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.

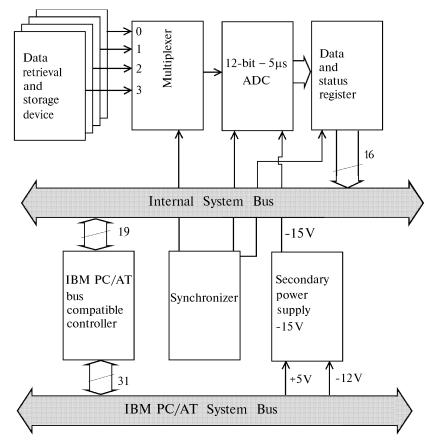


FIG. 1. Structural scheme of the hardware part of the complex.

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 μ sec), 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 μ sec.

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 μ sec 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.

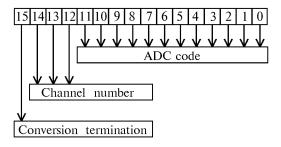


FIG. 2. The structure of the data and status register of the complex.

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 xor rcl | si,3FFFh ax,ax si,1 | ;the number of ms must not exceed 16382 ;set the carry flag into 0 ;multiply by 2 |
|-----------|--------------------------------------|--|---|
| ;clear tl | he variat movdi movcx xor | offset cs:chnl1, | |
| clrvar: | mov inc inc loop | cs:[di],ax di di clrvar | ;clear variables cycle |
| | mov mov | dx,300h ;ADC a di,offset cs:chnl | |
| | cli | | |
| | mov | bx,0 | ;the register BX will be used for fast addition ;with the carry |
| beg: | ; major mov o | | |
| beg1: | ;minor | cycle | |
| beg4: | in and cmp jnz | ax,dx ax,3000h ax,3000h beg4 | ; wait for the digitization of the 4th channel |
| m1n: | in test jnz | ax,dx ax,3000h m1n | ;wait for the digitization of the 1st channel |
| m12n: | in or jns | ax, dx ax, ax m12n | ;wait for the beginning of the digitization of the 1st channel |
| m11n: | in and js sub not | ax,dx ax,8FFFh m11n ax,2048 ax | ;allocate information bits ;wait for the end of the 1st channel digitization ;translate into the additional code ;invert to obtain the module |
| m14n: | add adc | cs:[di+0], ax cs:[di+2], bx | ;add the obtained number to the sum ;if there was a carry increase the high-order word |
| m22n: | in or jns | ax,dx ax,ax m22n | ; wait for the beginning of the 2nd channel digitization |
| m2n: | in and js sub jnc not | ax, dx ax, 8FFFh m2n ax, 2048 m24n ax | ;allocate information bits ; wait for the end of the 2nd channel digitization ;translate into the additional code ;invert to obtain the module |
| m24n: | add | cs:[di+4],ax | ;add the obtained number to the sum e was a carry increase the high-order word |
| m32n: | in or | ax,dx ax,ax | , |

| andax,8FFFh; isallocate information bits iswit for the end of the 3rd channel digitization sub ax,2048 inc m34nmotax;translate into the additional code inc m34nmotax;invert to obtain the module ;add the obtained number to the sum adc cs:[di+10],bxm34n:if there was a carry, increase the high order word ;the 4th channel is not detected | | | | |
|--|----------|----------|--------------------|---|
| and ax,8FFFh; allocate information bits sub ax,2048 ;translate into the additional code jnc m34n add ex:[di+8].ax ;add the obtained number to the sum add ex:[di+8].ax ;add the obtained number to the sum add ex:[di+10].bx ;if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word is and ax,8FFFh ;; vait for the beginning of the 4th channel digitization js m4n wot ex:[di+12].ax ;wait for the end of the 4th channel digitization js m4n wot ex:[di+12].ax ;write the value of 4th channel in the service variable loop beg1 ;end of the minor cycle and si,3FFFh ;the number of averagings in msec and si,3FFFh ;the since the carry flag into 0 rel si,1 if rst divide by 10 cwd ;ve syschn1 mov dx, csychn1 mov dx, csyc | | jns | | ;wait for the beginning of 3rd channel digitization |
| | m3n: | | , | |
| sub ax,2048 ;translate into the additional code jnc m34n into ax invert to obtain the module add cs[di+8],ax idt the obtained number to the sum add cs[di+8],ax if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if the the channel is not detected if the additional code; mov cs:[di+12],ax ; write the value of the thannel digitization if the mov cs:[di+12],ax ; write the value of 4th channel in the service variable if the mov cs:[di+12],ax ; write the value of 4th channel in the service variable if the mov cs:[di+12],ax ; write the value of the channel in the service variable if the mov cs:[di+12],ax ; it the number of averagings in msec and si,3FFFh ; the number of milliseconds must not exceed 16382 xor ax,ax ; iset the carry flag into 0 rel si,1 ; multiply by 2 ; the st channel mov bx,10 mov dx,cs:chnh1 mov dx,cs:chnh2 mov dx,cs:chnh2 mov dx,cs:chnh2 mov dx,cs:chnh2 mov dx,cs:chnh2 mov dx,cs:chnh3 mov dx,cs:chnh3 mov ax,es:chnl3 div si cwd div si cwd | | and | | |
| jnc m34n not ax invert to obtain the module add cs:[di+10],bx if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if there was a carry, increase the high order word if the the channel is not detected if the the detected is the additional code; if if the number of the detected signals mov bx,10p+61 mov bx,10p+61 mov dx,cs:chn11 mov dx,cs:chn11 mov dx,cs:chn11 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:c | | js | | |
| not ax invert to obtain the module add cs:[di+8],ax if there was a carry, increase the high order word if the was a carry, increase the high order word if the the hannel is not detected in a x, dx is invert to obtain the module is not detected if the the hannel is in a x, dx if the the beginning of the 4th channel digitization is m42 in ax, dx is allocate information bits and ax, 8FFFh is wait for the end of the 4th channel digitization is m4 in ax, dx is allocate information bits and ax, 2048 it translate into the additional code: is in z beg is end of the major cycle dec si in z beg is end of the major cycle sti is to bx.[Dp+6] mov cs:[di+12],ax is the the number of averagings in msec and si, 3FFFh is the the carry flag into 0 rel si, 1 is multiply by 2 is the carry flag into 0 rel si, 1 is multiply by 2 is the result into the variable k1 is now bx, 10 mov dx, cs: chnh1 mov ax, es: chnh1 mov ax, es: chnh2 mov ax, es: chnh2 mov ax, es: chnh3 mov bx, 10 mov dx, cs: chnh3 mov ax, es: chnh3 mov ax, e | | | | ;translate into the additional code |
| m34a: add est[di+8],ax if there was a carry, increase the high order word if there was a carry, increase the high order word if the 4th channel is not detected im42a: in ax,ax is wait for the beginning of the 4th channel digitization jas m42a into ax,ax islocate information bits and ax,8FFFh is wait for the end of the 4th channel digitization is word ax,8FFFh is wait for the end of the 4th channel digitization is word ax,8FFFh is wait for the end of the 4th channel digitization is word ax,2048 it ranslate into the additional code; mov est[di+12],ax is write the value of 4th channel in the service variable detected signals mov bx,10 is in a signal in the intervence of averagings in msec and si,3FFFh is the carry flag into 0 rel si,1 is multiply by 2 is infirst divide by 10 is intervence into a 4-byte one div bx is is ifrist divide by 10 is into the advisible by the number of the given averagings mov dx, essechn1 mov dx, essechn1 mov dx, essechn12 mov dx, essechn13 mov dx, essechn14 mov dx, essechn15 mov dx, essechn15 mov dx, essechn15 mov d | | | m34n | |
| 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 ; wait for the beginning of the 4th channel digitization jns m42n 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,[bp+6] mov si,[bp+6]; and si,3FFFh ; 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:chn11 mov ax,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn14 mov ax,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn14 mov ax,cs:chn12 mov dx,cs:chn14 mov ax,cs:chn12 mov dx,cs:chn14 mov ax,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov ax,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn14 mov ax,cs:chn19 mov dx,cs:chn14 mov ax,cs:chn14 mov ax,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn14 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn14 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn14 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn16 | | | | |
| in ax,dx in ax,dx in ax,dx in m42n: in ax,dx in int in ax,dx in in in ax,dx in in in ax,dx in in in ax,dx in in in in in | m34n: | | | |
| m42n: in ax, dx or ax, ax ; wit for the beginning of the 4th channel digitization or ax, ax, ax in ax, dx ; allocate information bits and ax, dx ; and information bits and ax, 2048 ; translate into the additional code; mov cs: [di+12], ax ; write the value of 4th channel in the service variable dec si jnz beg ; end of the minor cycle dec si jnz beg ; end of the major cycle sti ; compute the m=n value of the te-te-ted 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 rel si, 1 ; multiply by 2 ; the 1st channel mov bx, 10 mov ax, cs: chnl1 div si ; convert the 2-byte number into a 4-byte one div bx; divide by 10 cwd ; cs: chnl2 mov 4x, cs: chnl3 mov 4x | | adc cs | :[di+10],bx | ; if there was a carry, increase the high order word |
| or ax,ax ; wit 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 ax,[bp+6] mov si,[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 ax,cs:chn11 div si ; first divide by 10 cwd ; convert the 2-byte number into a 4-byte one div bx ; divide by 10 cwd ; convert the 2-byte number of the given averagings mov k1,ax ; assign the result into the variable k1 ; ; the 2nd channel mov bx,10 mov ax,cs:chn12 mov bx,10 mov ax,cs:chn12 mov bx,10 mov dx,cs:chn14 mov bx,10 mov dx,cs:chn15 mov dx,cs:chn14 mov bx,10 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn17 mov dx,cs:chn18 mov dx,cs:chn18 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn18 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn17 m | | | | ;the 4th channel is not detected |
| | m42n: | in | | |
| 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,10p+61 ;take the number of averagings in msec and si,3FFFh ;the number of averagings in msec and si,3FFFh ;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 dx,cs:chn11 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 dx,cs:chn12 mov ax,cs:chn12 mov dx,cs:chn14 div bx mov dx,cs:chn15 mov ax,cs:chn16 mov ax,cs:chn16 mov dx,cs:chn17 mov ax,cs:chn17 div bx mov dx,cs:chn18 mov ax,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 mov dx,cs:chn19 div bx mov dx,cs:chn19 div bx mov dx,cs:chn19 div bx mov dx,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn14 div bx mov dx,cs:chn15 mov ax,cs:chn15 mov ax,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn18 mov dx,cs:chn19 div bx mov dx,cs:chn19 div bx mov dx,cs:chn19 div bx mov dx,cs:chn19 div bx mov dx,cs:chn19 div bx mov dx,cs:chn19 div bx | | | | ;wait for the beginning of the 4th channel digitization |
| 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:chn11 div si ;comvert the 2-byte number into a 4-byte one div bx_10 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn18 mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn13 mov dx,cs:chn13 m | | v | | |
| 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 div 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:chn11 div si ;first divide by 10 cwd ;convert the 2-byte number of the given averagings mov k1,ax ;assign the result into the variable k1 ;the 2nd channel mov bx,10 mov dx,cs:chn12 mov bx,10 mov dx,cs:chn13 mov ax,cs:chn14 div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn13 mov bx,10 mov dx,cs:chn14 mov bx,10 mov dx,cs:chn15 mov dx,cs:chn16 mov bx,10 mov dx,cs:chn16 mov bx,10 mov dx,cs:chn17 mov dx,cs:chn18 mov bx,10 mov dx,cs:chn19 mov bx,10 mov dx,cs:chn19 mov bx,10 mov dx,cs:chn14 mov bx,10 mov dx,cs:chn15 mov dx,cs:chn16 mov bx,10 mov dx,cs:chn16 mov bx,10 mov dx,cs:chn17 mov dx,cs:chn18 mov bx,10 mov dx,cs:chn19 mov bx,10 mov bx,10 mov dx,cs:chn19 mov bx,10 mov bx,10 mov bx,10 mov bx,10 mov | m4n: | | | |
| sub mov ax,2048 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 ;end of the major cycle sti ;end of the major cycle sti ;end of the momer of averagings in msec and si,[bp+6] mov bx,[bp+6] mov si,[bx] and si,3FFFh ;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 mov bx,10 mov dx,cs:chn11 div si giride by the number of the given averagings mov k1,ax ;assign the result into the variable k1 ;the 2nd channel mov mov bx,10 mov bx,10 mov bx,10 mov bx,10 mov bx,10 | | | | ;wait for the end of the 4th channel digitization |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ | | | | |
| loop beg1 ;end of the minor cycle dec si jnz beg ;end of the major cycle sti sti sti compute the mean value of the detected signals mov bx,[bp+6] 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 sthe terms 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 sthe 1st channel mov bx,10 mov ax,cs:chn11 div si ;first divide by 10 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 sthe 2nd channel mov bx,10 mov ax,cs:chn12 mov ax,cs:chn12 mov k2,ax sthe 3nd channel mov bx,10 mov bx,10 mov ax,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn13 mov ax,cs:chn13 mov bx,10 | | | | |
| 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:chn11 mov ax,cs:chn11 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:chn12 mov ax,cs:chn12 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 div si cwd div bx | | mov | cs:[01+12],ax | ;write the value of 4th channel in the service variable |
| jnz beg ;end of the major cycle sti sti sti sti sti vov bx,[bp+6] mov bx,[b] ;take the number of averagings in msec mov si,[bx] ;take the number of averagings in msec mov si,[bx] ;take the number of averagings in msec mov si,[bx] ;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:chn11 mov ax,cs:chn11 div si 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:chn12 mov dx,cs:chn12 mov dx,cs:chn12 mov dx,cs:chn13 mov k2,ax ;the 3nd channel mov dx,cs:chn13 mov dx,cs:chn14 mov dx,cs:chn15 mov dx,cs:chn16 mov dx,cs:chn17 mov dx,cs:chn16 mov dx,cs:ch1 | | | | ;end of the minor cycle |
| sti sti sti scompute 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:chn11 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:chn12 div si ewd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 mov ax,cs:chn13 div si ewd div bx | | | | |
| <pre>;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:chn11 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 dx,cs:chn12 div si cwd ;div bx mov k2,ax ;the 3nd channel mov dx,cs:chn13 div si cwd ;coxchn13 mov ax,cs:chn13 div si cwd ;coxchn13 mov ax,cs:chn13 div si cwd ;coxchn13 div si cwd ;coxchn14;coxchn15;coxchn15;coxchn15;coxchn15;coxchn16;coxchn16;coxchn16;coxchn16;coxchn17;coxchn17;coxchn17;coxchn17;coxchn18;coxch18;coxchn18;coxchn18;coxchn18;coxchn18;coxchn18;coxchn1</pre> | | jnz | beg | ;end of the major cycle |
| 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:chn11 ; 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 dx,cs:chn12 mov dx,ncs:chn12 giv si cwd div si div bx si giv bx,10 mov dx,cs:chn12 mov dx,cs:chn13 giv si cwd div bx giv bx,10 mov dx,cs:chn13 mov dx,cs:chn13 giv si cwd div si div si giv si cwd div si div </td <td></td> <td>sti</td> <td></td> <td></td> | | sti | | |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ | ;comput | e the m | ean value of the c | detected signals |
| 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:chn11 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:chn12 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 div si cwd div bx | | mov | bx,[bp+6] | |
| xor ax,ax ;set the carry flag into 0 rcl si,1 ;multiply by 2 ;the 1st channel mov bx,10 mov dx,cs:chn11 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:chn2 mov dx,cs:chn12 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 div si cwd div si | | mov | si,[bx] | |
| rcl si,1 ; multiply by 2 ;the 1st channel mov bx,10 mov ax,cs:chn11 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 ax,cs:chn12 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov ax,cs:chn13 div si cwd div si cwd div bx mov ax,cs:chn13 div si cwd div bx | | and | si,3FFFh | |
| <pre>;the 1st channel mov bx,10 mov dx,cs:chn11 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:chn12 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 div si cwd div si</pre> | | xor | ax,ax | ;set the carry flag into 0 |
| mov bx,10 mov dx,cs:chn11 mov ax,cs:chn11 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:chn12 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 div si cwd div si cwd div si cwd div bx mov ax,cs:chn13 div si cwd div bx mov bx,10 mov | | rcl | si,1 | ; multiply by 2 |
| mov dx,cs:chn11 mov ax,cs:chn11 div si ;first divide by 10 cwd channel 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:chn12 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chn13 div si cwd div si cwd div bx mov bx,10 mov dx,cs:chn13 div si cwd div bx mov bx,10 mov bx,10 | ;the 1st | channel | l | |
| mov ax,cs:chnl1 div si ;first divide by 10 cwd 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:chnl2 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chnl3 mov ax,cs:chnl3 div si cwd div bx mov bx,10 mov bx,1 | | mov | | |
| 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 bx,10 mov bx,10 m | | mov | | |
| 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 mov ax,cs:chnl2 div div si cwd div bx mov k2,ax the 3nd channel mov mov bx,10 mov mov dx,cs:chnh3 mov mov ax,cs:chnl3 div div si cwd div bx si cwd div bx | | mov | ax,cs:chnl1 | |
| 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 si cwd div bx | | div | si | ;first divide by 10 |
| mov k1,ax ;assign the result into the variable k1 mov bx,10 mov dx,cs:chh2 mov ax,cs:chl2 div si cwd div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chh3 mov ax,cs:chl3 div si cwd div bx | | | | |
| $ \begin{array}{cccc} \mbox{the 2nd channel} & & & & & & & & \\ \mbox{mov} & & & & & & & & \\ \mbox{mov} & & & & & & & & & \\ \mbox{mov} & & & & & & & & & \\ \mbox{div} & & & & & & & & & \\ \mbox{div} & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & \\ \mbox{the 3nd channel} & & & & & & & & & \\ \mbox{mov} & & & & & & & & & & \\ \mbox{mov} & & & & & & & & & & \\ \mbox{sicc} & & & & & & & & & & & \\ \mbox{mov} & & & & & & & & & & & \\ \mbox{sicc} & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & & & & & & \\ \mbox{div} & & & & & & & & & & & & & & & & & & &$ | | div | bx | |
| 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 bx | | mov | k1,ax | ;assign the result into the variable k1 |
| 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 si cwd div bx | ;the 2nd | l channe | | |
| 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 | | |
| 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 | | |
| 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 | ax,cs:chnl2 | |
| div bx mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chnh3 mov ax,cs:chnl3 div si cwd div bx | | div | si | |
| mov k2,ax ;the 3nd channel mov bx,10 mov dx,cs:chnh3 mov ax,cs:chnl3 div si cwd div bx | | cwd | | |
| ;the 3nd channel mov bx,10 mov dx,cs:chnh3 mov ax,cs:chnl3 div si cwd div bx | | div | | |
| movbx,10movdx,cs:chnh3movax,cs:chnl3divsicwddivbx | | mov | k2,ax | |
| movbx,10movdx,cs:chnh3movax,cs:chnl3divsicwddivbx | ;the 3nd | l channe | el | |
| mov dx,cs:chnh3 mov ax,cs:chnl3 div si cwd div bx | | | | |
| mov ax,cs:chnl3 div si cwd div bx | | mov | | |
| div si cwd div bx | | mov | | |
| div bx | | div | | |
| | | cwd | | |
| mov k3,ax | | div | | |
| | | 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. R | ostov 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 clock⁵ 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).