로그램으로는 128개)가 측정되면 이들은 곧 바로 "INTERVALS"라는 이름의 file로서 disk에 기록된다.

. 그림 7에 주어진, R-R 구간 측정을 위한 전산 프로그램을 설명하면 다음과 같다. 우선 initializing program, impulse 확인 loop, 그리고 impulse routine을 disk로부터 memory상의 0C00 번지부터 0C46번지 사이에 옮겨놓는다(20번 문장). Timing pulse routine을 0D00번지부터 0D26번지 사이에 옮겨놓는다(30번 문장). Memory상의 0C00(10진법으로 3072)번지를 찾아가서 R-R 구간 측정을 시작한다(40번 문장). 50번문장부터 130번까지의 문장은, 지정된 번지의 memory로부터 R-R 구간 자료를 회수하여 msec 단위로 환산한 다음, disk에 file로 기록하는 프로그램이다. 128개의 R-R 구간 자료는 memory의 0E00 번지부터 0EFF번지 사이에 기록되어 있다. 한개의 R-R 구간 자료는 8 bit memory 두개를 사용하므로 첫번째 R-R 구간자료는 0E00(10진법으로 3584), 0E01(10진법으로 3585)번지에, 두번째 R-R구간 자료는 0E02, 0E03 번지에, ……, 이와 같은 식으로 기록되어 있다. 따라서 R-R 구간자료를 memory에서 회수할 때에는 이들 번지로부터 peek해야 한다(80,90번 문장). R-R 구간 자료는 첫번째 번지의 memory에 100 단위의 timing pulse 갯수가, 두번째 번지의 memory에는 1단위의 timing pulse갯수가 기

록되어 있다. 따라서 첫번째 번지의 memory에 들어있는 수에 100을 곱하여 두번째 번지의 memory에 들어있는 수와 합해야 R-R 구간 內의 총 timing pulse 갯수가 구해진다. timing pulse는 주기가 $\frac{1}{2048000}$ msec이므로, timing pulse 갯수에 주기를 곱하여 R-R 구간 자료를 msec로 환산한다(100번 문장).

參 考 文 獻

〔1〕 Apple Ⅱ reference manual, Apple Computer Inc., 1979.

〔2〕 Thurderclock Plus installation & operating manual, Thunderware Inc., Oakland California, 1980.