NUCLEAR INSTRUMENTS AND METHODS 96 (1971) 35–41; © NORTH-HOLLAND PUBLISHING CO.

A DATA ACQUISITION SYSTEM WITH MASS STORAGE

P. B. J. VAN ELSWIJK, R. ENGMANN, A. M. HOOGENBOOM and P. DE WIT

Fysisch Laboratorium, Rijksuniversiteit, Utrecht, The Netherlands

Received 29 April 1971

A data handling system consisting of a small computer with disk storage and magnetic tape has been developed for various kinds of nuclear spectroscopy experiments. The operator can specify the data structure and parameters of the experiment in a suitable code to the computer.

1. Introduction

At the moment, various computer based data processing systems for nuclear experiments are commercially available. Since in most of these systems the small computer is not equipped with mass storage peripherals, the available number of channels is limited by the core memory. In many nuclear research centres users have incorporated in their systems mass storage devices in order to remove this limitation. The magnetic tape is used for storage of raw data[1]) whereas the reduced data (e.g. spectra) is stored on a magnetic disk or drum[2–4]). A system which combines both features of magnetic tape and drum storage has been described by Alderson and Dawson[5]). In many systems, however, the applications have been limited to two-parameter experiments.

This paper describes a flexible system which can handle up to three pulse height parameters and incorporates both disk storage and magnetic tape. A sorting program (SPECTR) analyses the data according to instructions given in a special language (SPECTRAL). The instructions specify the experimental set up used for experiments such as Doppler-shift measurements, angular correlation measurements and analysis of data from position sensitive detectors in a magnetic spectrograph. These specifications are used under program control to organize proper data selection, reduction, display and various other types of output.

2. Configuration

The present configuration is shown in fig. 1. The central processor is a Control Data 1700 computer with 16 k words of core memory, each word containing

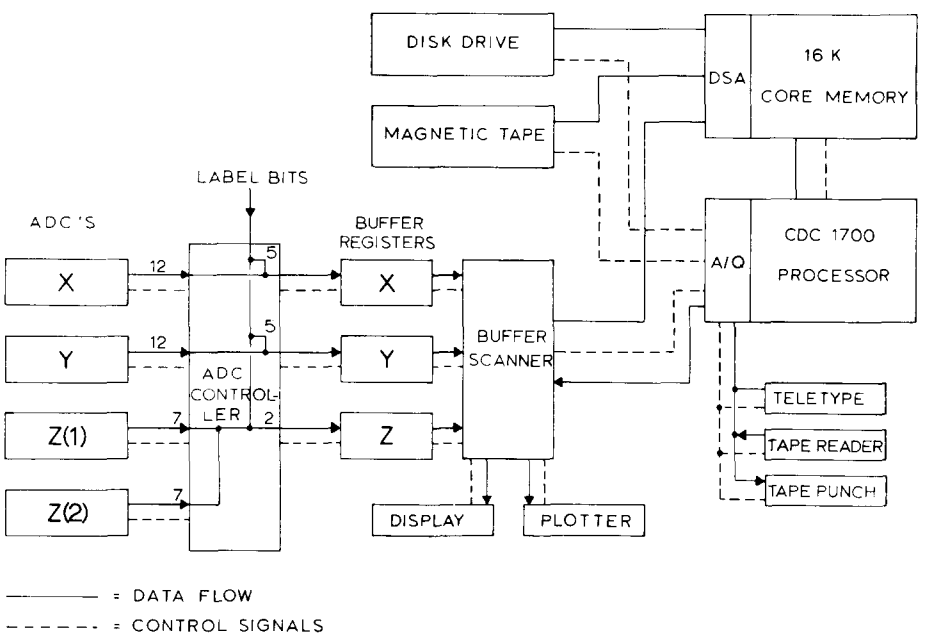

Fig. 1. Block scheme of the CDC 1700 computer system as it is used for on-line data acquisition. Lines with arrows represent unidirectional connections, the others are bidirectional.

16 data bits, a parity bit and a program protection bit. Peripherals of the system include one magnetic tape unit, a display device, an ADC interface and one moving-head disk memory, providing storage for 1.5 million words.

Two input/output channels are provided in the processor, one channel operates through the arithmetic registers (A/Q channel) and is used in controlling various low speed devices, the other channel provides high speed direct storage access. The latter is used by the display device, the disk, the ADC interface and the magnetic tape.

The four Laben ADC's are connected with the computer through the ADC interface which comprises an ADC controller, a buffer register system and a scanner. Of the four ADC's two are high resolution converters, providing up to 12 data bits and having separate buffer registers which are denoted with the symbolic names X and Y. Because a computer word contains 16 bits it is possible to add additional bits for labelling purposes. For example when more than one detector is connected to an ADC it is possible to have the analogue information accompanied by a bit pattern identifying a particular detector; this bit pattern becomes the label for that detector. The label together with the digitized analogue information is assembled in the interface and is stored in one word of core memory. The present software arbitrarily limits the number of labelling bits per ADC to 5.

The other two ADC's are 7-bit devices and are connected to a single buffer register with the symbolic name Z. The information of both ADC's is stored in one computer word together with two label bits. The 7-bit ADC's are used for converting timing pulses from time-to-analogue converters in coincidence experiments.

The ADC controller checks the presence of the end-of-conversion signals from the ADC's and controls the sequential transmission of the data, via the direct storage access channel, to a fixed 256-word buffer in the computer memory. This buffer receives coincident data in groups of four words, each group describing one event. Of these four words the first three contain the information from the various ADC's as described above. The fourth word remains empty and serves as a check on the operation of the interface.

3. Software

3.1. General

The processor is provided with a standard monitor system, the Mass Storage Operating System (MSOS) to which a few non-standard features have been added in order to incorporate the ADC interface in the system. The MSOS provides processor resource allocation on a priority basis, and is therefore particularly suited to real-time applications.

3.2. The program SPECTR

A software package SPECTR has been written to handle data from various kinds of experiments and attempts a unified approach to data processing for experiments in nuclear physics. SPECTR utilizes the real-time capabilities of the MSOS by assigning different priorities to different processes involved in the data handling, the general argument being that the more processor time a process requires, the lower its priority shall be.

SPECTR can operate in either on-line or off-line mode. In on-line mode data are received directly from the ADC interface; in addition all incoming raw data are dumped on magnetic tape for future reference. In off-line mode data are read from such a magnetic tape. Apart from this, however, there is no distinction between an on-line and an off-line operation. The process of copying the raw data on magnetic tape has the highest priority in order to minimize loss of data; the maximum event rate that can be processed without loss is approx. 1000/sec.

Both in on-line and off-line mode SPECTR sorts the incoming data into a number of spectra which are stored on the disk. In on-line mode with high event rates only random samples of the event stream are sorted because of the time involved in the sorting process. However, this is still sufficient to monitor the experiment. At any rate, the sorting can be repeated afterwards in off-line mode to process all the data. The sorting is performed according to user-supplied criteria as is described below.

3.3. The language SPECTRAL

3.3.1. *Introduction*

To introduce the large number of parameters required in a typical experiment it was necessary to develop a simple language, called SPECTRAL, with which the user can conveniently arrange and define his parameters in a kind of (pseudo-) program. Such a program will contain the specification of the type of coincidences, the number of channels per spectrum and the number of detectors for each ADC, the position of the labels, the definitions of spectra, and provisions for the output.

A SPECTRAL program consists of sequential statements, each occupying a single line of text. The language is based on English and is built around a number of *delimiters*, i.e. words with a reserved meaning. The use

of spacing, punctuation and words like "from", "to", "is" etc. is unrestricted so that the user has the option of writing either a skeleton statement or a statement resembling everyday English. The first word of a line is always a delimiter and determines the type of the statement. Delimeters may be abbreviated with their first 3 letters, for example a MODE statement could be written as:

MODE: COINCIDENCE OF X, Y AND Z

or as

MOD COI X,Y,Z.

If the first character of a line is an asterisk (*) the entire line is treated as comment and is ignored.

A SPECTRAL program logically consists of three parts:

1. Specification of the hardware configuration. This part pertains only to on-line operations, and may be omitted in the off-line mode, as this information is written on magnetic tape preceding the raw experimental data.

2. Definitions of the spectra to be generated. This part may be omitted in on-line mode, in which case only magnetic tape output is produced.

3. Output instructions. These may be omitted.

3.3.2. *Facilities*

A SPECTRAL program offers the following facilities:

1. To each ADC a symbolic name can be assigned; this name will then appear in all output concerning that ADC. An ADC name consists of a letter followed by up to 11 letters or digits. If no name is assigned to an ADC, it is referred to as X, Y, or Z. All references to an ADC can use either the standard name (X, Y or Z) or the defined name. The latter use is recommended because it greatly improves the readability of a SPECTRAL program.

2. All spectra will have a name for identification purposes. Each reference to a spectrum (e.g. in an output instruction) uses this name. A spectrum name consists of one letter followed by a number $\leqq 1023$, and is chosen by the user. It is possible to assign the same name to more than one spectrum. References to such spectra then use a subscript in parentheses to identify a particular one. The advantage of this feature lies in the possibility to refer to a number of spectra simultaneously. This avoids a lot of tedious writing.

3.3.3. *The hardware specification*

The hardware configuration is specified by the MODE, ADC and LABEL statements. A MODE statement should be the first statement in an (on-line) SPECTRAL program; it specifies which ADC's are coincident, and has one of the following forms:

MODE: COINCIDENCE OF A, B AND C

MODE: COINCIDENCE OF A AND B

or, of course, abbreviated forms. Here (A,B,C) is a permutation of (X,Y,Z). There are still other forms, which need not concern us here. The first form specifies a three-parameter experiment, the second a two-parameter experiment. For each ADC referred to in the MODE statement (except Z) an ADC statement, possibly followed by a LABEL statement, should follow. The ADC statement specifies the number of detectors (c.q. the largest possible label value) connected to an ADC and the number of channels or bits of its data. The form is as follows:

ADC B HAS n DETECTORS AND k BITS

ADC B HAS n DETECTORS AND k CHANNELS

also:

ADC B,n,k etc.

where B stands for X or Y. The rules are that n should not exceed 32, and that k is taken to be the number of bits, if $k \leqq 12$. If $k \geqq 128$, then k is assumed to represent the number of channels.

If the ADC statement specifies more than one detector, a LABEL statement should follow to indicate the positions of the label bits (needed to distinguish between the various detectors) in the 16-bit word containing the ADC data. This statement has the following formats (m and p are bit positions in a word):

LABEL m

LABEL m TO p

The first form is sufficient if $n = 2$ because one bit suffices to distinguish between two detectors. The second form means that the label is contained in the bits m through p.

To assign a name to an ADC the NAME statement is used, having the following format:

NAME "oldname": "newname"

Here "oldname" represents X, Y or Z, or a previously defined name; "newname" is an arbitrarily chosen name consisting of a letter followed by up to 11 letters or digits. When a name is redefined the old name is lost; however, the standard names X, Y and Z can always be used. Of course, it is not allowed to assign the same name to two ADC's.

3.3.4. *Spectrum definitions*

In SPECTRAL the spectrum definitions have a block structure. The beginning of a block is defined by a COINCIDENCE statement, the end by the next COINCIDENCE statement, or by a START or RETURN statement. The latter statements terminate the spectrum definitions. A block contains one or more

embedded blocks, starting with a DEFINE statement and defining spectra for a particular detector. A COINCIDENCE statement specifies a coincidence of two or three detectors which is valid for the entire block; therefore it is not necessary to specify detector numbers within a block, as these follow from the COINCIDENCE statement. The possible formats of this statement are:

COINCIDENCE OF A(k), B(m) AND C(n)

or

COINCIDENCE OF A(k) AND B(m)

where A, B and C are (standard or defined) names of different ADC's, A(k) is the detector k connected to ADC A. It is required that A, B and/or C be defined as coincident in the MODE statement, and that k, m and/or n agree with the corresponding ADC statements in being not larger than the specified number of detectors. Omission of a detector number implies detector number 1, i.e. X is equivalent to X(1). It is allowed to specify only two ADC's even if the MODE statement defines a threefold coincidence. In such a case the information from the unmentioned ADC is not used as a parameter in the subsequent spectrum definitions.

A DEFINE statement has the format:

DEFINE A-SPECTRA:

or simply:

DEFINE A

where A is the name of an ADC which should have appeared in the preceding COINCIDENCE statement. The statements following a DEFINE (until the next DEFINE) define spectra of the A-detector mentioned in the preceding COINCIDENCE statement. To specify division of ADC data in the subsequent spectra, the DIVIDE statement is used, having the format:

DIVIDE A BY B

or simply:

DIVIDE A

because the divisor always follows from the dividend, as only X/Y and Y/X are allowed. Division can only be used if X and Y are defined to be coincident.

The actual spectrum definitions (following a DEFINE statement) define a window and assign a name to the associated spectrum; they are an exception to the rule that all SPECTRAL statements start with a delimiter. In fact, the format is one of the following:

"name": SINGLE

"name": B FROM b_1 TO b_2

"name": B FROM b_1 TO b_2; C FROM c_1 TO c_2.

B and C are ADC names, and are required a) to be different from the ADC mentioned in the preceding DEFINE statement, and b) to agree with the preceding COINCIDENCE statement. The "name" is the name of the spectrum, consisting of a letter followed by a number $\leqq 1023$. When a spectrum is defined as SINGLE this means that, if the spectrum belongs to the ADC A, there are no restrictions on the B and C ADC's (apart from the required coincidence, of course).

3.3.5. *Example of a SPECTRAL program*

The following example shows a SPECTRAL program for lifetime measurements (see ref. 7). One gamma-detector is connected to the X ADC, two particle detectors to the Y ADC, and two time-to-analogue converters (TAC's) to the Z ADC. Each particle detector is associated with a TAC, so that the two particle detectors can be distinguished either by labelling the Y or Z ADC. The latter is simpler because label bits for the Z ADC do not need to be specified. The example exhibits the more or less abbreviated forms possible:

```
MODE: COINCIDENCE X, Y, Z
ADC X 1 DET, 12 BIT
ADC Y 1,512
NAME X: GAMMA
NAME Y: ALPHA
NAME Z: TIME
*
COINCIDENCE OF GAMMA, ALPHA, TIME(1)
*
  DEFINE GAMMA-SPECTRA:
    G1: SINGLE
    G2: ALPHA 260,305; TIME 24,46
*
  DEF ALPHA
    A1: SIN
    A2 TIME 24,46
*
  DEF TIME
    T1 SINGLE
*
COI TIME(2), ALPHA, GAMMA
*
  DEF GAMMA
    G1 SIN
    G2 TIME 22 TO 43; ALPHA 247,291
*
  DEF ALPHA
    A1 SIN
    A2 TIME 22,43
*
  DEF TIME
    T1 SIN
```

This example also exhibits the use of multiple names. When the program is compiled the first spectrum

named G1 is referred to as G1(1), the second as G1(2) etc. In the above example the subscript is even meaningful, as it corresponds to the TIME-detector number used in the definition. This is accomplished by judiciously writing down the definitions.

3.3.6. *Other features of SPECTR*

The operation of SPECTR starts with the compilation of a SPECTRAL program; during this suitable diagnostic messages are issued whenever necessary. When the sorting process is started the operator has continuous access to a number of facilities offered by SPECTR. These include: listing the presently defined spectra, generating a dynamic display of any desired spectrum, and output of spectra on magnetic tape, punched tape, teletype or incremental plotter. Another useful feature is the possibility to define new spectra during a run, again in the form of a SPECTRAL program.

The system has been very useful in analysis of coincidence experiments for choosing the parameter values in such a way as to obtain an optimal peak to background ratio. In practice this is done by replaying the magnetic tape several times in off-line mode (the processing of a full tape usually takes about 30 min). The resultant spectra are finally written on magnetic tape and are processed either by the same computer or by the University Computer Center.

4. Internal operation of the software

4.1. The sorting process

By appropriate instructions the user can instruct the program to sample spectra from the incoming data. These sorted spectra are stored in the disk memory, where each channel is written in double precision integer format, so that the maximum channel content is $2^{31}-1$. The total number of channels (summed over all spectra) is restricted by the available disk memory to about 600000 as part of the disk is used as Program and System Library. These 600000 channels are subdivided into spectra at the user's discretion, providing e.g. 600 spectra of 1024 channels each, or any mixture of spectra of different sizes.

Although in principle the channel contents on the disk can be augmented by reading its contents from disk, updating the number and writing it back, this is hardly feasible if a reasonable processing time is required, due to the large random access time (ca. 110 msec). For this reason the updating should be done systematically, taking both the position of the read/write head assembly and multiple references to the same channel into account. These requirements imply that references to channel entries should be ordered according to their physical position on the disk, and that for each channel reference a count should be maintained to enable updating with increments larger than one. These operations are performed in the core memory available for this purpose (about 6k), in the following way: the selected incoming events (those resulting in the updating of a particular word of the disk memory) are collected in a linear ordered list[6]). The elements of this list are groups of three consecutive words, with the format as shown in fig. 2.

The second word contains the disk address of the channel to which this element refers; the first word records the number of references made to this particular channel, i.e. the increment to be added to the contents of this channel; the third word contains a pointer (address) to the element containing the next higher disk address (or contains 0 if there is no such element, i.e. if it is the last of the list) and serves to maintain the ordering of the list. This list is therefore ordered according to disk addresses, a fact that is exploited by the actual updating process. The main advantage of the list structure is the fact that insertions and deletions preserving the order can be made without changing the address A of any element, but simply by changing one or two pointers, as is illustrated by the following example (fig. 3a,b). Consider the case that a new element has to be inserted in the list, and suppose that $a_1 < a_3 < a_2$. This implies that the new element's place in the ordering is between the elements containing a_1 and a_2. Now, if the new element is located at some address, say A3, the following operations effect the insertion:

a. set the pointer of element 3 to the value of the pointer of 1 and
b. set the pointer of 1 to the address A3.

Fig. 3b shows the resulting structure. This process covers all elements, the last one included.

The deletion of an element is essentially the reverse of this operation.

The actual sorting is carried out by the following process: an event is checked to determine whether it results in the updating of some channel. If so, the corresponding disk address is computed and the current list is examined to see if it already contains a reference to the same address. If this happens to be the case the

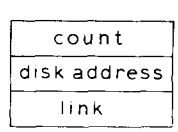

Fig. 2. Representation of a list element.

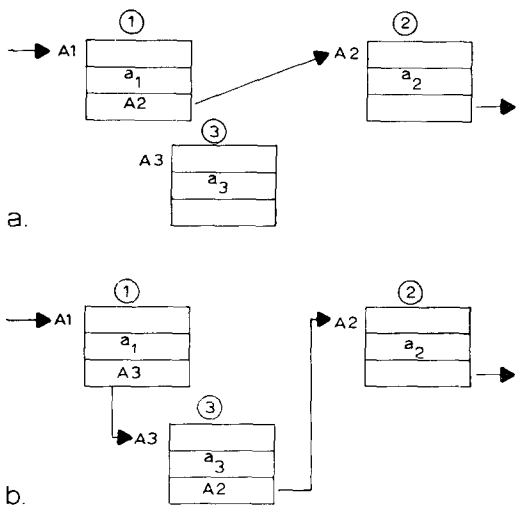

Fig. 3. Portion of the list before (a) and after (b) insertion of a new element.

corresponding count is incremented by 1 and the process terminates.

However, if no previous reference is recorded, a new element is created (referring to this channel) and is inserted in the appropriate position with its count set to 1. Previsions have to be made to process the overflow of a count or the overflow of core by the list.

Since the shorting is the most time-consuming process, it has the lowest priority.

An intermediate priority is assigned to the actual updating process. This process sequentially reads, updates and writes back areas of spectra as referred to by the elements of the update list, using the ordering contained therein. It attempts to minimize the disk access time by keeping track of the position of the read/write head assembly, and by reading large or small areas according to the distribution of successive references [i.e. two small areas are read if the distance between successive references is too large, and one larger area is read if the distance is less than some (rather arbitrary) limit]. The updating process deletes elements from the list whenever they have been processed, thus develops a dynamic equilibrium between the sorting and the updating. In practical cases this equilibrium lies at a list size well within the bounds of the available core memory. This means that in such cases the effect of the disk access time is effectively removed.

4.2. Generation of spectra

The internal representation of an event (cf. section 2) consists of three consecutive words in core memory containing labelled ADC information. The SPECTR software assumes that a label identifies a detector. This, however, is merely formalizing the way to speak about labelled ADC data, and does not preclude a different external convention; it simply states that the label bit pattern is interpreted as a number (ranging from 1 up to 32), called "detector number". The actual meaning may be quite different.

SPECTR refers to the three consecutive data words as X, Y and Z. In the following description the names A, B and C will be used where (A,B,C) is a permutation of (X,Y,Z).

The following notation will be used: A(k), pronounced as "detector k of A", refers to all those data of A, of which the label is interpreted as the number k. Then, with the above convention, an event can be described as a 6-tuple:

$$(k,a;m,b;n,c)$$

where each pair k,a etc. describes the contents of one word. This is interpreted as the coincidence of a signal a from detector A(k), a signal b from detector B(m) and a signal c from detector C(n).

The function of SPECTR is to sort the incoming events into spectra. One or more spectra can be generated for each detector (i.e. for each possible label value) with the information from the other detectors as parameters. Generally speaking for each detector A(k) a list of conditions (called spectrum definitions or windows) can be presented by the user, having the following form:

$$[B(m),b_1,b_2;C(n),c_1,c_2].$$

The meaning of this notation is as follows: the spectrum generated for the detector A(k) uses only such information from A(k) as is simultaneously coincident with signal b from B(m) and c from C(n), satisfying:

$$b_1 \leqq b \leqq b_2, \qquad c_1 \leqq c \leqq c_2.$$

The windows are allowed to overlap either completely or partially; each set of windows causes the formation of a particular spectrum. There is no inherent limit to the number of sets (apart from hardware constraints).

As two-parameter measurements are a special case in the above scheme, no special mention of this will be made.

Next to the sorting, SPECTR offers the possibility to divide the ADC data of the X and Y by each other and to generate spectra of the resultant quotients; both x/y and y/x are possible. This feature is very useful in experiments with position sensitive detectors. Such detectors produce two signals, one proportional to the energy E of the particle, the other proportional to

energy times position ($E \cdot x$). By division of these signals a position spectrum can be generated which has a much better resolution than the $E \cdot x$ spectrum in the case that the particles are not stopped in the detector and range straggling will cause peak broadening[8,9]).

The internal representation of the spectrum definitions again uses an ordered list structure: to each detector is assigned a list of all definitions pertaining to that detector.

The ordering of these elements is performed according to the detector number, and is used to decrease the time spent in searching through the list.

5. Results

The system has been used in a number of experiments were the Doppler shift attenuation method was applied to obtain life-times of excited states[7,10]). In these experiments coincidences are detected between γ-rays and particles [in e.g. (p,$\alpha\gamma$), (τ,pγ) reactions]. The low counting rate and the high background require complete data analysis including time-interval information between "coincident" pulses. Since the width and the position of the "time-window" depend on the energies of the detected particles and γ-rays a repeated reanalysis could substantially improve the real-over-random ratio for different parts of the γ-ray spectrum. Furthermore, the background contribution in the lower part of a γ-ray spectrum due to absorption processes near the surface of a Ge(Li) detector has been suppressed efficiently by selecting the proper time-window[11]).

The divide option used in the analysis of the particle spectra obtained through position sensitive detectors placed in the focal plane of a magnetic spectrograph simplifies the application of these detectors and improves their position resolution. Especially if a "mixture" of particles (e.g. protons and deuterons) is present of which one type (e.g. proton) is not stopped in the detector the divide option enables a separate analysis.

In general the almost unlimited number of channels and their well ordered application has greatly simplified and improved the data processing of the present experiments with an order of magnitude reduction in beam-time.

6. Conclusion

For high resolution experiments with more than one parameter a computer system is indispensable. Even small computers are sufficiently powerful if the system includes a random-access mass storage device and magnetic tape.

The authors wish to thank Dr. G. van Middelkoop for many valuable discussions and Mr. W. Coppoolse for building some of the interfaces. The project was partly supported by the joint program of the "Stichting voor Fundamenteel Onderzoek der Materie" and the "Nederlandse Organisatie voor Zuiver Wetenschappelijk Onderzoek".

References

1) J. C. Hiebert, C. C. Hamilton, T. H. Sathre and R. C. Rogers, Nucl. Instr. and Meth. **86** (1970) 45.
2) J. P. Gomidec, Nucl. Instr. and Meth. **88** (1970) 125.
3) J. P. Adam, Nucl. Instr. and Meth. **73** (1969) 89.
4) B. Souček, IEEE Trans. Nucl. Sci. NS-**17** (1970) 20.
5) P. R. Alderson and N. Dawson, Nucl. Instr. and Meth. **86** (1970) 35.
6) D. E. Knuth, *The art of computer programming*, vol. I (Addison Wesley, Reading, Mass., 1968).
7) R. Engmann, E. Ehrmann, F. Brandolini and C. Signorini, Nucl. Phys. A**162** (1971) 295.
8) R. Bock, H. H. Duhm, W. Melzer, F. Pühlhofer and B. Stadler, Nucl. Instr. and Meth. **41** (1966) 190.
9) P. M. Debenbam, D. Schuhard and K. W. Goodwin, Nucl. Instr. and Meth. **67** (1969) 288.
10) M. Ivascu, D. Popescu and G. van Middelkoop, Nucl. Phys., to be published.
11) M. G. Strauss and R. N. Larsen, Nucl. Instr. and Meth. **56** (1967) 80.