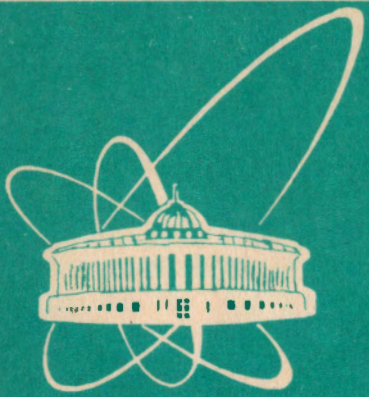

сообщения
объединенного
института
ядерных
исследований
дубна

E10-93-92

I.Stekl, A.Kreipe, A.V.Salamatin

APPLICATION OF RESIDENT PROGRAMS, HARDWARE INTERRUPTS AND EXPANDED MEMORY IN PHYSICAL EXPERIMENTS

1993

At present many personal computers IBM PC AT,XT are used in the JINR. Some of them are connected to CAMAC crates by the controller KK009, which was proposed and constructed in the Laboratory of Nuclear Problems. The controller is fully described in /1/. The possibilities of IBM PC AT for the tasks of the real time are limited. Hardware interrupt is one of the chances to implement the task of the real time on the computers PC AT and improve the performance of computer PC AT.

The data acquisition from CAMAC through the controller can be performed by two types of software. The first type tests all the time the registers of CAMAC controller for signal LAM. The computer is fully engaged in the testing. The second type uses the hardware interrupt. The advantage of interrupt lies in the fact that computer co-operates with any modules in the crate CAMAC only when the data are prepared. It was decided to write for our experiments terminate and stay resident programs (TSR) working with the interrupt.

The computers IBM PC AT have also often the expanded memory. The expanded memory is memory over 640kBytes. Our TSR programs use this memory for data buffers. It decreases the volume of the conventional memory (below 640kBytes) occupied by TSR programs. Software was written on Turbo Pascal. The interrupt procedures were written in Turbo Pascal's inline assembler.

1. Controller KK009

The connection between the computer and modules in the crate CAMAC is performed by the controller KK009 and a computer bus adapter PK 009. The bus adapter occupies one slot on the computer main board. The following features are included in the controller - i) up to 7 crates in one CAMAC system, ii) access to the registers of CAMAC as part of

the computer memory, iii) the execution of multi-crate commands. The bus adapter contains the switches which define number of interrupt and the address of the system of the crates. System of crates occupies addressable space 32kBytes long. The first 128 bytes of the memory space are reserved for the control and status registers and for the multi-crate commands. Writing to the control registers performs some commands (e. g., select crate, init controller, reset signal Inhibit). The control register 0 (the first five bits) determines modules which are the source of the interrupt. The most significant bit (IE) of the control register 0 must be set to logical 1 to enable the interrupt in the controller.

2. How to Stay Resident, Delete Resident Program

As mentioned above the program for the data acquisition from any modules in the crate CAMAC was made to be resident and reacting on interrupt. The resident program is loaded in the operating memory and starts to run at some moment (e.g., when the hot key is pressed or the interrupt appears). The program can stay resident by INT 27H or by Disk Operating System (DOS) function 31H :

```
INT 27H         expects : DX = last address+1 to keep resident
                returns : none

function 31H    expects : AH = 31H
                          AL = exit code
                          DX = memory size to keep resident in
                               16-byte paragraphs
                returns : none.
```

INT 27H or DOS function 31H set the initial memory allocation to the size specified by register DX and return control to the parent process. DOS function 31H supersedes INT 27H which

does not send back an exit code and cannot install resident program larger than 64kBytes. It is possible to use the procedure Keep in Turbo Pascal. Procedure Keep terminates the program and makes it stay in memory.

DOS function 49H releases memory occupied by the TSR programs :

```
function 49H    expects : AH = 49H
                          ES = segment address ( paragraph )
                               of allocated memory to release
                returns : AX = error code if CF is set.
```

Before either COM or EXE format program is loaded, DOS (together with other actions) makes a duplicate of the current DOS environment for the program. DOS environment is an area of memory that holds a set of strings. The strings can be used by applications to obtain certain system level informations. If resident program needs no DOS environment, it is recommended that a TSR program sets free its environment block during installation. This allows the next program to use that memory block. It is possible to free memory in Turbo Pascal in the following way :

```
regs.AX:=49H;
regs.ES:=PrefixSeg:2CH;
MsDos(regs);                { release Dos environment }.
```

When TSR program ends its operation, it is necessary to release code from the operation memory. The following sequence deletes TSR program from the computer memory and the part of memory occupied by TSR program can be used by another application :

```
regs.AH:=49H;
regs.ES:=PrefixSeg;
MsDos(regs);                { release code }.
```

3. Interrupts

Our program changes vectors for several interrupts. One of the following hardware interrupts : 0BH (IRQ3), 0CH (IRQ 4), 71H (IRQ9, only for PC AT) or 0AH (IRQ2, only for PC XT) can be selected for the controller KK009. It depends on the switch at the controller board. Further TSR program changes the hardware interrupts 08H (timer), 09H (keyboard) and the software interrupts 28H and 80H. DOS functions 35H and 25H are used for saving of the original interrupt vector and for installing new one :

function 35H expects : AH = 35H
AL = interrupt number (00H - 0FFH)
returns : ES:BX = address of the interrupt vector

function 25H expects : AH = 25H
AL = interrupt number
DS:DX = address of the new vector
returns : none.

Turbo Pascal has for this purpose the two procedures, GetIntVec and SetIntVec.

Demands for hardware interrupts are accepted in IBM PC AT by the integrated circuit INTEL 8259A [2]. This circuit is programmable and can serve 8 interrupts. It also contains interrupt mask register (IMR). IRQ0 corresponds to the low significant bit of IMR and IRQ7 to the most significant one. The demand for hardware interrupt is blocked, if the bit of IMR corresponding to interrupt is logical 1. This is the reason why it is necessary to carry out unblocking of the corresponding bit for the interrupt used by the controller CAMAC :

```
in AL,21H          ; for IRQ9 address is 0A1H
mov AH,int_mask    ; e.g., int_mask for IRQ4 = 0001 0000B
not AH
and AL,AH
out 21H,AL         ; for IRQ9 address is 0A1H.
```

On the contrary, the following sequence blocks the corresponding hardware interrupt :

```
in AL,21H          ; for IRQ9 the address is 0A1H
or AL,int_mask     ; e.g., int_mask for IRQ3 = 0000 1000B
out 21H,AL         ; for IRQ9 the address is 0A1H.
```

The procedure for hardware interrupt should include at the end the sequence which sends End-Of-Interrupt signal to 8259A in-service register :

```
mov AL,20H
out 20H,AL         ; for IRQ9 address is 0A0H.
```

We should not forget in the new interrupt procedures about the old ones. Their addresses were saved and if it makes sense the original interrupt procedures are carried out when the corresponding interrupt appears (INT 08H, INT 09H, INT 28H). Generally, the interrupt procedure is always with the attribute FAR, return address and register of flags are saved in the stack. Return from procedure is performed by instruction IRET. Calling of the original procedure must fulfil these requirements and looks like this :

```
pushf              ; save flag register
call OldIntVec     ; OldIntVec =  segment + offset
                     address of the original procedure.
```

The following remark is concerned with using of the stack. You can define your own stack or use stack of the interrupted program. But if you use stack of the interrupted program, it is not possible to find depth of the stack. Overfilling of the stack means mistake (e.g., for operating system it is

DOS internal stack overflow fault) and the operating system stops. Two ways exist to solve this problem. The former way is to define your own stack and to control the depth of the stack. The latter way means not to allow nested hardware interrupts during data reading from CAMAC modules. In this case it is not possible to use in the procedure for serving CAMAC module instruction STI. Instruction STI sets interrupt-enable flag (IF). IF is automatically reset after interrupt acceptance and set after instruction IRET. So next hardware interrupt is blocked and stack cannot be overfull.

3.1. INT 08H (Timer)

This hardware interrupt is executed approximately in every 55 ms (accurately 1193180/65536 times in second). ROM BIOS code for the interrupt updates the clock value and turns off the diskette drive motors after about two seconds without read or write activity. We used it together with DOS re-entrancy flag (DosRF) for writing the data buffer to hard disk. Calling of DOS functions in TSR programs is impossible, if another DOS function is working. It means crash of the operating system. Writing to hard disk uses DOS functions and that is why it is necessary to test DOS re-entrancy flag before every saving of the data to a file. If DosRF is non-zero, it is impossible to call DOS functions :

```
push ES
push DI
mov ES,Dos_F_Seg  ; Dos_F_seg = seg. address of DosRF
mov DI,Dos_F_Ofs  ; Dos_F_Ofs = ofs. address of DosRF
test byte PTR ES:[DI],0FFH
pop DI
pop ES
jnz @exit
call WriteToFile  ; if DosRF = 0, write data to file.
```


Segment and offset address of DosRF can be found by DOS function 34H :

```
function 34H   expects : AH = 34H
               returns : ES:BX = address of flag.
```

INT 08H is also used for the time measurement. Some type of experiments require to measure exact time. Ordered time of the experiment is decremented in every call of INT 08H and if it is zero, the TSR program stops its activity. Dead time of the spectroscopy system can be found. It is possible to compare number of ticks which occur during procedure serving hardware interrupt from CAMAC module with one coming alone. Measurement of the dead time increases time of procedure for serving CAMAC module by 1-2%.

3.2. INT 09H (Keyboard)

IRQ1 is generated upon each press and release of a key. TSR program tests which key was pressed. If the key was F11 and the computer is in the text mode, the information window appears on the screen. It provides some basic informations about the program (name, simple help, status of the program, the remainder of the experiment time). The next press of the F11 key tells to TSR program to restore original contents of the screen. The hot key F12 was selected to stop TSR program.

If the key F11 was pressed, the original procedure for keyboard is blocked and the following code is needed to satisfy the keyboard interrupt :

```
in AL,61H       ; get value of keyboard control lines
mov AH,AL
or AL,80H       ; set the enable keyboard bit
out 61H,AL      ; and write it to the control port
xchg AH,AL
```

```
    out 61H,AL      ; write back original control port value
    mov AL,20H      ; send End-Of-Interrupt to 8259A
    out 20H,AL.
```

3.3. INT 28H

This software interrupt is executed by DOS as it waits for a keystroke and as a predecessor to DOS functions higher than 0CH. It is also trapped by a variety of TSR programs. When DOS calls INT 28H, it is safe for TSR program to use DOS functions (e.g., perform file I/O). Our program changes interrupt vector for INT 28H and if it is needed (e.g., data buffer is full) sets a flag. Then, upon execution of INT 28H, our interrupt procedure first calls the original INT 28H handler, then, if our flag is set, it executes the procedure for writing the data buffer to a file.

Unfortunately, INT 28H has some limitations. You must not use DOS functions less than or equal to 0CH and when DOS for a long time does not call any DOS function, INT 28H is blocked. Therefore, the second method mentioned above (INT 08H) for I/O operations is applied in the program.

3.4. User Interrupt

Sometimes it is necessary to control the measured data and the activity of the TSR program during the experiment. TSR program should be as small as possible not to occupy the memory of the computer. So a separate program for a graphic presentation of the measured data was written. Mutual data exchange is required between the TSR and graphic program. The data exchange can be performed by some unused software interrupt. INT 80H was chosen for this purpose. Originally, INT 80H is reserved for BASIC. When BASIC is non-active, it is free. INT 80H looks in TSR program like this:


```
{$F+}
procedure INT80(Flags,CS,IP,AX,BX,CX,DX,SI,DI,DS,ES,BP:word);
          interrupt;
begin
  case AX of
    0:
    1:
    2: begin
         AX:=seg(seconds);BX:=ofs(seconds);
       end;
  end;
end;
{$F-}.
```

Now if we want to know in the program for graphic presentation the time to the end of a measurement, we should perform in graphic program the following sequence:

```
    regs.AX:=2;
    Intr($80,regs);
    Number_Of_Seconds:=Ptr(regs.AX,regs.BX);
    write('Number of seconds : ',Number_Of_Seconds^);,
```

where Number_Of_Seconds is pointer to longint. It is also possible to direct the TSR program by changing the parameters between two programs (e.g., calling INT 80H with regs.AX:= 1 is used to clear the data buffer).

4. Expanded Memory

One of our TSR programs works with 16 amplitude-digital convertors (ADC). It saves to the memory 16 spectra, 4096 channels each. One spectrum is 16kBytes long and the TSR program demands only for data 256kBytes of memory. We decided to use for data buffers expanded memory to spare 640kBytes conventional memory address space. Expanded memory is the one

above 640kB limit. It uses an Expanded Memory Specification (EMS) compatible memory board and some software driver (e.g.EMM386.sys). EMS board creates four 16kBytes physical pages of addressable memory and up to 32MBytes total memory available to be swapped in and out. This 32MBytes memory is devided into 16kBytes logical pages. The direct access is only to the physical pages. All functions connected with expanded memory are provided by INT 67H with different contents of the register AH. First of all it is necessary to test the presence of Expanded Memory Manager (EMM). Program should take the address of the interrupt procedure for INT 67H, and examine offset 0AH from this address. It should contain the characters "EMMXXXX0". Next step is finding a segment address for physical pages (they are called EMS frame) :

```
INT 67H   expects : AH = 41H
          returns : BX = segment address
                    AH = EMM status.
```

The segment address is valid only for the EMM status=0 (no error). The TSR program also controls the number of EMS logical pages needed for the data buffer and number of EMS logical pages currently available. The program stops its activity if the number of logical pages needed for the data buffer is higher than one currently available. INT 67H, function 42H can be used for finding the number of non-allocated EMS logical pages :

```
INT 67H   expects : AH = 42H
          returns : DX = total EMS pages in system
                    BX = number of EMS pages that are
                         currently available
                    AH = EMM status ( it should be 0 ).
```

Using INT 67H, function 43H program opens EMM handle and allocates logical pages (each 16kB) :


```
INT 67H   expects : AH = 43H
                    BX = number of logical pages requested
          returns : DX = EMM handle
                    AH = EMM status ( 0 means no error ).
```

Every program working with expanded memory has unique EMM handle. It is used for following operations. As mentioned above program has addressable access only to the four physical pages. So INT 67H, functions 44H or 50H make one or four logical pages accessible :

```
INT 67H expects : AH = 44H
                  AL = physical page number ( from 0 to 3 )
                  BX = logical page ( 0 to n-1, where n is
                       the number of logical page allocated
                       by a handle )
                  DX = EMM handle
        returns : AH = EMM status
```

```
INT 67H expects : AH = 50H
                  DX = EMM handle
                  CX = count of 4-byte elements in array at
                       DS:SI
                  AL = subfunction number
                     = 0 : DS:SI = series of 2-word elements.
                           In each element, the first word is
                           logical page number and the second
                           is a physical one.
                     = 1 : the same as AL = 0,but second word
                           of each element is segment address
        returns : AH = EMM status.
```

But before calling one of these functions you should save the current mapping of physical to logical pages by INT 67H, function 47H. Only after that a resident program can page in

its own logical pages. The original mapping is restored by INT 67H, function 48H :

```
INT 67H expects : AH = 47H
                  DX = EMM handle of requesting process
         returns : AH = EMM status

INT 67H expects : AH = 48H
                  DX = EMM handle of requesting process
         returns : AH = EMM status.
```

At the end of TSR program it should be INT 67H, function 45H. It releases all logical pages allocated to handle :

```
INT 67H expects : AH = 45H
                  DX = EMM handle
         returns : AH = EMM status.
```

5. TSR Programs Test, Velocity Measurement

To test TSR programs working with interrupts there were written several resident programs serving different modules and experiments. The simplest program co-operates with ADC KA007 or KA011 and measures Pulse Height Analysis (PHA) spectrum. TSR program for ADC occupies in memory 31.4kB (including 16kB spectrum). In addition various versions of TSR program for ADC exist - 1 ADC, spectrum in expanded memory (occupied memory is 16.8kB); 16 ADCs, all data buffers in expanded memory (program needs 43.3kB).

TSR programs are now tested in several physical experiments. The first one is connected with the multidetector correlation device (MUK). The TSR program controls two connected data buffers KL006. The second TSR program was written for 4-detector system using in the measurements of the time differential perturbed angular correlations. System is based on modules KA 010 (QDC),

KA007 (ADC) and master module. TSR program creates during experiment 12 time spectra. The next TSR program supervises 16 HPGe detectors equipment for measurement of double beta decay. TSR program reads all data from three connected crates (1.crate contains 16 ADC - KA007, 2.crate 16 QDC - KA001 and 3.crate 16 TDC - KA304). All reading data are saved into the file and 48 spectra are formed during experiment. All spectra are saved in expanded memory.

computer	spectrum in mem.< 640kB		spectrum in EMS
	without EMM	with EMM	memory
386, 16 MHz	56	140	220
8 MHz	110	400	700
386, 20 MHz	12	70	370
10 MHz	25	200	1100
386, 25 MHz	12	33	140
8 MHz	24	100	370
286, 16 MHz	12		
8 MHz	20		

Table 1. The hardware interrupt velocities (all times are in microseconds)

The velocity of hardware interrupt serving CAMAC module KA007 (ADC) on different computers was also measured. The measurement can be done either by electronic equipment or by software. The results are given in table 1. The highest velocity was reached on computer with microprocessor INTEL 80 286. The velocity does not depend on the number of hardware interrupt. It is the same for interrupt IRQ3 or IRQ9, which is accepted by the second integrated circuit INTEL 8259A. Maximum velocity of the interrupts for ADC is approximately 35000 events/s (of course computer is then fully busy with hardware interrupt).

6. Conclusions

We see two major advantages in using this type of TSR programs. The first one is background activity of program, computer may be used for other tasks (e.g., compilation, editing) and together high velocity of events registration is achieved. The second one is work with expanded memory. TSR program occupies minimum of memory space below 640kB and it is possible to increase the volume of the saved informations during experiment.

We would like to thank V.T.Sidorov and A.V.Zernov for support of the work and for valuable advices.

REFERENCES

[1] Georgiev A., Churin I.N., A CAMAC Crate Controller KK009 for the Pravetz-16 and IBM PC/XT Personal Computers.

[2] Valasek P., Monolithic Microprocessors and Microcomputers, SNTL, Prague, 1989.

[3] Ryskunov A., Computer Press 4/92, p.3.

[4] Ryskunov A., Computer Press 5/92, p.53.

Received by Publishing Department
on March 24, 1993.