MULTICHANNEL HARDWARE–SOFTWARE SYNCHRONOUS DETECTOR WITH AVERAGING OF OUTPUT DATA

A.P. Ivanov and A.P. Rostov

Institute of Atmospheric Optics,
Siberian Branch of the Russian Academy of Sciences, Tomsk

Received December 13, 1995

Software–hardware complex for high-quality multichannel synchronous detector with averaging of output data is developed.

Modern experimental instrumentation for studying the atmosphere is almost always equipped with personal computers (PC) which enable one to monitor the experiment, record the results obtained into various storage media of digital information, or even to completely automate it. For these purposes, the signals from analog devices are converted into a digital form by a multichannel analog-to-digital converter (ADC) built into the system block of a PC or an experimental device.[1]

Synchronous detectors and smoothing filters are widely applied in various facilities for experimental investigations of the atmosphere, such as sodars, nephelometers, resonant spectrometers, etc. The amplitude detectors and smoothers for the frequency range of 1–10000 Hz are usually constructed from electronic components of a certain degree of integration. It is not difficult to make a one-channel high-quality synchronous detector with a filter for a single frequency, but it is a real problem in the case of a multichannel and multifrequency one. To obtain similar amplitude–phase characteristics, one must accurately match electronic components of the device, and temperature stabilization should be used for stabilization
of characteristics depending on the field conditions. As a whole, it is a complicated and bulky system.

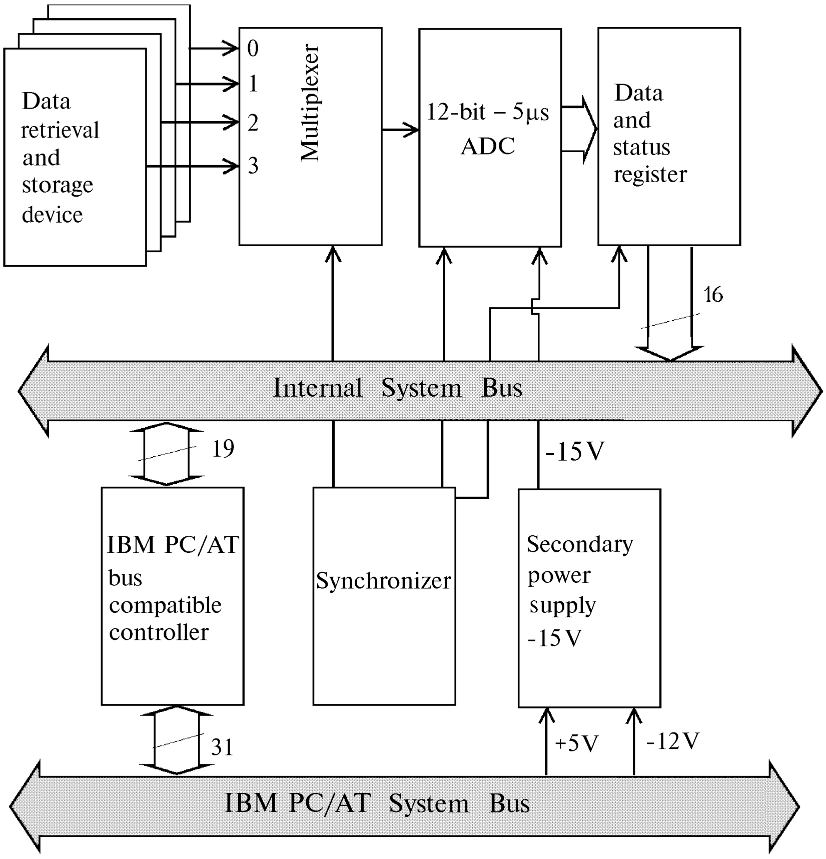

FIG. 1. Structural scheme of the hardware part of the complex.

At present, use of personal computers operating at clock rates higher than 20 MHz made it possible to realize, in this frequency range, some programmed analog devices for experimental units with sufficiently high and stable characteristics.[2]

As an example of a solution of such a problem, the description of a hardware–software complex (HSC) intended for recording signals from an optoacoustic spectrometer[3,6] is given below.

HSC characteristics

dynamic range	72 dB
frequency range	1–10000 Hz
maximum input signal amplitude	+1–5 V
averaging time range	0.1–1000 ms
interface	IBM–PC/AT bus

The alternating current signals from three linear amplifiers of the transducers and the master clock signal are converted into a digital code, and the synchronous detecting and smoothing are programmed for all three channels in real time scale. The signal of the master clock is the synchronous signal in this case.

The structure of the hardware part of the complex is presented in Fig. 1. It consists of a 12-bit middle performance ADC (conversion time is 5 μsec), four data retrieval and storage devices, 4-channel multiplexer, data and status register, generator, device controller in the IBM–PC standard, and the power supply. The unit operates in a cyclic mode with the period of 10 μsec.

The order of the cycle is as follows. Input signals are stored for the digitization time in four data retrieval and storage devices, and then they are converted into the digital ADC code in turn. The rest 5 μsec are used for switching the multiplexer channels and preparing the ADC for the next run and for recording the results into the computer memory.

Figure 2 presents the structure of the data and status register. To improve the performance, the unit status flags are entered into the data register. The flags are in the bits from 12th to 15th. The 12th, 13th, 14th bits show the number of the channel digitized at the moment, and the 15th (sign) bit flags the conversion termination. Thus we managed to minimize the quantity of calls of the unit to ADC when reading the information data and status flags.

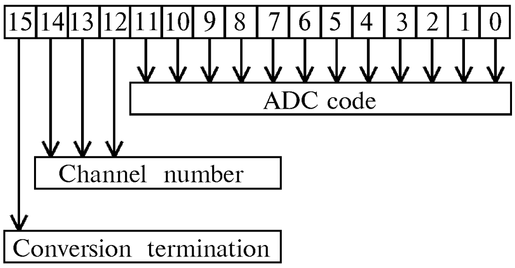

FIG. 2. The structure of the data and status register of the complex.

The software part of the complex is written in the Microsoft assembly language MASM6.1 (Ref. 4). The assembly language enables one to use all the registers and the operation time of the central processor optimally what is very important for real time programs. Using this procedure as a standard external subroutine with a finite set of transferred parameters, a user can develop the operation program for the whole device in high-level languages. This is especially urgent in the work under the control of the multitasking operating system Windows–95 where the access to ports is restricted or not allowed. The initial text of the program realizing the synchronous detection of three processes with smoothing for the C++ DOS language is given below.

```
;= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
;Driver function ADC4ks for programs in C++ language
;= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
;A. Rostov, Tomsk. Ver. 1.1 of 10.05.95
;------------------------------------------------------------------------------------------------------------
;** Call from C ADC4ks (k1, k2, k3, k4, N)
;------------------------------------------------------------------------------------------------------------
;Where k1, k2, k3 are 12-digit detected and averaged over N ms values in 1, 2, 3-channels, and k4 is the value of the
4th channel without any transformations.
; -----------------------------------------------------------------------------------------------------------
;Translate by instruction ML/c ADC4ks.ASM

          .MODEL medium,c; mean model of memory allocation for C++

;Describe the procedure name and the list of transferred parameters
ADC4KS PROTO C k1:SWORD, k2:SWORD, k3:SWORD, k4:SWORD, N:SWORD

         .CODE            ; code part

ADC4KS PROC k1:SWORD, k2:SWORD, k3:SWORD, k4:SWORD, N:SWORD
         movsi,N                 ;take the averaging time in ms
```

```
            and     si,3FFFh            ;the number of ms must not exceed 16382
            xor     ax,ax               ;set the carry flag into 0
            rcl     si,1                ;multiply by 2

;clear the variables
            movdi,offset cs:chnl1
            movcx,8
            xor     ax,ax

clrvar:     mov     cs:[di],ax
            inc     di
            inc     di
            loop    clrvar              ;clear variables cycle

            mov     dx,300h ;ADC address
            mov     di,offset  cs:chnl1

            cli

            mov     bx,0                ;the register BX will be used for fast addition
                                        ;with the carry

beg:        ; major cycle
            mov  cx,10

beg1:       ;minor  cycle

beg4:       in      ax,dx
           and      ax,3000h
           cmp      ax,3000h            ;wait for the digitization of the 4th channel
           jnz      beg4

m1n:       in       ax,dx
           test     ax,3000h            ;wait for the digitization of the 1st channel
           jnz      m1n

m12n:      in       ax, dx
           or       ax, ax
            jns     m12n                ;wait for the beginning of the digitization of the 1st channel

m11n:       in      ax,dx
            and     ax,8FFFh            ;allocate information bits
            js      m11n                ;wait for the end of the 1st channel digitization
            sub     ax,2048             ;translate into the additional code
            not     ax                  ;invert to obtain the module
m14n:       add     cs:[di+0], ax       ;add the obtained number to the sum
            adc     cs:[di+2], bx       ;if there was a carry increase the high-order word

m22n:       in      ax,dx
            or      ax,ax
            jns     m22n                ;wait for the beginning of the 2nd channel digitization
m2n:        in      ax, dx
            and     ax, 8FFFh           ;allocate information bits
            js      m2n                 ; wait for the end of the 2nd channel digitization
            sub     ax, 2048            ;translate into the additional code
            jnc     m24n
            not     ax                  ;invert to obtain the module
m24n:       add     cs:[di+4],ax        ;add the obtained number to the sum
            adc cs:[di+6],bx ;if there was a carry   increase the high-order word
m32n:       in      ax,dx
            or      ax,ax
```

```
        jns     m32n                ;wait for the beginning of 3rd channel digitization
m3n:    in      ax,dx
        and     ax,8FFFh;           allocate information bits
        js      m3n                 ;wait for the end of the 3rd channel digitization
        sub     ax,2048             ;translate into the additional code
        jnc     m34n
        not     ax                  ;invert to obtain the module
m34n:   add     cs:[di+8],ax        ;add the obtained number to the sum
        adc cs:[di+10],bx           ;if there was a carry, increase the high order word

                                    ;the 4th channel is not detected
m42n:   in      ax,dx
        or      ax,ax               ;wait for the beginning of the 4th channel digitization
        jns     m42n
m4n:    in      ax,dx               ;allocate information bits
        and     ax,8FFFh            ;wait for the end of the 4th channel digitization
        js      m4n
        sub     ax,2048             ;translate into the additional code;
        mov     cs:[di+12],ax       ;write the value of 4th channel in the service variable

        loop    beg1                ;end of the minor cycle
        dec     si
        jnz     beg                 ;end of the major cycle

        sti

;compute the mean value of the detected signals
        mov     bx,[bp+6]
        mov     si,[bx]             ;take the number of  averagings in msec
        and     si,3FFFh            ;the number of milliseconds must not exceed 16382
        xor     ax,ax               ;set the carry flag into 0
        rcl     si,1                ; multiply by 2

;the 1st channel
        mov     bx,10
        mov     dx,cs:chnh1
        mov     ax,cs:chnl1
        div     si                  ;first divide by 10
        cwd                         ;convert the 2-byte number into a 4-byte one
        div     bx                  ;divide by the number of the given averagings
        mov     k1,ax               ;assign the result into the variable k1

;the 2nd channel
        mov     bx,10
        mov     dx,cs:chnh2
        mov     ax,cs:chnl2
        div     si
        cwd
        div     bx
        mov     k2,ax

;the 3nd channel
        mov     bx,10
        mov     dx,cs:chnh3
        mov     ax,cs:chnl3
        div     si
        cwd
        div     bx
        mov     k3,ax
;the 4th channel take the direct value of the last measurement
```

```
          mov     ax,cs:chnl4
          mov     k4,ax

          ret 10            ;return from the procedure with stack displacement by 10 bytes

;the service variables will be included in the procedure
chnl1     dw 0              ;the 1st channel low-order word
chnh1     dw 0              ;the 1st channel high-order word
chnl2     dw 0              ;the 2nd channel low-order word
chnh2     dw 0              ;the 2nd channel high-order word
chnl3     dw 0              ;the 3rd channel low-order word
chnh3     dw 0              ;the 3rd channel high-order word
chnl4     dw 0              ;the 4th channel low-order word
chnh4     dw 0              ;the 4th channel high-order word


Copyr BYTE "V1.1 A. Rostov Tomsk–95"
ADC4KS            ENDP      ;the end of the procedure
     END                    ;the end of the program
```

As one can see from the text of the program, the calls to the main memory are reduced to a minimum. Most of the arithmetic and logic operations are performed at the level of processor registers. All the hardware interrupts of the processor are unallowable during the data accumulation operation. Such a construction of the program made it possible to realize the process of reception and detection of the signals with the above-stated characteristics in the 12-MHz computer IBM-PC/286. The backwardness of system hours during the operation of this program can be removed by reading the time code from the non-volatile computer clock[5] at the end of the data set-up process.

The above-mentioned program can be easily adapted for other programming languages. One should only change the log of data transfer. For instance, in order to include it into the dynamically linked library (*.dll) for the medium Windows 3.*, it is necessary to establish the Pascal agreement about the connection and write the definition file (Adc4ks.def), compile by the instruction ml/c/w3 Adc4ks.asm, and link by the instruction Link Adc4ks MyLib.dll, LibW.lib mnocrtdw.lib, Adc4ks.def.

ACKNOWLEDGMENTS

The work described in this paper was partially supported by the International Science Foundation and the Russian Government, Grant No. J29100.

REFERENCES

1. A.P. Rostov, Atm. Opt., **1**, No. 3, 125–126 (1988).
2. K.G. Finogenov, *Programming of Real Time Measuring Systems* (Energoizdat, Moscow, 1990), 256 pp.
3. A.B. Antipov, V.A. Kapitanov, Yu.N. Ponomarev, and V.A. Sapozhnikova, *Optico-Acoustic Method in Laser Spectroscopy of Molecular Gases* (Nauka, Novosibirsk, 1984), 128 pp.
4. Microsoft Corporation, *Microsoft MASM Version 6.1 for MS-DOS and Windows Operating Systems* (1992), 454 pp.
5. R. Jourdain, *Programmer's Problem Solver for the IBM PC, XT&AT* (Brady, New York, 1986).
6. V.A. Kapitanov, G.E. Kulikov, and V.I. Tyryshkin, Atmos. Oceanic Opt. **8**, No. 9, 763–765 (1995).